Safe robot compliance strategies in potential human clamping situations

handed in
MASTER'S THESIS

B.Sc. Fabian Hirt

born on the 13.06.1991
living in:
Rainweg 4
73572 Heuchlingen
Tel.: 017683538794

Human-centered Assistive Robotics
Technical University of Munich

Univ.-Prof. Dr.-Ing. Dongheui Lee

Supervisor:	Prof. Dongheui Lee, Matteo Saveriano, Ahmed Wafik
Start:	31.05.2017
Intermediate Report:	03.10.2017
Delivery:	28.05.2018

TECHNISCHE UNIVERSITÄT MÜNCHEN
Human-centered Assistive Robotics
UNIV.-PROF. DR.-ING. DONGHEUI LEE

May 12, 2017

M A S T E R ' S T H E S I S
for
Fabian Hirt
Student ID 03628889, Degree EI

Safe robot manipulation strategies in potential human clamping situations

Problem description:

Humans and robots will share more and more a common workspace. In the field of household robotics, the shared workspace is the respective household, in an industrial setting the robot shares the same workspace with humans in collaborative tasks. The interaction between robots and humans, yet alone the presence of a robot near a human, leads to safety concerns. In recent years, the safety of physical human-robot interactions (pHRI) have been studied, however, many scenarios in pHRI still require more attention.
The biomechanical aspect of the collision between a robot and a human was investigated by [2, 1]. The results lead to dynamic velocity constraints - depending on the geometry of the end effector (POIs) and the trajectory dynamics. Together with a collision detection and reaction scheme, the robot trajectory is non-harmful to humans. However, in highly collaborative scenarios, quasi-static contacts between robots and humans are often intended. Here, standard collision reactions, such as stopping the robot or switching into zero-gravity mode, are rather disruptive. Nonetheless, the robots trajectory have to be safe in these situations, especially when there is the danger of clamping human body parts.
The main aim of this thesis is to identify situations where such clamping injuries can occur and to develop a method to prevent those injuries. By considering the robots trajectory, its end-effector (and entire structure), biomechanical injury data and the environment, a control strategy should be formulated to make the robots movement safe. This control strategy should finally be incorporated in a complete robot control architecture and validated.

Tasks:

- Literature review on soft-robotics, physical human robot interaction, and robotic control.
- Selection of relevant industrial use-cases to base and evaluate the control scheme on.
- Distance calculations between robot and environment for robots trajectory.
- Formulation, implementation and evaluation of a clamping-conscious control scheme.
- Implementation of trajectory optimization strategies for safe robot movements. (*optional*)

Bibliography:

[1] Sami Haddadin, Alin Albu-Schäffer, and Gerd Hirzinger. Safety evaluation of physical human-robot interaction via crash-testing. In *Robotics: Science and Systems*, volume 3, pages 217–224, 2007.
[2] Sami Haddadin, Simon Haddadin, Augusto Khoury, Tim Rokahr, Sven Parusel, Rainer Burgkart, Antonio Bicchi, and Alin Albu-Schäffer. On making robots understand safety: Embedding injury knowledge into control. *The International Journal of Robotics Research*, 31(13):1578–1602, 2012.

Supervisor:	Prof. Lee Dongheui, M.Sc. Ahmed Wafik
Start:	XX.XX.2017
Intermediate Report:	XX.XX.2017
Delivery:	XX.XX.2017

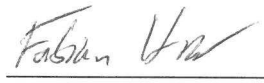
Confidentiality Clause

The master thesis on hand

Safe Robot Compliance Strategies in Potential Human Clamping Situations

has been done at Franka Emika GmbH. In order to comply with the non-disclosure agreement (signed to allow for the present research work), confidential data has been removed and/or changed from the report and the source code of the experiments and is accordingly noted. However, the underlying theory of this thesis, as well as most parts of the experiments are unaffected by this and can be reproduced independently.

Munich, 28.05.2018



Fabian Hirt

Abstract

When human and collaborative robots share their workspace, they manipulate objects together, exchange forces and they might even be in direct physical contact. In this context, the safety of human coworkers becomes a crucial aspect that classical collision avoidance cannot guarantee. This thesis introduces new concepts to minimize the potential physical harm that a cobot can cause in quasi-static manipulations, especially in clamping situations. Using anthropometric data, as well as a priori geometric knowledge about the environment, we propose a method to continuously monitor possible clamping situations, and a control scheme to reduce their hazards. Therefore, we elaborate several circumstances that must apply in clamping situations and demonstrate an algorithm that checks their fulfillment in real-time using a hierarchy of Bounding Volumes. Robot manipulation is rendered safe by limiting the contact forces adaptively in clamping situations.

Zusammenfassung

Menschen und kollaborative Roboter werden immer öfters gemeinsam Aufgaben in einem geteilten Arbeitsraum durchführen. Dabei kommt es vor, dass am selben Werkstück gearbeitet wird, Kräfte ausgetauscht werden und sogar direkter physischer Kontakt zwischen Mensch und Roboter besteht. In diesem Kontext ist die Sicherheit des Menschen essenziell, sie kann aber durch herkömmliche Methoden der Kollisionsvermeidung nicht gewährleistet werden. Die vorliegende Masterarbeit führt neue Konzepte ein, die die mögliche Gefahr von kollaborativen Robotern, besonders in Einklemmsituationen, miniert. Dabei werden anthropometrische Daten und Wissen über die Struktur der Umgebung genutzt, um in Echtzeit mögliche Einklemmsituationen zu erkennen und ihnen die Gefahr durch sichere Regelung zu nehmen. Dazu erläutern wir Gegebenheiten, die in Einklemmsituationen zutreffen und entwickeln einen Algorithmus, der diese Gegebenheiten mithilfe von Bounding Volumes in Echtzeit überprüft. Das sichere Verhalten des Roboters in Einklemmsituationen wird dadurch gewährleistet, dass die Kontaktkräfte limitiert werden.

Für Mama und Papa

Contents

Chapter 1

Introduction

In recent years, robots entered more and more human environments. Small mobile robots like robotic vacuum cleaners are driving autonomously through households, robot delivery services are tested, and industrial robotic arms are leaving their cages. This development is only possible due to active research in robot safety. Whereas small mobile robots do not pose a great threat to human safety, industrial robots can. Especially in future conceptions, powerful robots will work alongside humans. Therefore, it is mandatory to guarantee safety whenever robots and humans share the same space.

Today's *collaborative robots (cobots)* do already profit from the intensive research in *physical Human Robot Interaction (pHRI)*. Here, the greatest threats to ban are dangerous robot-human collisions. To accomplish this, cobots exhibit several safety features. Safe mechanical design reduces the danger potential of collisions [ASEG+08, SK14, SSK08, SSP+10, ZRKS04]. Dynamic velocity scaling and trajectory optimization lower the danger potential even further [HHK+12]. Finally, collision avoidance algorithms try to avoid any physical contact at all [DLF12, BK02, Kha86, KZB12, SHL17]. However, in highly collaborative scenarios, quasi-static contacts between robots and humans are often intended. Handovers, guiding, collective object manipulation or signaling intentions do often need interaction forces between the human and the robot. Standard collision reactions, such as stopping the robot, might be rather disruptive. More advanced collision reactions, like retracting or floating in zero gravity mode reduce the risk of injuries, but can compromise successful task execution [HASDLH08]. Special attention has to be given to scenarios where clamping a human is possible. If the human is not constrained in his motions, he can avoid high forces by yielding and escaping the contact situation (assuming slow robot motions, which are common for interaction tasks). When the human is clamped, this reaction is no longer possible and the contact forces can rise to harmful values.

The main goal of this thesis is to identify situations where such clamping injuries can occur and to develop a method to prevent these injuries. Identifying clamping situations requires knowledge about the robot, its trajectory, the environment and the human co-workers. Whereas the robot and the environment can be modeled quite accurately, there is in general a high uncertainty in the representation of human co-workers. Dealing with these uncertainties, as well as making the entire identification and control system real-time capable is posing a notable challenge. Therefore, we propose and evaluate a fast clamping identification method (based on Bounding Volumes Hierarchies) that accounts for uncertainties by assuming worst-case conditions. The danger of these identified scenarios is analyzed with the help of bio-mechanical injury data, which influences the clamping conscious control strategy. This control strategy limits the force that is exerted on the clamped body part, leading to safe robot manipulations. To evaluate this control strategy, it is firstly simulated and then used in an experiment with a 7 DoF manipulator interacting with human participants.

1.1 Outline

The remainder of this thesis begins with an analysis of *related work* on safe pHRI in Sec. 1.2. Chapter 2 first introduces the *concepts* of the proposed *Clamping Conscious Control* scheme and continues with a more detailed description of its *implementation*. The various components of the Clamping Conscious Control are then evaluated in chapter 3 by several *simulations* and *experiments*. After *discussing* the results and giving a glimpse on *future work*, the obtained insights are *summarized* in chapter 4.

1.2 Related Work

Research on safe *physical human-robot interaction (pHRI)* is distributed among different fields. The earliest research focused on the robot's mechatronic design and its control. Later, research extended to include also human biomechanical characteristics. All of those fields are important to prevent human injuries due to clamping. Safe mechatronic design lowers the potential danger of a robot and provides accurate and fast measurements of the robot's exerted force. Safe control uses the given information to create non-harmful trajectories and safe collision reactions. And finally, better understanding of biomechanical dynamics during human-robot collision leads to more realistic risk and harm assessments.

1.2.1 Mechatronic Design

Concerning the mechatronic aspect, the actual design choices are dependent on the robot's field of application. In cases where there is no (or very restricted) inter-

action between human and robots, task execution performance (like high accuracy, high workload and high speed) governs the design decisions; whereas in the application area of this thesis - pHRI - safety concerns have at least as much influence on the robot design as performance aspects. Most safety concerns arise from collisions between humans and robots in their shared workspace, hence much research was done to soften the impact of such a collision [AASB+06, HSAS+02, ASEG+08, BT04, Had13] or to prevent it altogether [FKDLK12, DLF12, BK02, Kha86].

One of the main concepts of softening collision impacts is to reduce the effective mass and inertia of the robot. This can be achieved by various means. Recent robots are designed using lightweight materials for their links (e.g. Bischoff et al.'s industrial arm [BKS+10], or the humanoids of Wyrobek et al. and Willow Garage [WBVdLS08, Gar17]). Other approaches reduce the effective mass and inertia by placing the heavy robot components (primarily motors) near the base. Consequently, these parts become either stationary (except for the motor rotation itself) or their velocity is limited. Timing belts, cables and pulleys usually transfer the motor torque to the links. Although these components are adding weight to the robot structure, they are normally not as heavy as comparable electrical motors. This relocation principle is for example used in [LKVS+10, QAN11, SK14].

Another important aspect is *"robot compliance"*, which was greatly shaped by pHRI research. Compliant robots react to unforeseen events and change their behavior accordingly. A popular example is collision detection and reaction. A non-compliant robot would not deviate from his programmed trajectory, even in the case of a collision. On the other hand, a compliant robot would sense the collision, possibly softening the impact by intrinsic (passive) compliant design, and would finally alter its trajectory according to a collision reaction scheme. Certainly, a perfect compliant robot would not even enter into an unwanted collision, since it would change its behavior accordingly beforehand, given that it possesses adequate sensors. Compliance can be separated into active and passive compliance. Active compliance is realized purely through robot control and needs appropriate sensors to comply to human contact motions. On the contrary, passive (intrinsic) compliance is mostly introduced into the robot by adding elastic elements that get deformed when colliding with objects. Robot researchers have evaluated various elastic elements and multiple locations where to add these elements in terms of robot safety and performance. Obviously, the links of the robot can be covered with soft material. However, this approach can restrict the robot's operational range and collision sensing is often aggravated, unless the robot's elastic cover possesses distributed contact sensors [AASB+06, SK16]. Adding elasticity into the robot joints proved to be more efficient. The elasticity in the joint decouples the motor inertia from the link inertia and reduces therefore the reflected inertia that is felt in a collision [AASB+06, BT04]. A scheme of the decoupling is shown in Fig. 1.1. Standard electrical motors need high speed and high gear reduction

ratios to produce sufficient torque for robot manipulation. With no decoupling, the reflected inertia, introduced by the motor, often surpasses the reflected inertia introduced by the robot's links. This is firstly due to the common lightweight structure of todays robots and secondly, because the reflected inertia rises with the square of the gear reduction ratio [ZRKS04]. The authors of [PW95] termed the elastic transmission in the joints *Series Elastic Actuator (SEA)* and argue that in addition to inertia decoupling, SEA also leads to more accurate and stable force control. Without elastic elements, force or impedance control have to be implemented purely in software and requires fast and accurate force sensors. With elasticity, the joint constitutes a passive impedance itself, which can be controlled to achieve a desired impedance (within bounds). To achieve this, the torque or rotor position before and after the elastic transmission have to be measured. Concerning robot durability, it is also reasonable to incorporate SEA, since the elasticity low-pass filters shock-loads. However, the actuator's output is low-pass filtered as well, reducing the mechanical bandwidth of the robot.

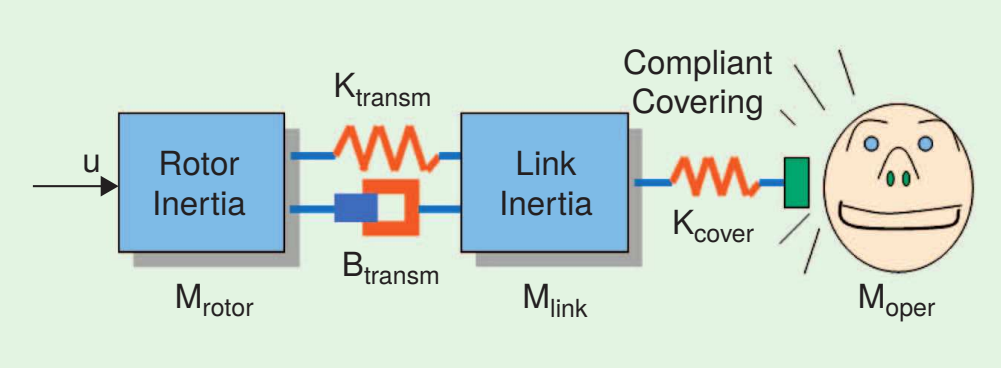


Figure 1.1: Decoupling of motor and link inertia. K_{transm} and B_{transm} represent the elasticity and the damping in the transmission. Figure copied from [BT04].

Zinn et al. remedy this performance degradation by introducing the *Distributed Macro-Mini (DM^2)* approach [ZRKS04]. Here, the desired torque signal is divided into a low-frequency (mainly arising from common manipulation task) and a high-frequency component (predominantly performing disturbance rejection). DM^2 uses a large low-frequency actuator (to supply sufficient power) and a small high-frequency motor (joint motor) in parallel to cover the entire torque bandwidth. The large motor is decoupled from the link with an elastic element (often cable or belt transmissions) and placed near the base to reduce the effective inertia. Since the large motor is only responsible for the low-frequency torques, the decoupling does not decrease the performance. The small-inertia motor is coupled to the link with high stiffness and is placed directly on the joint to perform small torque adjustments. As a unit, a DM^2 actuator is a near-perfect torque source with near-zero output impedance and high control bandwidth. However, there

are two major factors that deteriorate the performance. First, the joint motor can saturate when the disturbances are too big, and second, the high-stiffness coupling between the link and the joint motor introduces a disturbing oscillatory pole into the system. The latter is because the inertia of the joint motor is usually an order of magnitude lower than the inertia of the driven link. Zinn et al. outline possible approaches to these problems in [ZRKS04].

A different solution for safe and fast robot manipulation is the *Variable Stiffness Transmission (VST)* from [BT04] or more generally *Variable Impedance Approach (VIA)*, as summarized in [VASB+13]. The VST approach aims at solving the *safe brachistochrone* problem, which can be phrased as finding the fastest trajectory between two given configurations while not exceeding safety thresholds. When no safety aspects are considered, the fastest trajectory would consists of maximal acceleration at the beginning and maximal deceleration at the end. If the robot joints are stiff, the high velocity after the acceleration creates high reflected inertia. On the other hand, if the robot joints are elastic, the link's acceleration lags behind the motor's one, and the link oscillates after being decelerated. To optimize both safety and performance, the joints stiffness should be high during acceleration and deceleration, and low during high speed velocities, as shown in Fig. 1.2. Bicchi and Tonietti report improved performance compared to the DM^2 approach, when the controllable stiffness of the joints can be altered between $0.5\bar{\sigma}$ and $1.5\bar{\sigma}$, where $\bar{\sigma}$ is the stiffness center value [BT04].
To adjust the stiffness of a joint dynamically, Vorndamme et al. differentiate between three major designs: Spring preload, changing transmission between load and spring, and varying physical properties of the spring [VASB+13]. Numerous designs exists yet alone for the spring preload category, ranging from using a single spring, to a setup of antagonistic springs with antagonistic/ independent motors as in [ASEG+08]. However, all these designs need a second motor to adjust the stiffness, apart from the primary motor driving the link. For further details on different VST implementations, refer to [VASB+13].

As with SEA or DM^2, force control problems are translated into position control problems by incorporating the impedance of the joint, but higher bandwidth are possible with VST. This is because the controlled impedance bandwidth is centered around the system's natural impedance, which can be only altered with the VST approach. Additionally to the stiffness change in a VST system, a VIA system can also adjust its inertia and damping factors. Inherent damping is important to suppress the oscillations that arise from the joint elasticity. Multiple variable damping designs are elaborated in [VASB+13].

Instead of using elastic elements in joints, Shafer and Kermani research the use of Magneto-Rheological (MR) clutches to transmit the torques from the motor to the links [SK14]. MR clutches provide fully inertia decoupling, since no elastic or

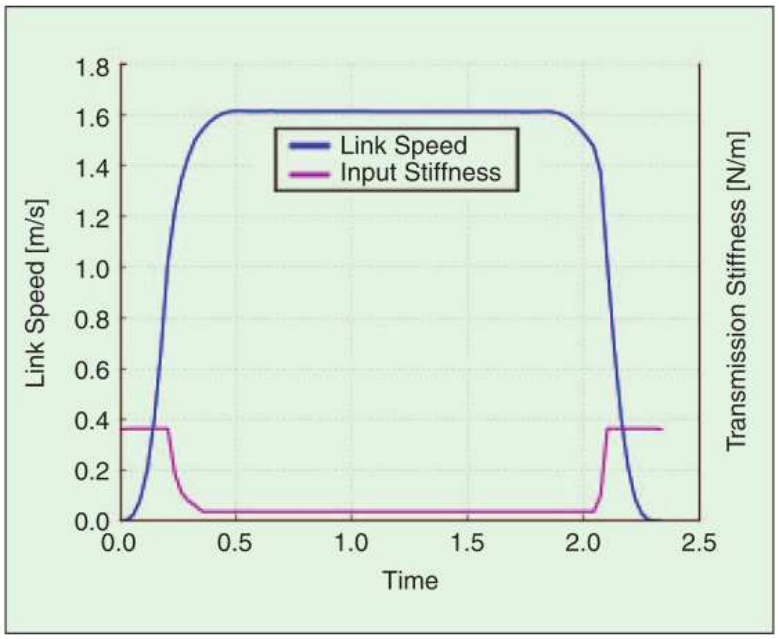


Figure 1.2: Optimal joint stiffness and velocity during a rest-to-rest task. Image taken from [BT04].

viscous forces act in the transmission. However, in their current design, the mass of the MR clutches is non-negligible, compromising their inertia decoupling effect on safety.

Additional safety designs become important when the entire robot structure and its environment are considered. Usually, robots operate under the influence of gravity. In order to compensate for the gravity load, the robot's motors have to produce constantly torque. Wyrobek et al. installed a gravity compensation system for the arms of a humanoid robot, consisting of springs, steel cables and small motors [WBVdLS08]. With this system, the arms can passively float through all arm-configuration, requiring only a small motor to adjust the leverage point of a spring. Since there are no counterweights introduced here, the reflected inertia of the robot setup is very low. Additionally, the joint driving motors can be downsized because they do not have to carry the arms' weight anymore.

A safe robot structure can further be simplified when high-frequency torque and accurate impedance control have only be provided at the robot's end-effector. In [ZRKS04, QAN11], DM^2 is only used for the three to four innermost joints of the robotic arm, the remaining outermost joints are actuated by small, high frequency motors. If the design of the last links is compact, the lack of elasticity in the last links does not diminish the safety nor the performance, but reduces the complexity and eases the control.

As shown by this summary, the mechatronic design of a robot manipulator influences highly its performance and safety. Most robotic structures developed for pHRI reduce the danger of dynamic human-robot collisions. For quasi-static scenarios, they allow reliable impedance or force control. Without elastic elements in the joints, a safe control scheme as presented in this thesis, would not be as effective.

1.2.2 Biomechanical Safety Aspects

Biomechanical criteria help to evaluate the severity of collisions between robots and humans. However, up to now, there is no consensus in collaborative robotic research about which biomechanical criteria to use in a specific collision scenario and where to set the threshold of tolerated injury or pain. One of the earliest approaches defined a universal contact force threshold by examining *pain tolerance limits* of different body parts ([YHH+97]). The resulting contact force threshold of 50 N is that low, because the authors wanted to have a universal threshold for all body parts. More differentiated analyses, based on the *onset of pain*, was done in [MMG14] whose results are incorporated in the ISO specification for collaborative robots [ISO16]. There, quasi-static pressure limits were developed for 29 different body parts.

A different viewpoint of tolerable thresholds for collision impacts is to consider resulting injuries instead of pain sensation. Here, the question is: what kind of injury is maximal to tolerate in human robot collisions? First answers to this question referred to the *Head Injury Criterion (HIC_{36})*, a well known metric in automotive crash tests. The HIC_{36} value is basically the integration of the head acceleration during 36 ms of the impact and is also referred to as a *Severity Index*. Since there are multiple Severity Indexes with different scales, they do not define a comparable injury measure. Therefore, they have to be mapped to broadly accepted injury classification metrics like the *Abbreviated Injury Scale (AIS)*, or, more famous in crash testing, the *EuroNCAP*[1], which is based on the former. The AIS range is depicted in Tab. 1.1. To analyze injuries in human-robot collisions, Haddadin et al. performed collisions experiments with robots and crash-test dummies [HASH07]. If not otherwise stated, the crash tests (or simulations thereof) were coupled with a collision detection mechanism, in order to command the robot to stop or retract when a collision is detected. The authors discovered that unconstrained blunt impacts at robot velocities of $2\frac{\mathrm{m}}{\mathrm{s}}$ pose no risk for humans according to the HIC criterion, independent of robot mass. Figure 1.3 depicts the HIC to injury level mapping and presents the values obtained for the Light Weight Robot III (LWRIII) robot. In the same work [HASH07], Haddadin et al. also analyzed the effect of unconstrained blunt impacts on the neck and chest of crash-test-dummies

[1] **Euro**pean **N**ational **C**ar **A**ssessment **P**rotocol

Table 1.1: Abbreviated Injury Scale. Table taken from [HASH07].

AIS	SEVERITY	TYPE OF INJURY
0	None	None
1	Minor	Superficial Injury
2	Moderate	Recoverable
3	Serious	Possibly recoverable
4	Severe	Not fully recoverable without care
5	Critical	Not fully recoverable with care
6	Fatal	Unsurvivable

and showed that these impacts are also sub-critical when crash-test injury metrics are used.

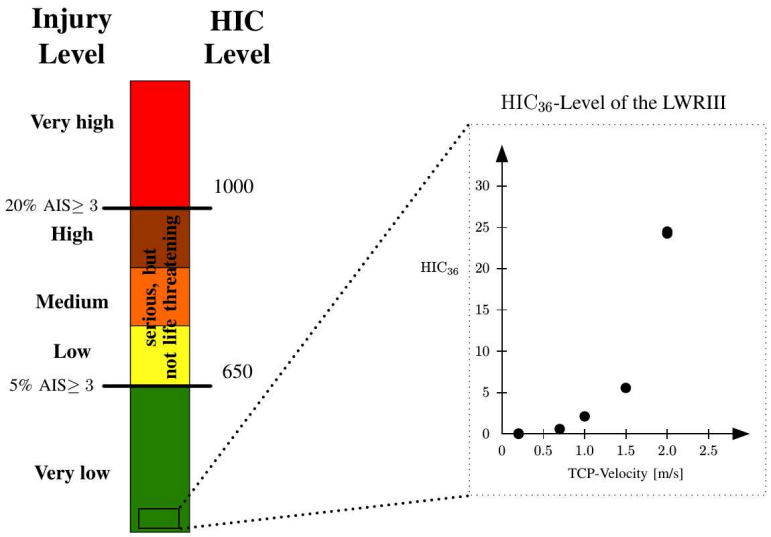


Figure 1.3: HIC and injury levels - together with EuroNCAP color code - for the LWRIII. Image taken from [HASH07].

In subsequent work [HASH08], Haddadin et al. argue that non-life threatening injuries, like fractures of facial bones, should limit collaborative robot operations, since they can already occur at robot's speeds and masses whose HIC values are considered safe. Assuming worst case stiffnesses (which lead to maximal energy transfer), the authors came to the conclusion that facial bone fractures can already occur at robot speeds greater than $0.5\frac{\mathrm{m}}{\mathrm{s}}$. However, with the exception of fragile bones, low-inertia robots need significantly higher velocities to cause bone fractures. *Severe* chest injuries are likely prevented when limiting the robot velocity to $4.5\frac{\mathrm{m}}{\mathrm{s}}$, even for high-inertia robots (e.g. a 2350 kg robot was simulated). For low-inertia robots, chest injuries in general can be avoided with a velocity limit of $3\frac{\mathrm{m}}{\mathrm{s}}$. Safety limits for other body parts can be looked up in the ISO specification for

collaborative robots [ISO16]. In addition to the previous mentioned - pain onset related - pressure limits, the ISO specification also includes force limits for several body parts. These force limits are injury based, in the sense that no injury above AIS 1 occurs.

Injuries become more severe when body parts are clamped between the robot structure itself or between the robot and the environment. The ISO specification for collaborative robots acknowledge this circumstance by setting the limits for free impacts twice as high as for clamped impacts. Clamped impacts can furthermore be subdivided into dynamic and quasi-static clamping. Dynamic clamping occurs when the robot collides with a constrained human body part with non negligible velocity, in contrast to quasi-static clamping, where the robot's velocity is negligible. In [HASFH08], possible injuries through dynamic clamping are examined by mock-up experiments and simulations. Here, one major factor for the severity of inflicted injury is the robot braking distance, which increases with higher mass and robot velocity. Thus, all examined industrial robots [2] (except the LWRIII) would fracture the facial frontal bone of a human at operational speeds of $2\frac{\mathrm{m}}{\mathrm{s}}$ when the human head is clamped - even though the robot is commanded to maximally brake when a collision is sensed. Weaker bones, like the Maxilla, would also be fractured with the LWRIII. Similar trends are shown for chest injuries. Only the LWRIII does not pose a thread to human safety in this setup, since its braking distance is significantly reduced due to collision forces. The time evolution of the impact forces even indicates that for a LWRIII - chest collision, it does not matter whether the chest is constraint or not, since the main impact is already over when the chest is pressed against the constraining element. As a conclusion, this type of impact cannot be rendered safer by introducing more elasticity in the robots joints. In subsequent work, Haddadin et al. further points out that this conclusion general applies to collisions with high stiffness contacts (e.g. head impacts) [HASH09]. A rather high intrinsic joint stiffness (like in the LWRIII) suffices to decouple the link inertia from the motor's in such an impact situation. In addition, no physical collision detection and reaction mechanism reacts fast enough to alleviate the force peak in such fast and rigid collisions. This can be seen in Fig. 1.4 where the collision detection signal (or r_4 respectively) triggers after the force peak is over. Even if the fast acceleration signal ($||\ddot{x}_{A1}||_2$) is used to trigger the collision reaction mechanism, no change in injury characteristics can be found, since the motor cannot decelerate fast enough. In some scenarios, high joint elasticity even contributes negatively to the robot's safe behavior, as described in [HASEH10]. The elastic elements can store non-negligible amounts of energy, which can be released during robot motion, increasing the robot's speed to unsafe values. On the other hand, if using this elastic energy by purpose to create peak velocities, the

[2] DLR LWRIII (14 kg), KUKA KR3-SI (54 kg), KUKA KR6 (235 kg), and KUKA KR500 (2350 kg)

robot can probably be designed with smaller motors to achieve comparable speeds to non-elastic robots.

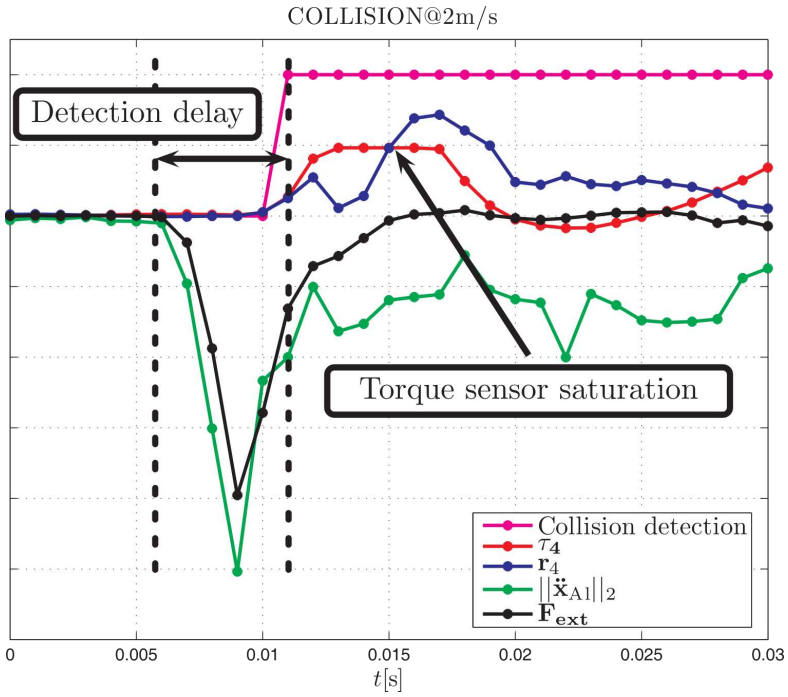


Figure 1.4: Impact characteristics of a collision between a dummy head and the LWRIII at $2\frac{\mathrm{m}}{\mathrm{s}}$. The signals are scaled to fit in a single plot such that their timings can be compared. An accelerometer on the robot's impactor was used to record the acceleration $\ddot{x}_{A1}$ of the impactor. For this experiment setup, τ_4 is the crucial joint torque and r_4 the external torque estimation. The collision detection is based on the *generalized momentum*. Figure copied from [HASH09].

Quasi-static clamping situations can in general be rendered safe when the collision is reliably detected and the robot is stopped in such an event. However, if the collision detection has high torque / force thresholds, even light-weight robots as the LWRIII can severely injure a human near singular robot configurations. Collision detections thresholds are often implemented as a percentage of the maximal exertable force, which goes to infinity at singular configurations. As a result, the collision detection threshold can achieve values which are above maximal tolerated values, leading to fractures or worse, as stated in [HASH09]. These injuries can be prevented by either limiting the allowed robot workspace to non-critical configurations, or by setting the collision detection threshold to very low percentages (2% in the case of the LWRIII).

The above experiments and injury analyzes consider blunt impacts, where major injury sources are fractures. Robot-Human collision with sharp contours or tools

however lead predominantly to skin or soft-tissue injuries before fractures occur. Therefore, Haddadin et al. analyze impacts with sharp tools like screwdrivers, steak knifes or scalpels in [HASH+11]. The collision scenarios are prevalently constrained stabbing and cutting, which are carried out with a pig leg cadaver. The penetration depth is used as severity index and is compared to the depth of vital organs in humans. The experiments show that with a suitable collision detection and reaction mechanism (together with a light-weight robot design) the injuries can often be mitigated or even prevented at all. Soft-tissue stabbing injuries could be prevented for the scissors or the kitchen knife (for a velocity of $0.16\frac{\mathrm{m}}{\mathrm{s}}$), whereas the scalpel penetrates the skin to life-threatening depths. For cutting motions, the severity of the injuries are mostly governed by the cutting velocity and the blade length. However, even at a cutting velocity of $0.8\frac{\mathrm{m}}{\mathrm{s}}$, injuries could be avoided at all with collision detection.

In industrial scenarios, robots' end-effectors, or the tools they carry, have often geometries that are between blunt and sharp. For these industrial relevant geometries, Haddadin et al. conducted further impact analyzes in [HHK+12]. Different impactor geometries were chosen of different sizes and masses (spheres, cuboids, pyramids and wedges) and they were dropped from different heights (leading to different impact velocities) on pig abdominal walls, simulating human skin tissue. The resulting injuries were classified according to the *AO classification*[3] and the severity *IC2* (contusion without skin opening) was chosen as maximally tolerable injury. With this experiment data, safety curves were created that give a safe velocity threshold for a given impactor geometry and its mass. By matching a robot's end-effector to geometric primitives, the authors were able to scale down adaptively the velocity of arbitrary robot trajectories to safe values.

To conclude the revision on biomechanical safety aspects, Fig. 1.5 gives an overview of the classification of physical human robot collisions and provides relevant injury criteria and their most crucial factors. For the main scenario investigated in this thesis, quasi-static clamping in non-singular configurations, the contact force $\boldsymbol{f}_{\mathrm{ext}}$ due to the robot's torque $\boldsymbol{\tau}$ affects the danger of contacts the most.

1.2.3 Control

Most standard industrial robots are used for simple trajectory orientated tasks, like spot welding or spray painting. They are normally position controlled, which suffices, when there are no uncertainties in their working environment [Cra05]. When industrial robots have to exert forces on the environment, as in scraping or polishing tasks, a hybrid position/force control, as in [FMR97], can also be used.

[3] Arbeitsgemeinschaft für Osteosynthesefragen

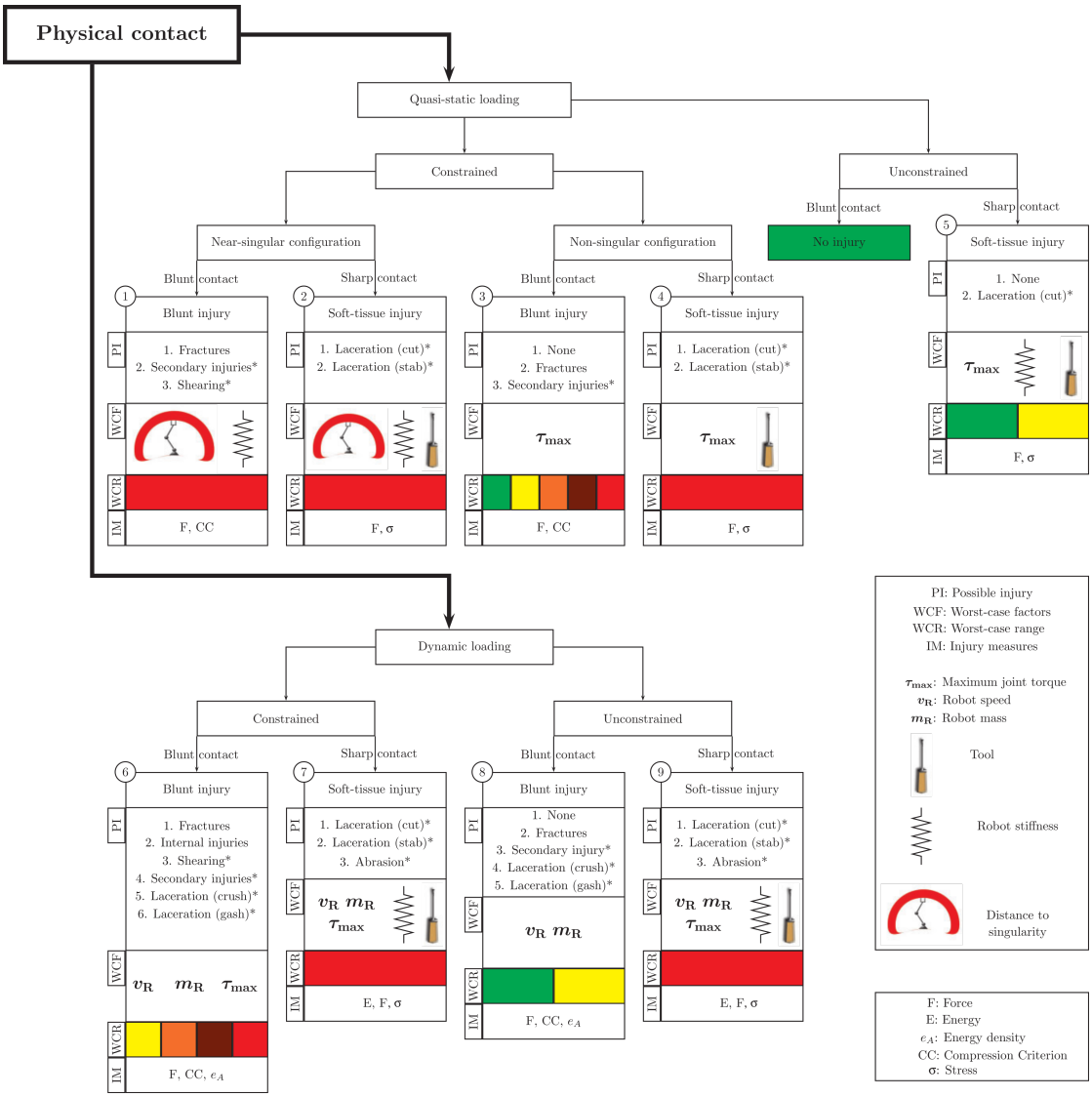


Figure 1.5: This safety tree shows for different physical contacts possible injuries (*PI*), worst-case factors (*WCF*), worst-case range (*WCR*) - expressed in the EuroNCAP color code (see Fig. 1.3) - and injury measures (*IM*). Figure copied from [HASH09].

However, the hybrid position/force control becomes unstable when contact to the environment is lost. In contrast to industrial robots, collaborative robots operate in unstructured environments and need to ensure the safety of human coworkers. Therefore, a compliant control scheme has to be implemented [TdVS14]. The most famous compliant control scheme is *impedance control*, introduced by [Hog84]. Here, the robot's pose is modeled as spring-mass-damper system and its equilibrium position is controlled together with the desired impedance dynamics. In general, impedance is the relation between an emerging force due to a combination of displacement, velocity and acceleration. So, if the equilibrium position of the robot's end-effector is deflected (e.g. in a collision with a human), the robot tries to get back to the equilibrium position by applying a force depending on its impedance dynamics. A pure position controller shows infinite impedance, which means that the robot would exert infinite force to get back to the equilibrium. The lower the controlled impedance, the less force is applied to objects in the robot's trajectory. The imposed impedance can be changed according to different robots tasks and safety requirements. If high impedance is requested, Calanca et al. suggest to use *admittance control* instead, which is closely linked to impedance control [CMF16]. The difference in these control strategies is that in impedance control, the environment is modeled as an admittance (displacement/ velocity/ acceleration output in reply to force input), whereas in admittance control, the environment is modeled as an impedance (force output in reply to displacement/ velocity/ acceleration input). Impedance control was continuously researched for stiff joints [Hog84], flexible joint robots [ASOFH03, ASOH07], and also for hydraulic arms [VSTH16].

Many collaborative robot tasks can be split into subtasks that require different impedance dynamics each. This is especially the case when high precision tasks alternate with tasks that need high compliance. Whenever the robot can stop its motion between two such subtasks, the controller can switch between stable sets of impedance parameters without affecting the overall stability. However, some tasks need to change these parameters during the ongoing robot's motion, e.g. when inserting a needle in a multilayer object as in [FSF13]. Imposing a varying impedance dynamics is the goal of *variable impedance control*. Here, additional stability constraints arise, since the change in impedance parameters can inject additional energy into the system rendering the overall system instable. Kronander and Billard state stability conditions for time varying stiffness and damping profiles in [KB16]. When a desired, a priori known, impedance dynamics satisfy these conditions, it can be implemented by a standard impedance control architecture. Ferraguti et al. show a tank-based approach to variable stiffness control when the desired stiffness profile is not known a priori [FSF13]. The controlled system is extended with a tank that stores the energy that is dissipated by the system. When a change in stiffness requires additional energy, this energy is taken from the tank. If there is not enough energy left, the stiffness change cannot take place.

It is thus guaranteed that a stiffness change adds no additional energy into the overall system.

The concept of energy tanks is applied to hybrid impedance/ force control by Schindlbeck and Haddadin in [SH15]. Compared to standard hybrid position/ force control, this control scheme does not require a separation between position and force controlled coordinates. It rather adds the force tracking as a supplementary control input to impedance control. The energy tank is needed to assure passivity, since the force tracking can potentially inject too much energy into the system. As accurate force tracking is then only possible with sufficient energy content, the energy tank is pre-filled with the estimated energy needed for the manipulation task.

A different approach to force tracking in combination with impedance control is introduced by Lee and Buss in [LB08]. Here, the stiffness of the impedance dynamic is adapted based on the force tracking error. Compared to other indirect force tracking approaches that replan the desired trajectory, this method does not need estimates of the environment's stiffness or location.

A control scheme specifically created to guarantee *pre-collision* safety is described in [HZ03]. Pre-collision safety assures that the collision forces at the time of impact are below a certain threshold. This can only be achieved by either assuring no collision at all, by safe mechanical design, or by limiting the velocity before the robot collides. *Post-collision* safety on the other hand deals with the exerted forces after the impact and can be influenced by safe collision reaction mechanisms. The pre-collision control scheme of [HZ03] introduces the notion of *impact potential*, which is the maximal force the robot can exert in its current state when colliding with a stationary object, indifferent of where on the robot's structure the collision occurs. The safety core of the control scheme asserts that this impact potential stays below a threshold. This is achieved by filtering and potentially modifying the torque commands that are sent by a regular motion controller (e.g. position controller). The safety core projects the threshold of the impact potential into the torque space and checks whether the commanded torque is within the convex space of safe torques. If yes, the command is passed through to the motors. If not, the safety core computes the closest safe torque to the commanded one by projecting the commanded torque onto the convex hull of safe torques. To improve controller performance, the safety core also returns the distance between the unsafe and the computed safe torque to the controller. To achieve realtime performance of the control scheme, there were only two control points picked on the robot structure in [HZ03]. Hence, it was guaranteed that the impact potential of these two points were below a certain threshold.

Concerning post-collision safety, several collision reaction strategies are evaluated in [DLASHH06, HASDLH08]. For various collision setups[4], the (estimated) external torques $\boldsymbol{\tau}_{\text{ext}}$ that are caused by the collisions, are analyzed. These torques are reduced the fastest when using the estimated torques $\boldsymbol{\tau}_{\text{ext}}$ to establish an admittance control, letting the robot retract from the contact. For high velocity impacts, stopping the robot immediately after sensing the collision reduces the external torques as fast as the admittance control strategy. However, as the robot does not retract from the contact, the external torques do not vanish.

To limit external contact forces, Haddadin et al. proposed *trajectory scaling* for position controlled robots in [HASDLH08]. In essence, the time as seen by the trajectory handler can be slowed down, sped up or even reverted, depending on the external torques or forces. With this indirect force control scheme, the robot's end-effector (or in general the controlled point) is constrained to its trajectory path while simultaneously limiting the forces that it exerts on the environment.

This thesis is mostly concerned about *post-collision* safety, because the dynamic effects of collisions play a minor role in quasi-static scenarios. As impedance control shows good results in unstructured environments, we employ a new variable impedance control scheme to limit steady-state contact forces in clamping situations. In contrast to velocity scaling, we allow the robot to deviate from its trajectory path in order to circumvent clamping situations.

1.2.4 Collision Avoidance

Most robots that are used in the industry, are separated from their human co-workers by physical barriers. The reason behind this separation is to avoid human-robot collisions completely. However, in modern collaborative robot (*cobot*) setups, physical barriers obstruct the collaboration highly. If collisions are regarded as disturbing or even dangerous, it is necessary to implement collision avoidance strategies without physical barriers for such collaborative setups. The international standard for collaborative robots [ISO16] gives two specifications that avoid human-robot collisions: *safety-rated monitored stop* and *speed and separation monitoring*. The safety-rated monitored stop is similar to using physical barriers, because as soon as a human co-worker enters the workspace of the robot, the robot motion should stop. It is therefore impossible to approach the moving robot. Speed and separation monitoring assures that there is always sufficient space between the robot and the human, the so called protective separation distance. If the robot is always at least this distance apart from any human in the workspace, it is guaranteed that the robot can stop its motion before colliding with

[4] I.e. LWR III colliding with clamped balloon, crash-test dummy, 1 DOF collision test bed or human subjects.

them. As a result, the protective separation distance is non-trivial to compute. It is for example dependent on the velocity of the robot and the human, as well as on sensors' reaction times and measurement tolerances. If the distance between human and robot falls below the protective separation distance, the standard demands the robot to stop immediately. As the name suggests, speed and separation monitoring is only surveilling the collaborative task, but it does not change the robot's motion, except for stopping it in dangerous situations.
More dynamic collision avoidance schemes have been investigated in the past and are still an ongoing topic of research. The method of Artificial Potential Fields [Kha86], for example, assigns attracting forces to the robot's goal position and repulsive forces to obstacles. The result is a potential field, on which the robot moves following the negative gradient. This potential field can be updated depending on the movement of the obstacles and thus constituting a dynamic collision avoidance approach. However, the main drawback of this method is that the robot can get stuck in a local minimum, although there might be a collision free path to the goal position. More recent collision avoidance approaches consist often of both an offline planning phase and an online reactive obstacle avoidance scheme. The elastic strips method [BK02] allows customizable obstacle avoidance motions by performing the collision avoidance in the nullspace of the task (if possible). Hence, the resulting motion can, on top, also avoid singularities or self-collision. Similarly, collision avoidance methods based on Dynamic Systems, as in [KZB12, SHL17], also change the globally defined trajectory locally when there is an obstacle.

Clamping situations can be avoided by implementation of collision avoidance policies. With the clamping identification algorithm presented in this thesis, the collision avoidance policy can be triggered selectively in these situations. However, we choose not to avoid collisions, but to render the collision safe by limiting the contact forces.

1.2.5 Distance Calculation

One requisite for reliable collision avoidance is a fast distance calculation between robot and obstacles. Since the geometry of the robot and the obstacles can be of arbitrary complexity, it is common to approximate these geometries with simpler ones. In [FKDLK12] for example, the robot is replaced by a chain of spheres. Depending on the application, such an approximation can be adequate or too coarse. More accurate distance computations, and especially (geometrical) collision detection algorithms, can be found in the field of physics simulation engines, which can be also transfered to robotic applications. Modern physics engines like *Bullet* or *ODE* detect collisions through a collision detection pipeline that consists basically of two steps [Wel13]. In the *broad phase* the objects that can possibly enter into collision are enumerated. The subsequent *narrow phase* consist of two more steps: *filtering* and *primitive collision checking*. The filter-

ing operation rejects all collision candidates that are not geometrical close to each other. The remaining collision candidates are then accurately checked for collisions in the primitive collision checking step. Thus, this pipeline architecture reduces to number of expensive primitive collision checks [5]. However, it is only efficient if the objects are represented by adequate data structures which enables fast proximity checks in the filtering step. Most often, the objects are represented by a *Bounding Volume Hierarchy (BVH)*. To check whether two objects are in collision to each other, both their hierarchies are traversed from top to bottom (coarse to fine). *Bounding Volumes (BVs)* that are separated are rejected and only the BVs that are not separated on this level are further traversed. Depending on the type of bounding volume used, these separation tests are more or less expensive. Common BVs are spheres, *Axis Aligned Bounding Boxes (AABBs)*, *discrete oriented polytopes* (*k-DOP*) and *Oriented Bounding Boxes (OBBs)*. Spheres and AABBs allow for example easy separation tests, but they often do not fit tightly on the object they approximate. K-DOPs and OBBs on the other hand are more customizable and approximate the objects better, but separation tests are more complicated. This trade-off can be quantified by following formula from [He99]:

$$T = N_v \times C_v + N_p \times C_p + N_u \times C_u + C_0,$$

where T is the total cost of a collision check, N_v is the number of BVs that undergo a separation test, C_v is the cost of this separation test, N_p is the number of (fine) primitive checks that have to be done and C_p are their costs. When the objects are moving, some of the BVs have to be updated. For example, AABBs have to be freshly calculated when the object is rotating. The number of BVs that must be updated is represented by N_u and the update cost is C_u. C_0 is a "one-time" update cost for each object for applying the movement transformation. Gottschalk et al. show in [GLM96] that *OBBTrees* (a tree structure of OBBs) perform better than AABB- or spheretrees in close proximity situations. The authors also developed a separation test for OBBs that is based on axial projections of the OBBs and on the *Separation Axis Theorem (SAT)*. As a result, the algorithm is one order of magnitude faster than previous separation tests, needing on average only about 100 operations.
Some of the algorithms used for collision detection, also output the *minimum distance* between two non-colliding objects, or the *penetration depth* for colliding obstacles. The penetration depth is often defined as the smallest vector that separates two overlapping geometries. The *Gilbert-Johnson-Keerthi (GJK)* algorithm [GJK88] is such an algorithm that works with convex hulls of objects and is therefore also applicable to most bounding volumes.

Instead of using an approximation that encapsulates the object, the data structure in [WZ09] approximates the object as close as possible from the inside with

[5] These checks depend on the representation of the object. If it is a point cloud, the checks are point to point distances. If it is a triangular mesh, triangles are checked for collisions.

a hierarchy of non-overlapping spheres. This *inner sphere tree* enables proximity and penetration volume queries. The distance returned from the proximity query constitutes an upper bound on the distance: Since the inner sphere tree is a non-encapsulating approximation, the real minimal distance can be smaller, but not bigger than the calculated one. Together with an encapsulating BVH, it is possible to get both an upper and a lower bound on the minimal distance between two objects. In [MPT05] the static environment is represented in a voxel map and the moving objects are represented as point sets. The authors termed the algorithm that leverages this combination of data structures *Voxmap PointShell*^{TM} algorithm. It is mainly developed for haptic rendering purposes, since it computes a short range force field around static objects, giving force feedback to a moving object (probe). However, an advancement of this method, described in [SSLeS14], is able to compute distances and contact manifolds, too. In an offline process, every voxel in the voxel map gets the shortest distance from its center to the static objects assigned. As a result, the voxel map represent a distance field. The points in the point shell (surface of the moving object) are packed into a sphere tree. By traversing the sphere tree, only promising points of the point shell are taken for collision or distance queries. During such a query, the algorithm determines in which voxel the examined point of the point shell lies. The returned minimum distance is composed as $d = v(P) + \boldsymbol{n}\boldsymbol{e}$, with $v(P)$ being the distance value saved in the respective voxel, $\boldsymbol{n}$ being the point's normal vector and $\boldsymbol{e}$ being the distance vector from the point to the center of the voxel. All points with negative distance are saved as a contact manifold and the reactive forces are a simple multiplication of the distance value with the point's normal.

The above geometrical collision detection and distance computation algorithms assumes known models of the objects in the scene. In many dynamic robot applications, and especially in collaborative scenarios, the objects in the scene are not known beforehand, much less their pose. The robot rather has to percept the objects with appropriate sensors. Most recent collision avoidance methods deploy visual sensors to gather information about the scene. Especially the present-day availability of affordable RGB-D cameras made them one of the preferred sensors for collision avoidance. The sensor's information can then be used to construct models of the objects and/or to update their pose. The works of [FKDLK15, SHL17] show that fast and accurate distance computation can also be done directly on the RGB-D camera's depth images. Not using any object models, the approaches employed a worst case treatment for areas that cannot be seen from the camera, e.g. because these areas are occluded by other objects or the robot itself. The use of multiple RGB-D cameras, as in [FDL17] remedies this drawback.

To identify clamping situations, we utilize distance computations between the robot and the environment. As we do not want to limit this algorithm to static

environments, we adapt a dynamic collision detection pipeline to our needs instead of using an implementation of the *Voxmap PointShell*^{TM} algorithm. Especially in contact situations, vision based distance computations have the drawback of occluded structures and do therefore pose additional challenges for identifying clamping situations.

1.2.6 Collision Detection

If a collaborative robot system does not implement collision avoidance, it is important that the system can detect collisions reliably.[6] Even if a robot system uses a collision avoidance method, collision detection and reaction methods can still be used as fall-back safety features. Robot collisions can obviously be detected by using appropriate additional sensors, like covering the robot structure with a sensitive skin [PCW12]. But it is also possible to detect contacts without additional sensors. Industrial robots are already using measurements of the actual currents in the electrical motors to detect unforeseen collisions by searching for sudden variations. Geravand et al. extend this basic method by using more accurate trajectory based thresholds [GFDL13]. Through filtering the current measurements, the industrial robot is also capable of distinguishing between intended human-robot contacts and accidental collisions. Robots that are equipped with joint torque sensors can also compare these measurements with the nominal torque that can be computed with the robot's dynamic model. A substantial difference between these values implies that an external force is influencing the joint torques. The drawback of this method is that the computation of the nominal torque needs acceleration data, which is subjected to noise due to numerical differentiation [Luc16].
Recent research has focused on a collision detection method that uses the *generalized momentum* $\boldsymbol{p} = \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$, where $\boldsymbol{M}$ is the robot's mass matrix, and $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are the joint positions and velocities. The works in [DLM05, DLASHH06, MFDL14] do all compute a *residual vector* based on the generalized momentum that can be seen as a filtered version of the joint torques that are generated by an external contact force. It is therefore not only possible to detect collisions, but also to partly identify on which link(s) the contact occurs. Compared to previous methods, no inversion of the mass matrix $\boldsymbol{M}$ nor acceleration information $\ddot{\boldsymbol{q}}$ is needed. Additionally, this method is independent on the robot control scheme; a change in the control strategy does not pose a problem. In [DLM05], the residual vector is used to create a hybrid force/position controller in joint space, without any external force sensors. The collision detection based on the generalized momentum is extended to robots with joint elasticity in [DLASHH06]. This work also presents several collision reaction schemes that take advantage of the directional information of the residual vector, and evaluates them on the DLRIII manipulator. A virtual force sensor is presented in [MFDL14]. The estimated external

[6] For safe human robot collaboration, the robot's motion has to fulfill additional constraints like low velocity and low exerted forces.

joint torques / residual vector $\boldsymbol{r}$ are mapped to a general contact point $\boldsymbol{P_c}$ on the robot's surface, generating a contact force $\boldsymbol{f_c}$ on that point through

$$\boldsymbol{f_c} = (\boldsymbol{J}_c^T(\boldsymbol{q}))^{\#}\boldsymbol{r},$$

where $\boldsymbol{J}_c^T$ is the Jacobian at $\boldsymbol{P_c}$. Since the contact Jacobian is in general not invertible, pseudo-inversion ($^{\#}$) is applied. The contact point $\boldsymbol{P_c}$ is estimated using a RGB-D camera. In an experiment, the human hands were tracked with such a camera, and the closest point on the robot's structure to these hands was considered as the contact point. Whole hand contacts were modeled by using an average of 10 contact points (fingers). In [MFDL15], this virtual force sensor is used to implement force and impedance control on arbitrary contact points. For restricted contact scenarios, the contact point can also be estimated by solving for the lever length when both a contact force and a contact moment are estimated [HDLAS17].

Given the good results of collision detection and estimation of contact forces $\boldsymbol{f_c}$ through the generalized momentum, our control architecture detects collisions with the same method. Additionally, the estimated contact forces are used to shape the impedance dynamics of the robot manipulator.

Chapter 2

Clamping Conscious Control

Working in the same workspace as a robot pose multiple threats to human co-workers. Collision related hazards can be circumvented by exploiting collision avoidance schemes, safe mechanical design and non-harmful trajectories. However, in collaborative tasks, contacts and transfer of forces between robots and humans cannot be avoided. In spacious settings, these contacts and forces do not pose high risks when the robot moves slowly, since the human co-worker can escape inconvenient contacts or high contact forces. The risk rises dramatically, if the human is constrained in his movements and is unable to escape dangerous situations. Hence, a safe robot control scheme has to be aware of potential clamping situations and has either to prevent these situations at all times or has to render such situations non-harmful. Section 2.1 explains the concepts of such a clamping conscious pipeline. The implementation details are described in Sec. 2.2.

2.1 Concepts

This section begins with describing the circumstances that apply in clamping situations in Sec.2.1.1. As distances between robot and objects play an important part therein, a short discussion about distance computation methods follows in Sec. 2.1.2. Afterwards, we explain our algorithm to identify clamping situations in Sec.2.1.3. As this algorithm computes box to box distances, we describe how the Separation Axis Theorem can be extended to approximate these distances in Sec.2.1.4. Since this thesis is concerned about clamping situations in pHRI, Sec. 2.1.5 outlines the robot model that we are using. Finally, the concepts of the proposed Clamping Conscious Impedance Control scheme are delineated.

2.1.1 Identifying Clamping Situations

There are mainly two different cases in which a robot can clamp a human co-worker: The robot can press a human body part against objects in the workspace, or it can squeeze the body part in between its own links. When the robot has

perfect knowledge about its geometry, the environment, and the pose and shape of the body parts in the workspace, clamping situations can be easily identified. If a body part is both in contact with a robot link and the environment, or with two different robot links, there is a clamping situation. However, in real life scenarios, the poses and shapes of body parts in the workspace are often only known to the robot up to a certain uncertainty. To protect the human, it is necessary to assume worst case poses and shapes, bounded by the degree of uncertainty. Identifying clamping situations is therefore highly dependent on the degree of uncertainty, and thus, the identification method used varies.

Let us assume that the robot has perfect knowledge about its static environment, but no means to detect the pose, nor the shape of human body parts in its workspace. To make valid and meaningful assumptions whether a body part can be clamped in the current robot configuration, one has to provide the robot with information about the body parts that can be in its workspace. Our approach is to define collaborative zones, together with body parts that can be in these zones. An example placement of collaborative zones can be seen in Fig. 2.1. If one has absolutely no knowledge about how co-workers interact with the robot, one can place a collaborative zone over the entire workcell, containing every possible body part as a worst case assumption. Additionally, we provide the robot with the extent of these body parts. Since every co-worker might have different body proportions, only minimal and maximal dimensions of these body parts are given to the robot (size_{min} and size_{max}). An illustrative example is given in Fig. 2.2. When there are more body parts present in one collaborative zone, the most conservative dimensions are used for identifying clamping situations. This means that for each collaborative zone, the smallest dimension size_{min} and the biggest dimension size_{max} is used. Section 2.2.1 gives more details about how the body part dimensions are obtained.
Clamping situations are detected separately for each robot link. For each link, the minimal and maximal distance to the environment is calculated (d_{min} and d_{max}). If it is geometrically possible that a body part (defined by the collaborative zone) can be trapped between the link and the environment, the controller is notified. This clamping case is indicated by:

$$d_{\text{min}} < \text{size}_{\text{max}}. \tag{2.1}$$

At the beginning of a clamping situation,

$$d_{\text{max}} > \text{size}_{\text{min}} \tag{2.2}$$

holds also. Naturally, a body part can only start to be trapped when the robot moves towards it. Figure 2.2b illustrates the validity of (2.1) and (2.2).

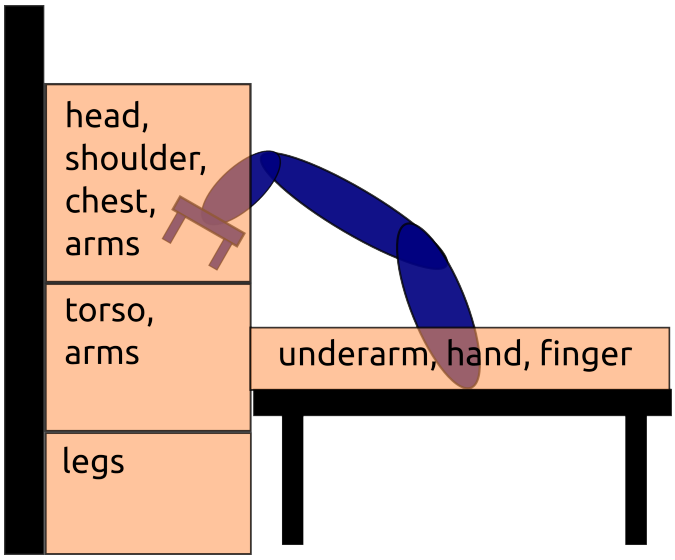


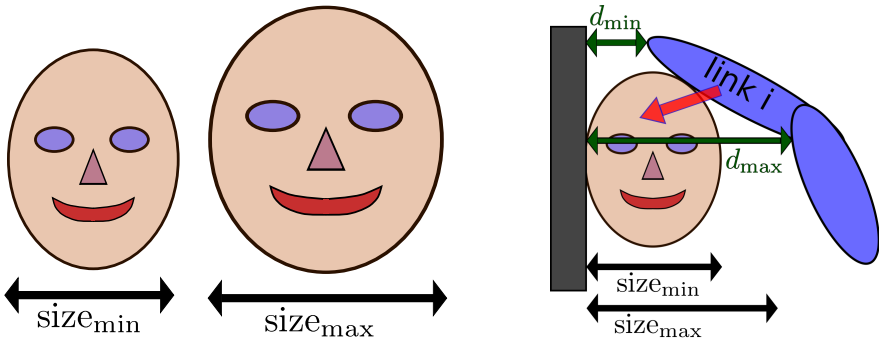
Figure 2.1: Placement of collaborative zones (light orange). Every collaborative zone contains the body parts that might be in the respective zone. Black objects (wall and table) constitute the static environment.

2.1.2 Distance Calculation

The clamping identification requires two distance measures per link: the minimum distance d_{min} and the maximum d_{max}. This bears an extra challenge to the distance calculation algorithm, since most common distance calculation methods are specialized to return only the minimum distance d_{min}. An additional challenge poses the control rate of modern robots, which is usually around 1kHz. A simple approximation of the scene and the robot with geometric primitives would not provide the necessary accuracy. An adaption of the *Voxmap PointShell*^{TM} algorithm would probably meet the accuracy requirements, but the voxel discretization of the robot's workspace would require much memory. Under normal circumstances, a robot workspace contains a lot of free space to allow for various robot motions. Therefore, a simple voxel discretization would be very wasteful. Additionally, the distance calculation algorithm should be extensible to also compute the distance (and contact manifold) between the robot and human body parts. Both of them are dynamic objects in the scene, but the *Voxmap PointShell*^{TM} algorithm can handle only computations between a static and a dynamic object.

2.1.3 Identifying Clamping Situations through OBBTree Traversal

We propose a hierarchical method to identify clamping situations time efficiently. The core idea is to start the identification process with a coarse approximation and refine the approximations where it is needed. Whenever a clamping situation at a



(a) Minimal and maximal dimensions

(b) Clamping requisites

Figure 2.2: **(a)**: For each of the considered body parts, a minimal and a maximal dimension value is assigned. The minimal value (size_{min}) expresses that no dimension of the body part is smaller than size_{min}. In the example of the head, it says that the height, width, and depth of the respective co-workers' heads are bigger than size_{min}. The biggest dimension of the body part is expressed through size_{max}. Continuing the head example, the co-workers' heads have smaller height, width, and depth than size_{max}. **(b)**: In a potential clamping situation, the minimal distance (d_{min}) between the robot and the environment is smaller than size_{max}. Otherwise, the human would not be constrained. Additionally, the maximal distance between the robot and the environment (d_{max}) must be greater than size_{min} for a *beginning* clamping situation. If this is not the case, the human cannot get constrained. The respective body part would not fit in the space between robot and environment. See also Fig. 2.7 for these cases. The velocity (red arrow) direction of the robot also plays an important part for predicting future clamping situations. Body parts only can get clamped, if the robot moves towards them.

given approximation scale cannot be ruled out, the approximations are refined until reaching a user-defined accuracy. To rule out clamping situations, we introduce several *Early Out Criteria (EOC)* that are dependent on the distance and the velocity between the respective objects.

OBBTree data structure

The chosen method is an adaption of the *Bounding Volume Hierarchy* (*BVH*) method (see Sec. 1.2.5). The distance calculation is accurate up to a user defined threshold, memory is only used for actual objects in the scene, and the BVH can be dynamically updated to allow distance computations between moving objects. As *Bounding Volumes* (*BV*) we use *Oriented Bounding Boxes* (*OBBs*), since they perform better for low-curvature surfaces, compared to sphere- or AABB-Trees [GLM96]. The reason is that they fit the geometry tighter, which leads to less traversals in the tree structure. Visual examination showed that more complicated bounding volumes are not necessary for the used robot.

Our OBBTree (tree structure of OBBs) implementation is an extension of the original one by [GLM96]. Instead of only approximating surface points with an OBB, we also save the normals of the points within it. This gives us optimized performance when identifying clamping situations. The normals of the points also influences the decision to which cluster they belong. Each cluster is later transformed into an OBB. An intermediate step in clustering and OBB creation can be seen in Fig. 2.3. The end result for the OBBs directly beneath the root of the OBBTree of the first robot link is displayed in Fig. 2.4. Further details on the OBBTree creation are described in Sec. 2.2.3.

OBBTree traversal

To identify a possible clamping situation between a robot link and any object in the environment, the OBBTree of the link is compared with every OBBTree of the environment. This comparison is done by traversing the OBBTrees synchronously. A schematic view of such an BVH is shown in Fig. 2.5. When traversing, the root nodes of the respective OBBTrees are compared to each other first. If this comparison cannot rule out a possible clamping situation, their children are compared to each other. Then, the same routine unrolls iteratively. This procedure is analogous to the collision detection pipeline of physics engines. There, a branch of a BVH (Bounding Volume Hierarchy) is dismissed when the top node of the branch cannot be in collision with the respective object. See Fig. 2.6 for an illustration. In this example, not intersecting BVs are used as an EOC. For the clamping identification algorithm, multiple EOCs are used. The first EOC rejects all environment OBBs from which the currently considered robot OBB moves away. These are the environment OBBs that does not lay in the current direction of movement of the

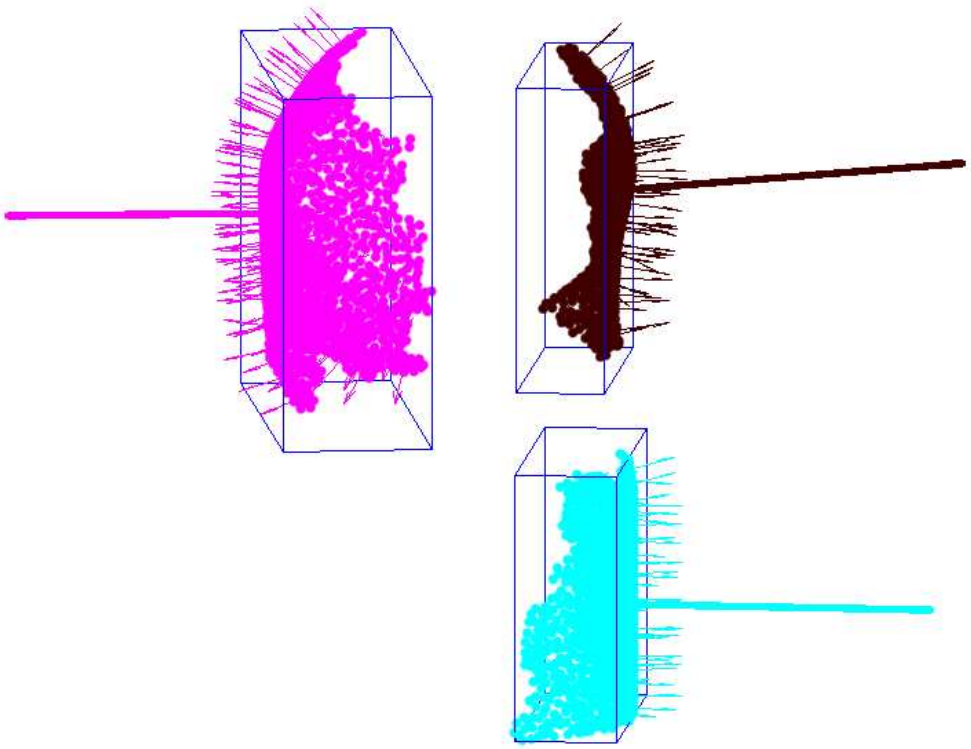


Figure 2.3: Snapshot of the OBB creation. The surface points are clustered depending on their locations and normal vectors. Each cluster is bounded by an OBB, for which also the centerpoint together with the average normal vector (displayed as large vector) is saved.

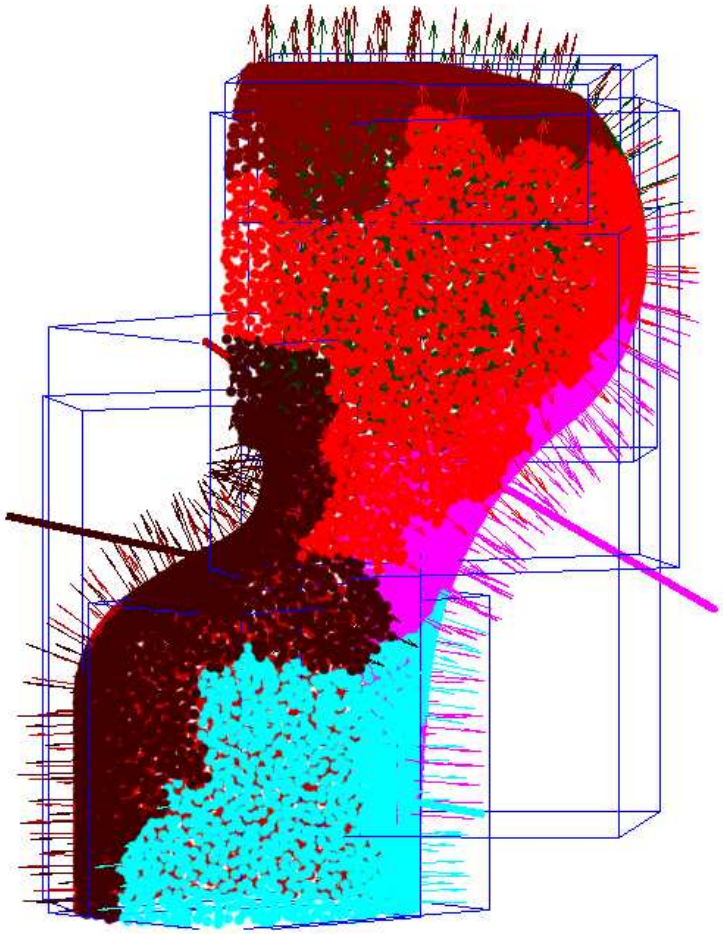


Figure 2.4: OBB creation result for the first link of the Panda robot. Each cluster is bounded by one OBB. The shown OBBs constitute the first depth of the OBBTree that has a total depth of 3.

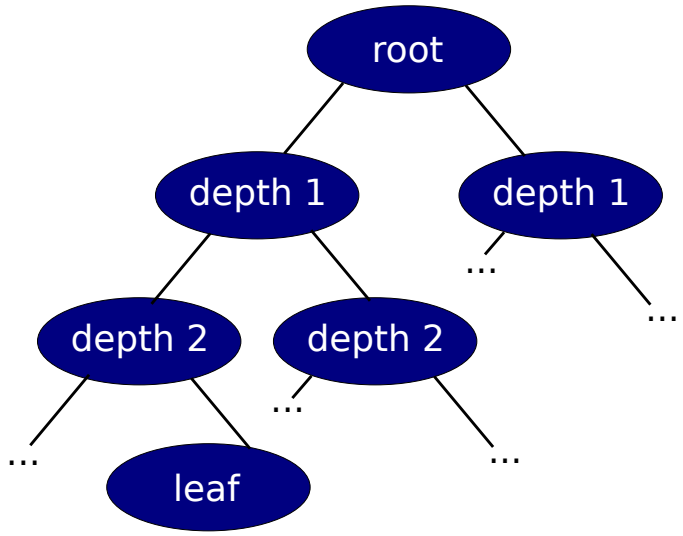
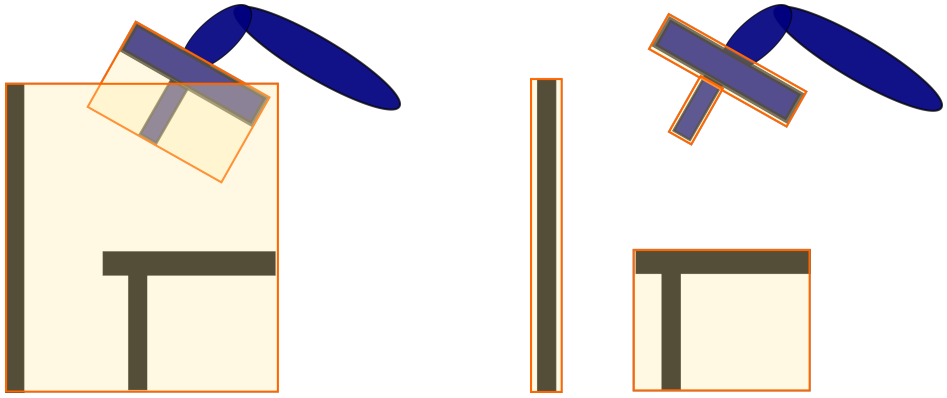


Figure 2.5: General layout of a BVH with 4 depth level (including the root) and a branching factor of 2.

robot OBB. The next EOC checks whether all normal vectors of the robot OBB are pointing approximately in this OBB's velocity direction. If they are not, this OBB represents a surface of the robot that is turned away from a possible contact that is initiated by the robot. Analogously, the normal vectors of the OBBs representing the environment have to point in the opposite velocity direction. The last EOC checks whether the geometric conditions for a clamping situation are fulfilled ((2.1) and (2.2)) and requires a minimum and maximum distance computation. These EOC are further detailed in Sec.2.2.5.

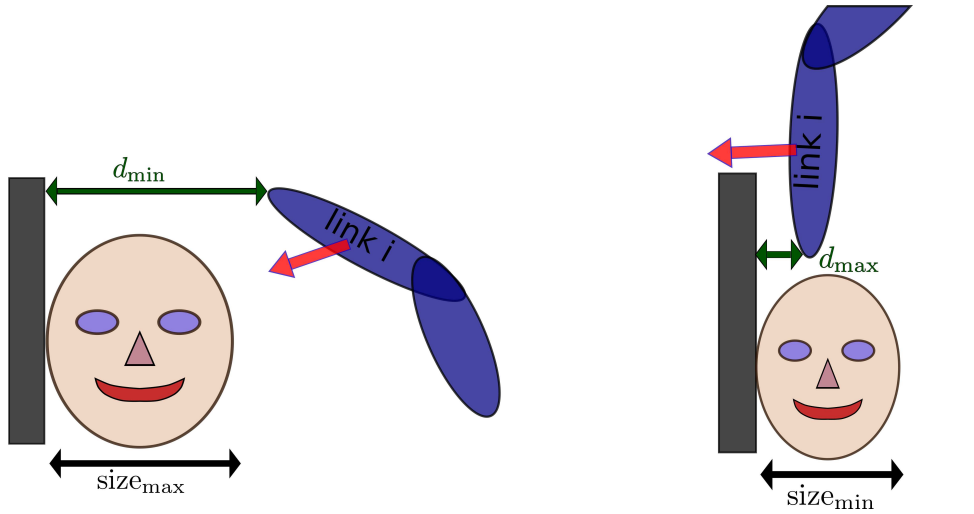
2.1.4 Separation Axis Theorem (SAT)

Given the fast execution time of the SAT in collision detection, we try to transfer its speed to approximative distance calculations. The general idea of the SAT is to check whether there exists a plane that separates two bodies. The normal vector of this plane is then a separation axis $\boldsymbol{l}$, as illustrated in Fig. 2.8. To check whether an axis is a separation axis, one has to project all points of the two bodies on this axis and check whether these point-sets are overlapping or not. This check can be simplified and tailored to specific Bounding Volumes. Moreover, the separation axes can be chosen smartly for different Bounding Volumes to limit the number of necessary separation checks. For OBBs, 15 axis have to be checked in total and Gottschalk et al. introduced an efficient way to check whether they are separation axes or not in [GLM96]. Referring to Fig. 2.9 and [GLM96], axis $\boldsymbol{l}$ is a separating



(a) BVs in collision. (b) BVs not in collision.

Figure 2.6: A collision between the robot's end-effector and the environment is queried. The root nodes of both bounding volume hierarchies are in collision in *(a)*. Therefore a collision cannot be ruled out on this approximation level. Further refining the approximation in *(b)* rules out a collision.



(a) EOC because of d_{min} (b) EOC because of d_{max}

Figure 2.7: Body part related early out criteria. In *(a)* the link is too far away from the environment, such that even the largest dimension of an head will fit in between with enough clearance: $d_{\text{min}} > \text{size}_{\text{max}}$. In *(b)* the link is already so close to the environment, such that even the smallest dimension of an head will not fit in between: $d_{\text{max}} < \text{size}_{\text{min}}$. This EOC is only valid if there was no actual clamping before.

axis if and only if

$$\begin{aligned} |\boldsymbol{t} \cdot \boldsymbol{l}| &> \sum_j |\alpha_j \boldsymbol{a_j} \cdot \boldsymbol{l}| + \sum_j |\beta_j \boldsymbol{b_j} \cdot \boldsymbol{l}| \\ &> r^A + r^B, \end{aligned} \tag{2.3}$$

where $\boldsymbol{t}$ is the translation between the OBBs' center points, $\boldsymbol{a_j}$ ($\boldsymbol{b_j}$) are unit vectors of OBB A's (OBB B's) face normals expressed in the world coordinate system and α_j (β_j) are the lengths in these directions from the box's center to the respective face. The dimensions of the OBBs are projected onto the axis $\boldsymbol{l}$ as r^A and r^B, with

$$\begin{aligned} r^A &= \sum_j |\alpha_j \boldsymbol{a_j} \cdot \boldsymbol{l}| \\ r^B &= \sum_j |\beta_j \boldsymbol{b_j} \cdot \boldsymbol{l}| \end{aligned} \tag{2.4}$$

The distance d_i between the two OBBs along axis $\boldsymbol{l}_i$ is

$$d_i = |\boldsymbol{t} \cdot \boldsymbol{l_i}| - r_i^A - r_i^B, \tag{2.5}$$

when $\boldsymbol{l}_i$ is a unit vector. It is possible that there exists multiple separation axes for two OBBs. If the separation axes are perpendicular to each other, the OBBs are separated along perpendicular directions. In this case, the distances d_i along direction i can be added together to approximate the minimum distance $d_{\min}$. As depicted in Fig. 2.10a, this approximative distance $\tilde{d}_{\min}$ is composed as

$$\tilde{d}_{\min} = \sqrt{\sum_i \sigma_i d_i^2}, \tag{2.6}$$

where

$$\sigma_i = \begin{cases} 1, & \text{for } d_i > 0 \\ 0, & \text{otherwise} \end{cases}$$

allows only positive distances d_i to be part of the distance approximation. Since the approximative distance $\tilde{d}_{\min}$ is always equal or smaller than the real minimum distance $d_{\min}$, its usage in the EOC determination is not leading to false positives. Further details follow in Sec. 2.2.5.

To approximate the maximum distance $d_{\max}$, several values from the minimum distance approximation are reused.

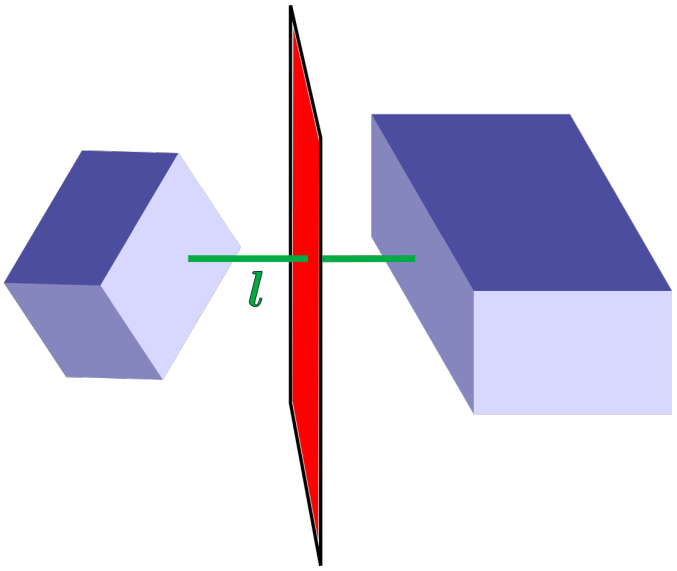


Figure 2.8: Two Bounding Volumes, in this case OBBs, are separated from each other if there exist a plane that separates them. The normal $\boldsymbol{l}$ of this plane is called a separation vector.

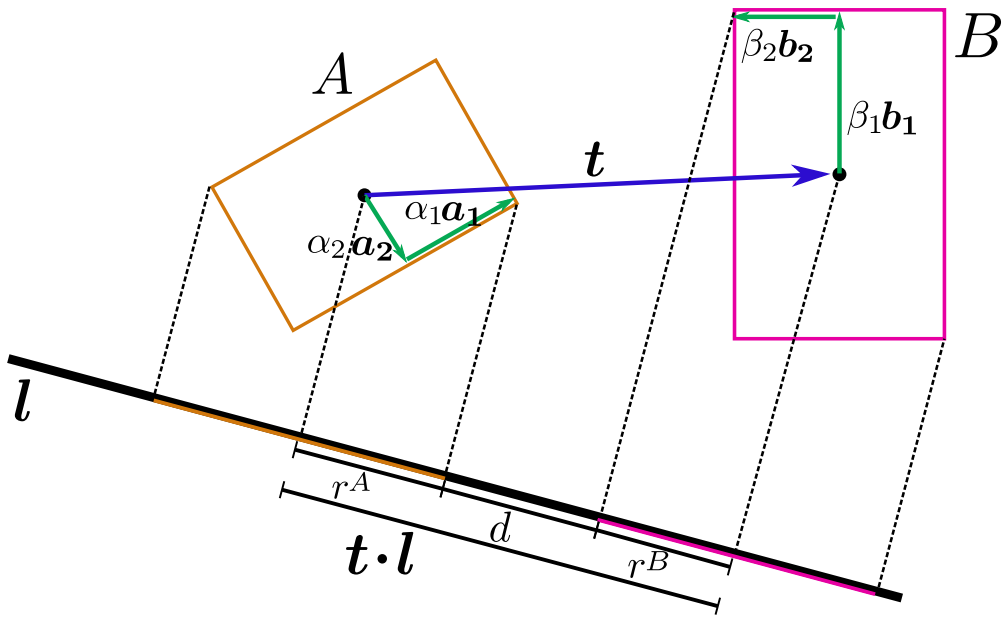
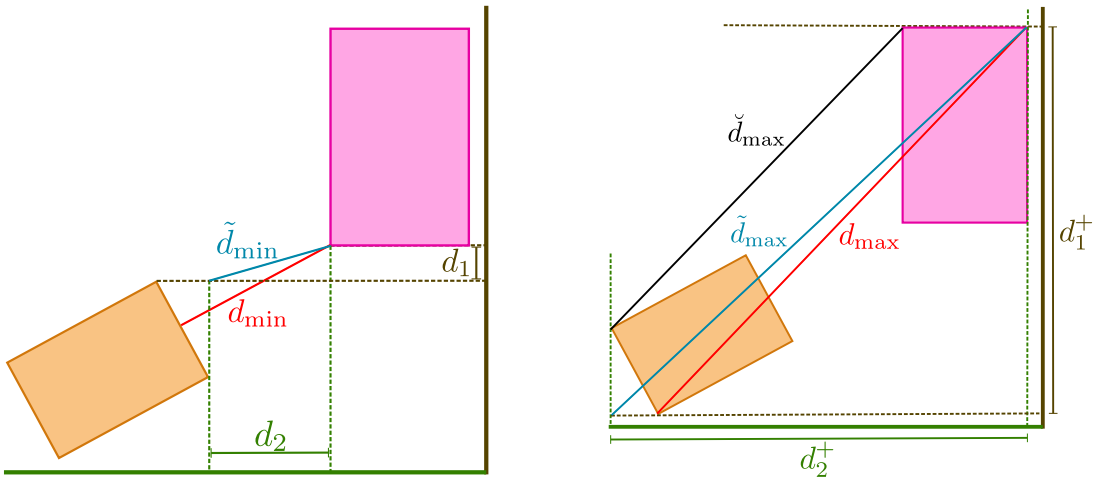


Figure 2.9: Separation axis test in 2-D. The OBBs A and B are projected on the separation axis $\boldsymbol{l}$. Here, the two projected point-sets are separated by distance d.



(a) Minimum distance approximation.

(b) Maximum distance approximation.

Figure 2.10: Approximations of the minimum and maximum distance between two OBBs. The approximative minimum distance $\tilde{d}_{\min}$ is calculated with the projected distances d_1 and d_2 according to (2.6). Note that $\tilde{d}_{\min}$ is always equal or smaller than the actual minimum distance $d_{\min}$. The real maximum distance $d_{\max}$ is taken to be the biggest distance between any point of the one shape (body) and the other. Its approximation $\tilde{d}_{\max}$ visually represents the diagonal of the rectangle (in 2-D) or the cuboid (in 3-D) that is spanned by the projected maximum distances d_i^+. Concerning clamping situations, both $d_{\max}$ and $\tilde{d}_{\max}$ are bigger than the distance $\breve{d}_{\max}$ that defines the maximum distance between points that face each other.

The projected maximum distance d_i^+ along separation axis $\boldsymbol{l}_i$ is obtained through

$$d_i^+ = |\boldsymbol{t} \cdot \boldsymbol{l_i}| + r_i^A + r_i^B, \tag{2.7}$$

for $\boldsymbol{l}_i$ being of unit length. For a system of perpendicular separation axes, we approximate $d_{\max}$ with

$$\tilde{d}_{\max} = \sqrt{\sum_i (d_i^+)^2}. \tag{2.8}$$

Figure 2.10b visualizes the approximative maximum distance $\tilde{d}_{\max}$ as well as the maximum distance $d_{\max}$ between each point pair. Both distance metrics form an upper bound on the maximum distance $\breve{d}_{\max}$ that is crucial for clamping situations, defining the maximum distance between planes that face each other. The rectangle (or cuboid in 3-D) spanned by d_i^+ does not always contain all points from both OBBs as can be seen in Fig. 2.11. However, its diagonal is still longer than any distance between the two OBBs. Further details on the efficient computation of the approximations $\tilde{d}_{\min}$ and $\tilde{d}_{\max}$ are stated in Sec. 2.2.6.

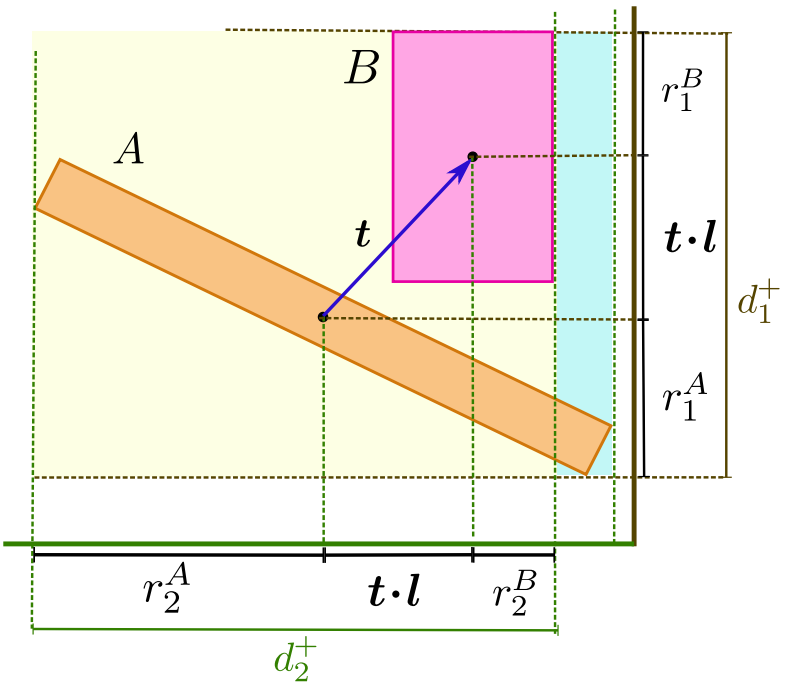


Figure 2.11: The yellow rectangle (cuboid) that is spanned by the projected maximum distances d_i^+ can be smaller than the rectangle (cuboid) containing all points of the two OBBs (yellow and turquoise rectangles combined).

2.1.5 Robot Model

A common approach to model serial robot manipulators is to view them as a composition of rigid links and joints. For purely rotational manipulators, the joints rotate their neighboring links relative to each other by the joint angles $\boldsymbol{q}$. The dynamic model of such a *rigid joint robot* with n joints can be expressed analogously to [SS12] as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{ext}}, \tag{2.9}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ are the joint positions, velocities and accelerations. $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the symmetric, positive definite inertia matrix. The Coriolis and centrifugal forces are composed of the Matrix $\boldsymbol{C}(\dot{\boldsymbol{q}}, \boldsymbol{q}) \in \mathbb{R}^{n\times n}$, containing the Christoffel symbols, and $\dot{\boldsymbol{q}}$. Gravity forces are modeled through $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$. The torques on the right-hand side are $\boldsymbol{\tau} \in \mathbb{R}^n$, being the commanded torques, and $\boldsymbol{\tau}_{\text{ext}} \in \mathbb{R}^n$ being the external torques acting on the robot's joints. Equation (2.9) is also known as a robot's *state-space equation* with the two state vectors $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$.

Modern *Light Weight Robot (LWR)* arms, as the KUKA-DLR LWR arm [HSAS+02] or the Panda [EMI17] contain elastic elements between the joint actuators and the links. These elements can make the robot's motion more compliant (see Sec. 1.2.1)

and can be used to estimate external forces acting on the robot [DLASHH06]. For robots with elastic joints, the motor positions and velocities $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ generally differ from the joint positions and velocities $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. The relationship between the motor-side state variables and the link-side ones are described by [ASOFH03]:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{K}(\boldsymbol{\theta} - \boldsymbol{q}) + \boldsymbol{\tau}_{\text{ext}}, \tag{2.10}$$

$$\boldsymbol{B}\ddot{\boldsymbol{\theta}} + \boldsymbol{K}(\boldsymbol{\theta} - \boldsymbol{q}) = \boldsymbol{\tau_m}. \tag{2.11}$$

Here, the left-hand side of the first equation is the same as for the rigid joint case. The second equation models the motor dynamics with $\boldsymbol{B}$ and $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ being both diagonal matrices describing the inertias of the individual motors and the joint stiffnesses. The control input is $\boldsymbol{\tau_m} \in \mathbb{R}^n$, specifying the motor torque. The two equations are coupled through $\boldsymbol{K}(\boldsymbol{\theta} - \boldsymbol{q})$, representing the torques acting on the joints due to motor commands. Given the complexity of this flexible joint model, it is often difficult to design controllers using the flexible joint assumption. One solution is to use the *singular perturbation approach* to transfer the flexible joint dynamics into a structure equal to the rigid joint case (2.9). As described in [ASOFH03], this can be done by adding a fast subsystem that commands the motor torques $\boldsymbol{\tau_m}$ as

$$\boldsymbol{\tau_m} = \boldsymbol{\tau_d} - \boldsymbol{K_T}(\boldsymbol{\tau} - \boldsymbol{\tau_d}) - \boldsymbol{K_S}\dot{\boldsymbol{\tau}}, \tag{2.12}$$

when given a desired joint torque vector $\boldsymbol{\tau_d}$. With $\boldsymbol{K_T}$ and $\boldsymbol{K_S}$ being positive definite controller matrices, the actual joint torque is controlled to be $\boldsymbol{\tau} = \boldsymbol{\tau_d}$. This leads in total to the following link dynamics:

$$\bar{\boldsymbol{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau_d} + \boldsymbol{\tau}_{\text{ext}}, \tag{2.13}$$

with

$$\bar{\boldsymbol{M}}(\boldsymbol{q}) = (\boldsymbol{M}(\boldsymbol{q}) + (\boldsymbol{I} + \boldsymbol{K_T})^{-1}\boldsymbol{B}). \tag{2.14}$$

Comparing (2.13) with (2.9), one sees that these equations are identical with the exception of a different inertia matrix $\boldsymbol{M}$. It is therefore possible to control elastic joint robots as if their joints are rigid by replacing their inertia matrix $\boldsymbol{M}$ with $\bar{\boldsymbol{M}}$.

2.1.6 Clamping Conscious Impedance Control

Impedance control is commonly used in human-robot interaction tasks, since it can be configured to provide high compliance. Moreover, compared to hybrid force/position control, the transition between contact and non-contact states does in general not lead to instabilities. This advantage makes impedance control preferable in unstructured environments. One major distinction in impedance control is whether the impedance dynamics (i.e. the relation between forces and displacements/ velocities/ accelerations) are established in the *joint space* or in another

operational space. In *operational space control*[Kha87], the robot's motion is expressed in task related coordinate systems. In many use-cases the respective task is expressed in Cartesian coordinates, leading to a *Cartesian impedance controller*. The ideal closed loop dynamics of a standard Cartesian impedance controlled robot can be described as

$$\boldsymbol{\Lambda_d}\ddot{\boldsymbol{e}} + \boldsymbol{D}\dot{\boldsymbol{e}} + \boldsymbol{K}\boldsymbol{e} = \boldsymbol{f}_{\text{ext}}. \tag{2.15}$$

Here, $\boldsymbol{e}$, $\dot{\boldsymbol{e}}$, and $\ddot{\boldsymbol{e}} \in \mathbb{R}^s$ are the pose, velocity and acceleration errors of the end-effector in the Cartesian space with dimension s. The pose error $\boldsymbol{e}$ is defined as $\boldsymbol{x} - \boldsymbol{x_d}$, with $\boldsymbol{x}$ being the end-effector pose. The subscript $\cdot_d$ denotes desired values, which are often assumed to be constant. The end-effector terms $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$, and $\ddot{\boldsymbol{x}}$, as well as the corresponding error terms $\boldsymbol{e}$, $\dot{\boldsymbol{e}}$ and $\ddot{\boldsymbol{e}}$ are in general time dependent. However, for the sake of readability, these time dependencies are dropped unless they require emphases.
Depending on whether the orientation of the end-effector should be controlled, the Cartesian dimension s is often chosen to be 3 (to control the $x-$, $y-$ and $z-$ directions), or 6 (to control additionally roll, pitch and yaw). External forces acting on the end-effector are described by $\boldsymbol{f}_{\text{ext}}$. $\boldsymbol{K}$, $\boldsymbol{D}$, and $\boldsymbol{\Lambda_d} \in \mathbb{R}^{s\times s}$ are the stiffness, damping, and desired inertia matrices of the impedance dynamic. This dynamic is stable in a passive environment if $\boldsymbol{K}$, $\boldsymbol{D}$, and $\boldsymbol{\Lambda_d}$ are positive definite and constant [KB16]. Usually, the matrices are chosen to be diagonal such that the dynamics in each coordinate direction is decoupled from each other.

In the following, some ideas and equations refer only to certain parts of matrices and vectors, or need them to be expressed in different coordinate systems. Therefore, we introduce here the respective notation for them.

Notation 1. Given the partitioning of the Cartesian pose vector $\boldsymbol{x} = [\boldsymbol{p}, \boldsymbol{\phi}]$ into position and orientation related coordinates, the subscripts $\cdot_p$ and $\cdot_\phi$ will be used to specify position and orientation elements of other matrices and vectors, too. For a matrix $\boldsymbol{A} \in \mathbb{R}^{6\times6}$, $\boldsymbol{A_p} \in \mathbb{R}^{3\times3}$ specifies the upper left block, $\boldsymbol{A_\phi} \in \mathbb{R}^{3\times3}$ the bottom right one. A vector $\boldsymbol{a} \in \mathbb{R}^6$ is also separated into $\boldsymbol{a_p} \in \mathbb{R}^3$ and $\boldsymbol{a_\phi} \in \mathbb{R}^3$ being the first three and the last three vector entries. As an example, $\boldsymbol{e_p} \in \mathbb{R}^3$ is the position error and $\boldsymbol{K_\phi} \in \mathbb{R}^{3\times3}$ contains the rotational stiffnesses. Individual entries of matrices and vectors are referred to by row and if necessary column indices starting from 1. Note that $K_{[4,4]}$ and $K_{\phi,[1,1]}$ denote the same element. For vectors, the indices can be stated without brackets if the indices cannot be confused with other subscripts. Consequently, a_1 is the first entry of vector $\boldsymbol{a}$. To specify a subvector, indices can be stated as a range, e.g. $\boldsymbol{a_{[2-4]}}$. For submatrices, the range applies to rows and columns likewise. We can for example write $\boldsymbol{A_p}$ as $\boldsymbol{A_{[1-3]}}$. When unrolling matrix vector multiplications, we make use of the notation $\vec{\boldsymbol{a_i}}$ to refer to the vector

that is constructed by setting all elements of vector $\boldsymbol{a}$ to 0 except element a_i. If it is important in which coordinate frame a vector is expressed, the coordinate frame is specified as a leading superscript. For instance ${}^{b}\boldsymbol{a}$ is the vector $\boldsymbol{a}$ expressed in coordinate frame ${}^{b}\chi$.

Notation 2. Special matrices:

- $\boldsymbol{I_n} \in \mathbb{R}^{n\times n}$ is the identity matrix.
- $\boldsymbol{0_n} \in \mathbb{R}^{n\times n}$ is a matrix full of zeros.

The robot dynamics in (2.15) allow the tracking of time varying end-effector pose $\boldsymbol{x_d}$. The control schemes developed in this thesis track piecewise constant poses, meaning that the desired poses remain constant until they are reached. Hence, the derivatives of the pose error are simply

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}} \text{ and } \ddot{\boldsymbol{e}} = \ddot{\boldsymbol{x}}.$$

Consequently, (2.15) reduces in this case to

$$\boldsymbol{\Lambda_d}\ddot{\boldsymbol{x}} + \boldsymbol{D}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{e} = \boldsymbol{f}_{\text{ext}}. \tag{2.16}$$

To elaborate several concepts of the Clamping Conscious Impedance Control (CCIC), we first look at a one-dimensional model of the controlled system and the environment before applying these concepts to the more general multi-dimensional case. In one dimension, (2.16) gets simplified to

$$\begin{aligned} m\ddot{x}_m + d\dot{x}_m + ke &= f_{\text{ext}} \\ \Leftrightarrow m\ddot{x}_m + d\dot{x}_m + k(x_m - x_d) &= f_{\text{ext}} \end{aligned} \tag{2.17}$$

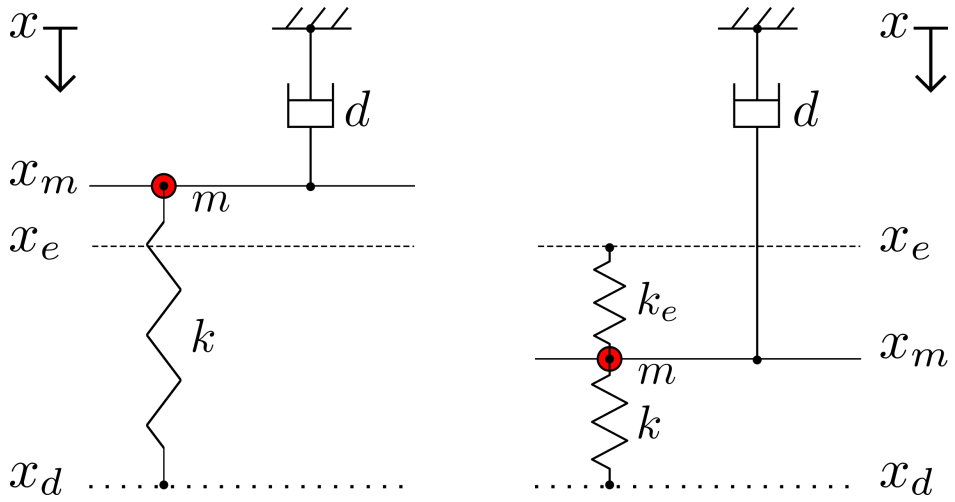
with m being the controlled point mass at position x_m, k being the scalar impedance stiffness, and d being the scalar damping coefficient. A schematic representation of this model is shown in Fig. 2.12. If the controlled mass m is in free motion (i.e. it is not in contact with the environment), then $f_{\text{ext}} = 0$. We model the environment as a simple spring with equilibrium position x_e and spring stiffness k_e. Hence, when the point mass m is in contact with the environment, (2.17) becomes

$$m\ddot{x}_m + d\dot{x}_m + k(x_m - x_d) = k_e(x_e - x_m). \tag{2.18}$$

Steady state analysis

The main goal of the CCIC is to render potential clamping situations safe. One critical aspect is to limit the exerted force on the clamped body part to safe values $f_{\max}$. Regarding the one-dimensional model (2.18) in steady state,

$$k(x_m - x_d) = k_e(x_e - x_m),$$



(a) Free motion

(b) In contact with environment

Figure 2.12: One-dimensional impedance model of Eq. (2.17). The controlled mass is shown as red circle and is at x_m. Its velocity $\dot{x}$ is damped by d. The spring k pulls the mass to reach x_d. In *(a)*, the mass is in free motion and the external force f_{ext} is 0. In *(b)*, the mass is in contact with the environment and has crossed the equilibrium position x_e of the environment. The resulting external force $f_{\text{ext}} = k_e(x_e - x_m)$ is counteracting the pulling force of spring k.

we see that the external force is equal to the spring force $k(x_m - x_d)$ that drags the mass m to its goal position x_d. Consequently, a minimum requirement for the CCIC is that the external steady state force is below $f_{\max}$:

$$\|k(x_m - x_d)\| = \|k_e(x_e - x_m)\| < f_{\max}. \tag{2.19}$$

We meet this requirement by incorporating a **velocity saturation** policy adapted from [Kha86, Sen07]. First, we rewrite (2.17) to be a velocity tracking control:

$$\begin{aligned} m\ddot{x}_m + d(\dot{x}_m - \nu\dot{x}_d) &= f_{\text{ext}} \\ \dot{x}_d &= \frac{k}{d}(x_d - x_m) \\ \nu &= min\left(1, \frac{v_{\max}}{\|\dot{x}_d\|}\right) \\ v_{\max} &= \frac{f_{\max}}{d} \end{aligned} \tag{2.20}$$

Here, $\dot{x}_d$ is the target velocity to track, but this velocity is saturated by $v_{\max} > 0$. This saturation becomes clearer when we look at two distinct cases separately.

- **Case 1:** $\|\dot{x}_d\| < v_{\max}$
  When the norm of the desired velocity $\|\dot{x}_d\|$ is smaller than the velocity limit $v_{\max}$, the impedance dynamic (2.20) equals the original dynamic (2.17):

$$\begin{aligned} \|\dot{x}_d\| < v_{\max} \Rightarrow \frac{v_{\max}}{\|\dot{x}_d\|} > 1 \Rightarrow \nu = 1 \\ m\ddot{x}_m + d(\dot{x}_m - \nu\dot{x}_d) = f_{\text{ext}} \\ \Leftrightarrow m\ddot{x}_m + d(\dot{x}_m - \dot{x}_d) = f_{\text{ext}} \\ \Leftrightarrow m\ddot{x}_m + d\dot{x}_m + k(x_m - x_d) = f_{\text{ext}}. \end{aligned} \tag{2.21}$$

  Additionally,

$$\|\dot{x}_d\| < v_{\max} \Leftrightarrow \left\|\frac{k(x_d - x_m)}{d}\right\| < \frac{f_{\max}}{d} \Rightarrow \|k(x_d - x_m)\| < f_{\max},$$

  which means that the steady state force is below $f_{\max}$.

- **Case 2:** $\|\dot{x}_d\| > v_{\max}$
  When the norm of the desired velocity $\|\dot{x}_d\|$ is bigger than the velocity limit $v_{\max}$, the steady state force is also bounded by $f_{\max}$:

$$\begin{aligned}
&\|\dot{x}_d\| > v_{\max} \Rightarrow \frac{v_{\max}}{\|\dot{x}_d\|} < 1 \Rightarrow \nu = \frac{v_{\max}}{\|\dot{x}_d\|} \\
&m\ddot{x}_m + d(\dot{x}_m - \nu\dot{x}_d) = f_{\text{ext}} \\
\Leftrightarrow\ &m\ddot{x}_m + d(\dot{x}_m - v_{\max}\frac{\dot{x}_d}{\|\dot{x}_d\|}) = f_{\text{ext}} \\
\Leftrightarrow\ &m\ddot{x}_m + d\dot{x}_m - f_{\max}\frac{\dot{x}_d}{\|\dot{x}_d\|} = f_{\text{ext}} \\
\Leftrightarrow\ &m\ddot{x}_m + d\dot{x}_m - f_{\max}\frac{x_d - x_m}{\|x_d - x_m\|} = f_{\text{ext}} \\
\Leftrightarrow\ &m\ddot{x}_m + d\dot{x}_m + f_{\max}\frac{x_m - x_d}{\|x_m - x_d\|} = f_{\text{ext}}.
\end{aligned} \tag{2.22}$$

With above velocity saturation, the steady state force can be limited. However, since (2.21) and (2.22) are second order differential equations, they can exhibit non-negligible transient behaviors. These transients can lead to force overshoots - which might be non-tolerable in robot-human collisions - and need to be further investigated.

Transient analysis

Comparing the dynamics of (2.21) and (2.22), the latter poses the more dangerous case because $f_{\max} \geq \|k(x_m - x_d)\|$. Hence, our transient analysis will focus on (2.22). We can reformulate (2.22) such that the right-hand side is only composed of (piece-wise) constant terms, simplifying the analysis:

$$\begin{aligned}
&m\ddot{x}_m + d\dot{x}_m + f_{\max}\frac{x_m - x_d}{\|x_m - x_d\|} = f_{\text{ext}} \\
\Leftrightarrow\ &m\ddot{x}_m + d\dot{x}_m + f_{\max}\frac{x_m - x_d}{\|x_m - x_d\|} = k_e(x_e - x_m) \\
\Leftrightarrow\ &m\ddot{x}_m + d\dot{x}_m + k_e x_m = k_e x_e - f_{\max}\operatorname{sgn}(x_m - x_d),
\end{aligned} \tag{2.23}$$

where $\operatorname{sgn}(\cdot)$ specifies the signum function

$$\operatorname{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

If we further assume that the controlled mass does not reach its goal position (otherwise, the dynamics would change to (2.21)), we can reduce (2.23) to

$$\begin{aligned}
&m\ddot{x}_m + d\dot{x}_m + k_e x_m = k_e x_e + f_{\max} \\
\Leftrightarrow\ &m\ddot{x}_m + d\dot{x}_m + k_e x_m = f_0.
\end{aligned} \tag{2.24}$$

Here, we set $x_m < x_d$ w.l.o.g. and combined the constant right-hand side forces to f_0. Before solving this non-homogeneous second order linear differential equation, it is brought to the more common form:

$$\ddot{x}_m + 2\zeta\omega_n\dot{x}_m + \omega_n^2 x_m = \frac{f_0}{m}, \tag{2.25}$$

with $\omega_n = \sqrt{\frac{k_e}{m}}$ being the natural frequency of the system, and $\zeta = \frac{d}{2\sqrt{k_e m}}$ being the damping coefficient. Depending on the damping coefficient ζ, the solution of (2.25) takes a different shape. When the system is underdamped (i.e. $\zeta < 1$), it oscillates. If it is critically ($\zeta = 1$) or overdamped (i.e. $\zeta > 1$), it approaches exponentially the steady state. Since a critically damped system approaches the steady state the fastest, it is preferred for our use case. However, it is non trivial to damp the dynamics critically because one has to know the stiffness of the environment k_e for this. In practice, the system is either underdamped or overdamped, depending on k_e. Below we state the solutions of the differential equation (2.25) for the different damping cases, the solving steps are elaborated in the appendix B.1. With $\omega_d = \omega_n\sqrt{1-\zeta^2}$ being the damped natural frequency, and v_0 being the initial velocity we obtain

- **Underdamped:**

$$\begin{aligned} x_m(t) &= e^{-\zeta\omega_n t} A\sin(\omega_d t + \phi) + \frac{f_0}{k_e} \\ A &= \sqrt{C_1 + C_2} \\ C_1 &= \frac{f_{\max}}{k_e} \\ C_2 &= \frac{v_0 + \zeta\omega_n \frac{f_{\max}}{k_e}}{\omega_d} \\ \phi &= \operatorname{atan2}(\frac{C_1}{C_2}) \end{aligned} \tag{2.26}$$

- **Critically damped:**

$$\begin{aligned} x_m(t) &= e^{-\zeta\omega_n t}\left(C_1 + tC_2\right) + \frac{f_0}{k_e} \\ C_1 &= \frac{f_{\max}}{k_e} \\ C_2 &= v_0 + \omega_n C_1 \end{aligned} \tag{2.27}$$

- **Overdamped:**

$$\begin{aligned} x_m(t) &= C_1 e^{z_1 t} + C_2 e^{z_2 t} + \frac{f_0}{k_e} \\ z_1 &= -\zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1} \\ z_2 &= -\zeta\omega_n - \omega_n\sqrt{\zeta^2 - 1} \\ C_1 &= \frac{f_{\max}}{k_e} - C_2 \\ C_2 &= \frac{-v_0 + z_1\frac{f_{\max}}{k_e}}{z_1 - z_2} \end{aligned} \tag{2.28}$$

The above dynamics assume a contact with the environment at $t = 0$, i.e. $x_m(0) = x_e$, and that the contact is maintained thereafter. Figure 2.13 and Fig. 2.14 illustrate the transient trajectories of the control point for the different damping cases. All trajectories reach the respective steady state equilibrium point

$$x_{ss} = x_e - \frac{f_{\max}}{k_e},$$

however there are differences in the settling time and in the position overshoot

$$\Delta x_m = \max_t(x_{ss} - x_m).$$

Here, we suppose that the control point hits the environment from above (positive x-direction). If the control point collide with the environment from the opposite direction,

$$\Delta x_m = |\min_t(x_{ss} - x_m)|$$

should be chosen as position overshoot. The position overshoot Δx_m leads direct to the transient force overshoot

$$\Delta f = k_e * \Delta x_m.$$

As can be seen in Fig. 2.13, the critical damped dynamics reaches the steady state x_{ss} the fastest without any position overshoot. However, if one increases the environment stiffness k_e, but leaves the initial velocity v_0 the same, even the critical damped dynamics shows a position overshoot Δx_m as illustrated in Fig. 2.14. This

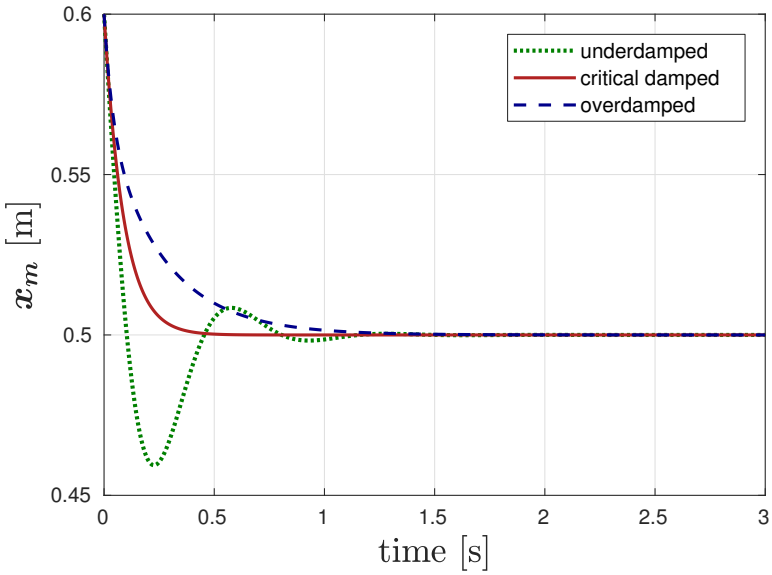


Figure 2.13: Control point trajectory with $k_e = 10^3$. In all three damping cases, the parameters except the damping coefficients d_i are kept identical and can be seen in Tab. 2.1.

Table 2.1: Parameters for Fig. 2.13 and Fig. 2.14. The damping coefficient for the underdamped case is calculated as $d_u = 2\sqrt{mk}$, for the critically damped case as $d_c = 2\sqrt{mk_e}$, and for the overdamped case as $d_o = 3\sqrt{mk_e}$. The initial velocity v_0 is taken to be $v_0 = \frac{f_{\max}}{d_u}$.

Figure	m [kg]	k $\left[\frac{\text{kg}}{\text{s}^2}\right]$	$f_{\max}$ [N]	v_0 $\left[\frac{\text{m}}{\text{s}}\right]$	x_e [m]	d_u $\left[\frac{\text{kg}}{\text{s}}\right]$	d_c $\left[\frac{\text{kg}}{\text{s}}\right]$	d_o $\left[\frac{\text{kg}}{\text{s}}\right]$	k_e $\left[\frac{\text{kg}}{\text{s}^2}\right]$
Fig. 2.13	10	200	100	-1.12	0.6	89.44	200	300	10^3
Fig. 2.14	10	200	100	-1.12	0.6	89.44	632.46	948.68	10^4

is because the steady state x_{ss} is close to the environment resting position x_e and the damping element cannot decelerate the mass fast enough. In this case, the overdamped system cause the least overshoot. The high initial velocity v_0 even causes the underdamped system to oscillate as much as that it looses contact with the environment again. The force overshoots for the trajectories in Fig. 2.13 and Fig. 2.14 are reported in Tab. 2.2.

Multi-dimensional scenario

In the one-dimensional contact scenario we are able to limit the steady state external force, as well as preventing transient force overshoots by choosing appropriate damping and initial velocity. Figure 2.15 illustrates a two-dimensional contact

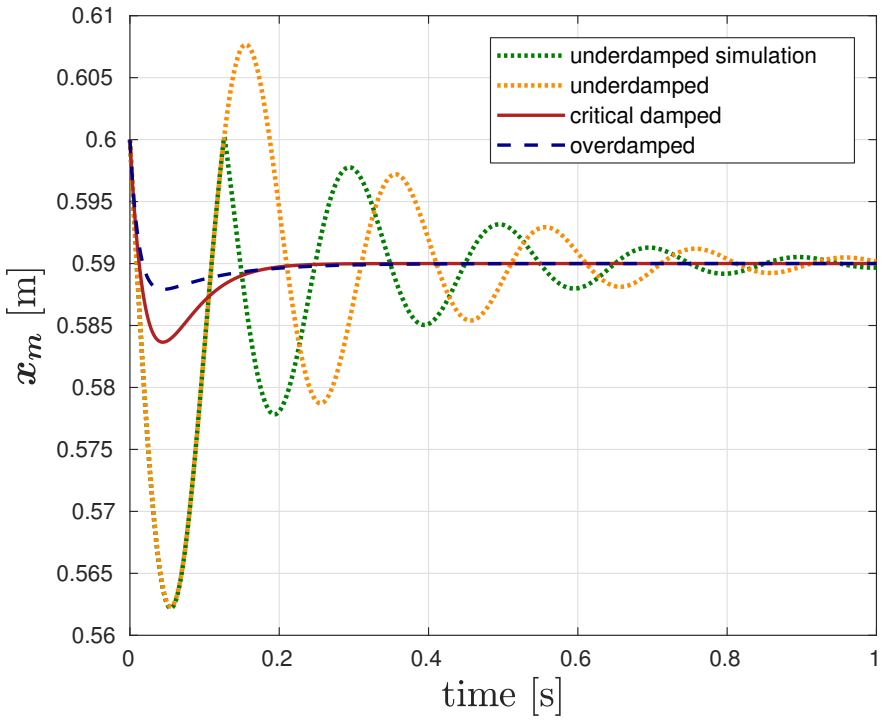


Figure 2.14: Control point trajectory with $k_e = 10^4$. The impedance parameters are listed in Tab. 2.1. In the underdamped case, there are two graphs shown because the control point oscillates such that it looses touch with the environment again. The line in orange shows the dynamics in (2.28), not considering the lost in contact, whereas the green line is the result of a simulation that does consider the lost in contact.

Table 2.2: Maximal transient force overshoots. Since the maximal force is $f_{\max} = 100\text{N}$, the values can be either interpreted as force values in Newton, or as the percentages of $f_{\max}$.

Figure	Force overshoots in [N] and [%]		
	underdamped	critical damped	overdamped
Fig. 2.13	40.6	0	0
Fig. 2.14	278.1	63.5	20.9

scenario. The environment is still modeled as a spring that exert forces only in the direction normal to its surface. In the general two dimensional case (Fig. 2.15a) the resulting spring force

$$\boldsymbol{f_k} = \boldsymbol{K}\boldsymbol{e} = K_{[1,1]}\vec{\boldsymbol{e_1}} + K_{[2,2]}\vec{\boldsymbol{e_2}} = \boldsymbol{f_1} + \boldsymbol{f_2}$$

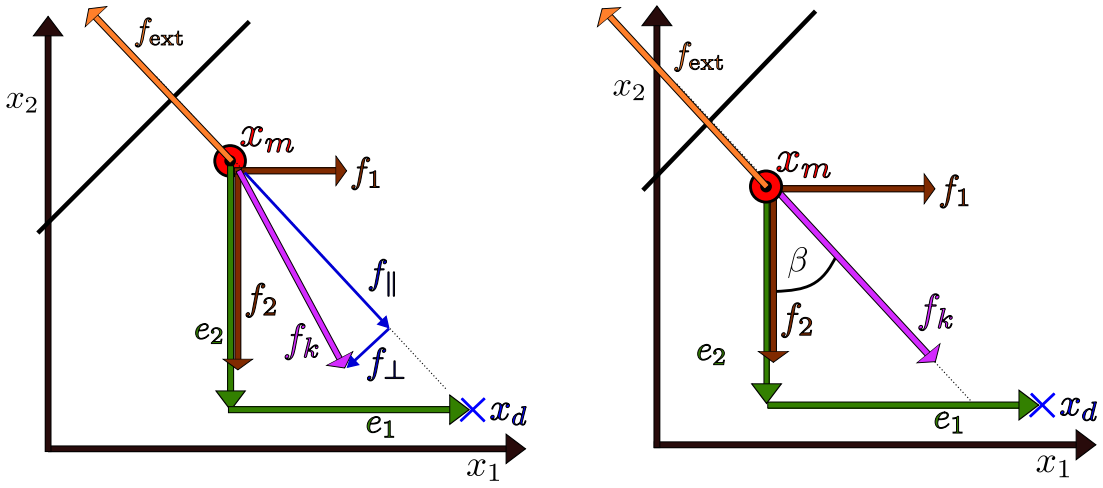
is not aligned with the surface normal of the environment. This leads to a division of $\boldsymbol{f_k}$ into a force $\boldsymbol{f}_{\parallel}$ parallel to the surface normal, and a force $\boldsymbol{f}_{\perp}$ perpendicular to the surface normal. Due to the perpendicular force $\boldsymbol{f}_{\perp}$, the control point $\boldsymbol{x_m}$ gets accelerated perpendicular to the environment's surface. When the system reaches its steady state (Fig. 2.15b), the spring force $\boldsymbol{f_k}$ is aligned and exactly opposed to the external force $\boldsymbol{f}_{\text{ext}}$:

$$\begin{aligned} \boldsymbol{K}\boldsymbol{e} &= -\boldsymbol{f}_{\text{ext}} \\ \Leftrightarrow\ K_{[1,1]}\vec{\boldsymbol{e_1}} + K_{[2,2]}\vec{\boldsymbol{e_2}} &= -\boldsymbol{f}_{\text{ext}} \\ \Rightarrow\ \left\| K_{[1,1]}\vec{\boldsymbol{e_1}} + K_{[2,2]}\vec{\boldsymbol{e_2}} \right\| &= \|\boldsymbol{f}_{\text{ext}}\| \\ \Rightarrow\ \|\boldsymbol{f_1} + \boldsymbol{f_2}\| &= \|\boldsymbol{f}_{\text{ext}}\|\,. \end{aligned} \tag{2.29}$$

The sign of (2.29) is flipped compared to (2.16), because we also flipped the sign of the error $\boldsymbol{e} = \boldsymbol{x_d} - \boldsymbol{x_m}$ such that the force illustrations in Fig. 2.15 are more intuitive.
To limit the norm of the external force $\|\boldsymbol{f}_{\text{ext}}\|$ to $f_{\max}$, we must limit $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$:

$$\begin{aligned} \|\boldsymbol{f_1}\| &< \sin(\beta) f_{\max} \\ \|\boldsymbol{f_2}\| &< \cos(\beta) f_{\max}, \end{aligned} \tag{2.30}$$

where β is the angle between the negative x_1-axis and the outward pointing surface normal of the environment. This can be achieved by setting up a velocity saturation policy like (2.20) for every coordinate direction with force limits as in (2.30). Since these limits depend on the environment geometry through angle β, the velocity limits of the different coordinate directions will change when the environment changes. Additionally, the effects of the stiffness matrix $\boldsymbol{K}$ are no longer transparent because the resulting spring forces are saturated for each environment differently. Therefore, we propose a special variable impedance control scheme to have more transparent control over the interaction forces.



(a) Transient state

(b) Steady State

Figure 2.15: Control and external forces in multi-dimensional contact situation. The control point $\boldsymbol{x_m}$ has deflected the object's equilibrium position (black bar on top left). The environment exerts thus a reactive force $\boldsymbol{f}_{\text{ext}}$ parallel to the objects surface normal. The position errors are defined as $\boldsymbol{e} = \boldsymbol{x_d} - \boldsymbol{x_m}$. Given that $\boldsymbol{K}$ is diagonal, the spring forces $\vec{\boldsymbol{f_i}}$ are aligned with the position errors $\vec{\boldsymbol{e_i}}$. In the transient state *(a)*, the resulting force $\boldsymbol{f_k}$ is split up into a force $\boldsymbol{f}_\parallel$ parallel to the environment's surface normal, and $\boldsymbol{f}_\perp$ perpendicular to it. In the steady state *(b)*, $\boldsymbol{f_k}$ is aligned with the surface normal direction. Angle β represents the slope of the environment with respect to x_1-coordinate.

Rotation of impedance dynamics

The dynamics of a standard Cartesian impedance controller (2.15) with diagonal matrices $\boldsymbol{\Lambda_d}$, $\boldsymbol{D}$, and $\boldsymbol{K}$ are decoupled from each other in the different Cartesian coordinate directions. Hence, in every Cartesian direction a different impedance dynamics can be pursued. Our variable impedance control enables this direction uncoupling for arbitrary directions by simply rotating the coordinate system of the impedance dynamics. Considering an impedance controller that controls the pose of a control point in a three-dimensional operational space[1], the rotated dynamics are

$$\boldsymbol{\Lambda_d R\ddot{e}} + \boldsymbol{DR\dot{e}} + \boldsymbol{KRe} = \boldsymbol{Rf}_{\text{ext}}, \tag{2.31}$$

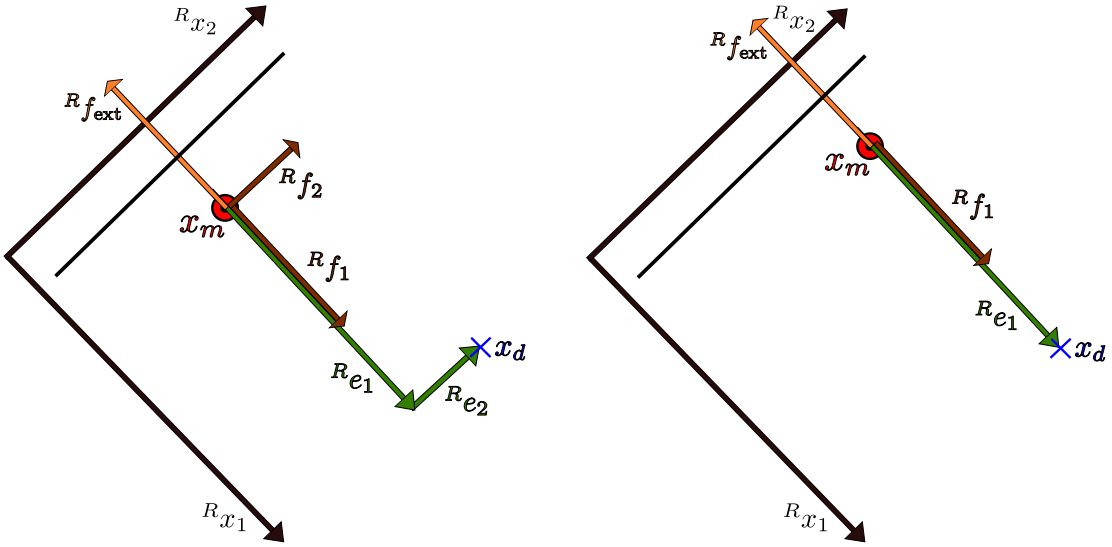
with $\boldsymbol{R}$ being an arbitrary rotation matrix and $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x_d}$. When the desired pose $\boldsymbol{x_d}$ remains constant, (2.32) reduces to

$$\boldsymbol{\Lambda_d R\ddot{x}} + \boldsymbol{DR\dot{x}} + \boldsymbol{KRe} = \boldsymbol{Rf}_{\text{ext}}. \tag{2.32}$$

With $\boldsymbol{\Lambda_d}$, $\boldsymbol{D}$, and $\boldsymbol{K}$ being diagonal, the dynamics in (2.31) and (2.32) are decoupled in the rotated coordinate system ${}^R\chi = \boldsymbol{R}\,{}^C\chi$ with ${}^C\chi$ being the standard Cartesian coordinate frame. Depending on the application, the choice of the rotation matrix $\boldsymbol{R}$ provides a transparent configuration of the impedance dynamics. Let us consider a general setup where a robot is commanded by placing desired poses $\boldsymbol{x_d}$ in its workspace. After the robot has reached one desired pose, the next desired pose is targeted. When specifying the rotation matrix such that it rotates the first coordinate axis of ${}^R\chi$ to the pose error $\boldsymbol{e}$ (see appendix B.2), the dynamics are separated into the pose error direction and the directions perpendicular to it. This enables e.g. a stiff impedance perpendicular to the velocity direction such that the robot won't deflect from its trajectory. At the same time, the impedance in velocity direction can be less stiff such that the robot can be stopped with low contact forces. In this thesis, the first coordinate axis of ${}^R\chi$ has special significance and is named **principal impedance direction ƥ**.

Applying the rotational dynamics (2.32) to the multidimensional contact scenario also benefits the transparency. In fact, when choosing the principal impedance direction to be aligned with the external force $\boldsymbol{f}_{\text{ext}}$, the force limitation problem resembles the one-dimensional case. As illustrated in Fig. 2.16, only the force ${}^{\boldsymbol{R}}\boldsymbol{f_1}$ in principal impedance direction affects the external force $\boldsymbol{f}_{\text{ext}}$ (when neglecting damping and inertia effects). The force ${}^{\boldsymbol{R}}\boldsymbol{f_2}$ does accelerate the control point perpendicular to the external force and vanishes in steady state conditions. This can also be seen when inspecting the rotated dynamics of (2.32) with $\boldsymbol{R}$ making the ex-

[1] I.e. controlling only positions and not orientations.



(a) Transient state (b) Steady State

Figure 2.16: The multi-dimensional contact scenario is presented in the rotated coordinate frame ${}^R\chi$. With $\boldsymbol{K}$ being diagonal, the spring forces ${}^{\boldsymbol{R}}\boldsymbol{f_i}$ are naturally partitioned to point parallel and perpendicular to the external force ${}^{\boldsymbol{R}}\boldsymbol{f}_{\text{ext}}$. For more intuitive visual representation, the pose error $\boldsymbol{e}$ in this figure is defined as $\boldsymbol{e} = \boldsymbol{x_d} - \boldsymbol{x_m}$.

ternal force direction to the principal impedance direction, i.e. $\boldsymbol{R}\boldsymbol{f}_{\text{ext}} = [f_{\text{ext}}, 0, 0]^T$

$$\begin{aligned} \boldsymbol{\Lambda_d} \boldsymbol{R}\ddot{\boldsymbol{x}} + \boldsymbol{D}\boldsymbol{R}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{R}\boldsymbol{e} &= \boldsymbol{R}\boldsymbol{f}_{\text{ext}} \\ \Leftrightarrow \begin{bmatrix} \Lambda_{d,[1,1]}\, {}^R\ddot{x}_1 + D_{[1,1]}\, {}^R\dot{x}_1 + K_{[1,1]}\, {}^Re_1 \\ \Lambda_{d,[2,2]}\, {}^R\ddot{x}_2 + D_{[2,2]}\, {}^R\dot{x}_2 + K_{[2,2]}\, {}^Re_2 \\ \Lambda_{d,[3,3]}\, {}^R\ddot{x}_3 + D_{[3,3]}\, {}^R\dot{x}_3 + K_{[3,3]}\, {}^Re_3 \end{bmatrix} &= \begin{bmatrix} {}^Rf_{\text{ext}} \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{2.33}$$

The above equations are decoupled from each other and the first equation is exactly as in (2.17). Therefore, we can apply the one-dimensional force analysis that have been already done above.

Changing environments

In real human-robot contacts, the environment (in this case the human) behaves differently than the model proposed in Fig. 2.12. Especially, the human does not maintain its posture throughout the contact phase. Hence, the direction of the external force that acts on the robot changes frequently. The rotational dynamics in (2.31) are thus not constant, but are changing due to different force directions. Rewriting (2.31) as

$$\boldsymbol{\Lambda_{d,R}}(\boldsymbol{f}_{\text{ext}})\ddot{\boldsymbol{e}} + \boldsymbol{D_R}(\boldsymbol{f}_{\text{ext}})\dot{\boldsymbol{e}} + \boldsymbol{K_R}(\boldsymbol{f}_{\text{ext}})\boldsymbol{e} = \boldsymbol{f}_{\text{ext,R}} \tag{2.34}$$

highlights the dependencies of the impedance matrices

$$\begin{aligned} \boldsymbol{\Lambda_{d,R}}(\boldsymbol{f}_{\text{ext}}) &= \boldsymbol{\Lambda_d R} \\ \boldsymbol{D_R}(\boldsymbol{f}_{\text{ext}}) &= \boldsymbol{D R} \\ \boldsymbol{K_R}(\boldsymbol{f}_{\text{ext}}) &= \boldsymbol{K R} \end{aligned}$$

on the external force direction. Equation (2.34) can be seen as a variant of **variable impedance control** with variable impedance matrices. Unrestricted variable impedance control is in general not stable because a change in the impedance matrices can inject additional energy into the system. A stability analysis and an adaption of the stiffness and damping profiles as in [KB16] is in our case not possible because firstly, the inertia matrix is not constant, and secondly, the stiffness and damping profiles are not known a priori. A modified version of the energy tank approach in [FSF13] could be implemented to assure stability, however in the case of an empty tank, the principal impedance direction won't align with the external force. Therefore, we propose a stability observer and controller suited for our application scenarios of clamping prevention.

Stability observer and controller

In our tasks, we control the position $\boldsymbol{p}$ and the orientation $\boldsymbol{\phi}$ of the robot's end-effector. Whereas we rotate the coordinate system for the position control according to our needs, we leave the coordinate system for the orientation control constant. To exert only low forces in potential clamping scenarios, we set a low stiffness and a proportional damping value in principal impedance direction. The principal impedance direction is chosen to be the velocity direction in free motion and the external force direction in contact situations. In optimal conditions, the position error $\boldsymbol{e_p}$ is purely in principal impedance direction. Due to perturbations or environment contacts, the parts of the position error $\boldsymbol{e_p}$ perpendicular to the principal impedance direction rise. Since the stiffness in these perpendiculars directions is substantially higher than in principal impedance direction, this injects much energy into the system. Especially if the perpendicular parts of the position error are oscillating and growing, it is an indicator that the system has become unstable.

Whenever an instable system is observed, the stability controller brings the system back to a stable state by extracting energy from the system. The energy extraction is easily incorporated into the impedance direction rotation policy by rotating the principal impedance direction onto the position error direction. Consequently, the error in perpendicular direction diminishes and the error in the principal impedance direction increases. Since the stiffness in the principal impedance direction is lower than in its perpendicular directions, the overall energy of the system decreases. Let $\boldsymbol{R_p}\left(\boldsymbol{e_p}\right) \in \mathbb{R}^{3\times3}$ be the rotation matrix that aligns the x-axis

of the robot base frame ${}^{C}\chi$ to the position error $\boldsymbol{e_p} \neq \mathbf{0}$. Then, the matrix

$$\boldsymbol{R}\left(\boldsymbol{e_p}\right) = \begin{bmatrix} \boldsymbol{R_p}\left(\boldsymbol{e_p}\right) & \mathbf{0_3} \\ \mathbf{0_3} & \boldsymbol{I_3} \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{2.35}$$

rotates only the position $\boldsymbol{p}$ and not the orientation $\boldsymbol{\phi}$ when applied on a pose vector $\boldsymbol{x} = [\boldsymbol{p}, \boldsymbol{\phi}]$. The desired dynamics of the stability controller are chosen as

$$\boldsymbol{\Lambda_d R}\left(\boldsymbol{e_p}\right)\ddot{\boldsymbol{e}} + \boldsymbol{D R}\left(\boldsymbol{e_p}\right)\dot{\boldsymbol{e}} + \boldsymbol{K R}\left(\boldsymbol{e_p}\right)\boldsymbol{e} = \boldsymbol{R}\left(\boldsymbol{e_p}\right)\boldsymbol{f}_{\text{ext}} \tag{2.36}$$

for $\boldsymbol{e_p} \neq \mathbf{0}$. When the position error $\boldsymbol{e_p}$ is zero, the rotation matrix $\boldsymbol{R}(\boldsymbol{e_p})$ from the last non-zero $\boldsymbol{e_p}$ is taken instead.

Stability analysis

Consider the desired closed loop dynamics of (2.36) without external forces:

$$\boldsymbol{\Lambda_d R}\left(\boldsymbol{e_p}\right)\ddot{\boldsymbol{e}} + \boldsymbol{D R}\left(\boldsymbol{e_p}\right)\dot{\boldsymbol{e}} + \boldsymbol{K R}\left(\boldsymbol{e_p}\right)\boldsymbol{e} = \mathbf{0}. \tag{2.37}$$

Since the rotation matrix $\boldsymbol{R}\left(\boldsymbol{e_p}\right)$ only affects the positional dynamics, the orientational dynamics remain constant. Thus, the closed loop dynamics can be split into

$$\boldsymbol{\Lambda_{d,p} R_p}\left(\boldsymbol{e_p}\right)\ddot{\boldsymbol{e}}_{\boldsymbol{p}} + \boldsymbol{D_p R_p}\left(\boldsymbol{e_p}\right)\dot{\boldsymbol{e}}_{\boldsymbol{p}} + \boldsymbol{K_p R_p}\left(\boldsymbol{e_p}\right)\boldsymbol{e_p} = \mathbf{0} \tag{2.38}$$

and

$$\boldsymbol{\Lambda_{d,\phi}}\ddot{\boldsymbol{e}}_{\boldsymbol{\phi}} + \boldsymbol{D_\phi}\dot{\boldsymbol{e}}_{\boldsymbol{\phi}} + \boldsymbol{K_\phi}\boldsymbol{e_\phi} = \mathbf{0} \tag{2.39}$$

The dynamics of (2.38) and (2.39) are decoupled since there is no coupling between the state variables ($\boldsymbol{e_p}$, $\dot{\boldsymbol{e}}_{\boldsymbol{p}}$) and ($\boldsymbol{e_\phi}$, $\dot{\boldsymbol{e}}_{\boldsymbol{\phi}}$). As (2.39) constitutes a constant mass-damper-spring system, the equilibrium point

$$\left(\dot{\boldsymbol{e}}_{\boldsymbol{\phi}}\,, \boldsymbol{e_\phi}\right) = \left(\mathbf{0}\,, \mathbf{0}\right)$$

of this subsystem is globally asymptotically stable if $\boldsymbol{\Lambda_{d,\phi}}$, $\boldsymbol{D_\phi}$, and $\boldsymbol{K_\phi}$ are symmetric and positive definite [KB16]. In the following we will derive constraints for the stability of the variable impedance dynamics in (2.38).
Let us take

$$V(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t) = \frac{1}{2}\left(\boldsymbol{R_p}\dot{\boldsymbol{e}}_{\boldsymbol{p}}\right)^T \boldsymbol{\Lambda_{d,p} R_p}\dot{\boldsymbol{e}}_{\boldsymbol{p}} + \frac{1}{2}\left(\boldsymbol{R_p e_p}\right)^T \boldsymbol{K_p R_p e_p} \tag{2.40}$$

as a Lyapunov candidate function. Since $\boldsymbol{\Lambda_{d,p}}$ and $\boldsymbol{K_p}$ are chosen as constant, diagonal, and positive definite matrices, $V(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t)$ is also positive definite (see appendix B.3). Taking the derivative of $V(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t)$ along the system trajectories leads to

$$\begin{aligned} \dot{V}(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t) &= \left(\boldsymbol{R_p}\dot{\boldsymbol{e}}_{\boldsymbol{p}}\right)^T \boldsymbol{\Lambda_{d,p}}(\dot{\boldsymbol{R}}_{\boldsymbol{p}}\dot{\boldsymbol{e}}_{\boldsymbol{p}} + \boldsymbol{R_p}\ddot{\boldsymbol{e}}_{\boldsymbol{p}}) + \left(\boldsymbol{R_p e_p}\right)^T \boldsymbol{K_p}(\dot{\boldsymbol{R}}_{\boldsymbol{p}}\boldsymbol{e_p} + \boldsymbol{R_p}\dot{\boldsymbol{e}}_{\boldsymbol{p}}) \\ &= \left(\boldsymbol{R_p}\dot{\boldsymbol{e}}_{\boldsymbol{p}}\right)^T \boldsymbol{\Lambda_{d,p}}\dot{\boldsymbol{R}}_{\boldsymbol{p}}\dot{\boldsymbol{e}}_{\boldsymbol{p}} + \left(\boldsymbol{R_p}\dot{\boldsymbol{e}}_{\boldsymbol{p}}\right)^T \boldsymbol{\Lambda_{d,p} R_p}\ddot{\boldsymbol{e}}_{\boldsymbol{p}} \\ &\quad + \left(\boldsymbol{R_p e_p}\right)^T \boldsymbol{K_p}\dot{\boldsymbol{R}}_{\boldsymbol{p}}\boldsymbol{e_p} + \left(\boldsymbol{R_p e_p}\right)^T \boldsymbol{K_p R_p}\dot{\boldsymbol{e}}_{\boldsymbol{p}}. \end{aligned} \tag{2.41}$$

Substituting

$$\boldsymbol{\Lambda_{d,p}} \boldsymbol{R_p} \ddot{\boldsymbol{e}}_{\boldsymbol{p}} = -\boldsymbol{D_p} \boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}} - \boldsymbol{K_p} \boldsymbol{R_p} \boldsymbol{e_p}$$

from (2.38) into (2.41) result in

$$\dot{V}(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t) = (\boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}})^T \boldsymbol{\Lambda_{d,p}} \dot{\boldsymbol{R}}_{\boldsymbol{p}} \dot{\boldsymbol{e}}_{\boldsymbol{p}} - (\boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}})^T \boldsymbol{D_p} \boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}} + (\boldsymbol{R_p} \boldsymbol{e_p})^T \boldsymbol{K_p} \dot{\boldsymbol{R}}_{\boldsymbol{p}} \boldsymbol{e_p}. \tag{2.42}$$

The derivative of a rotation matrix $\boldsymbol{R}$ can in general be written as the product of the rotation matrix itself with a skew symmetric matrix $\boldsymbol{S}$ [Zha16]. Remember that the rotation matrix $\boldsymbol{R_p}$ rotates the robot base frame ${}^{C}\chi$ into the frame ${}^{e}\chi$ whose x-axis is aligned with the error direction $\boldsymbol{e_p}$. The time derivative of $\boldsymbol{R_p}$ can then be expressed as

$$\dot{\boldsymbol{R}}_{\boldsymbol{p}} = [{}^{e}\boldsymbol{\omega}]_{\times} \boldsymbol{R_p}, \tag{2.43}$$

where ${}^{e}\boldsymbol{\omega}$ is the angular velocity of frame ${}^{e}\chi$ expressed in frame ${}^{e}\chi$ and the operator $[\cdot]_{\times}$ transforms its operand into a skew symmetric matrix:

$$[\boldsymbol{\omega}]_{\times} \triangleq \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}.$$

Substituting (2.43) in (2.42) yields

$$\begin{aligned}
&\dot{V}(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t) \\
&= -(\boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}})^T (\boldsymbol{D_p} - \boldsymbol{\Lambda_{d,p}} [{}^{e}\boldsymbol{\omega}]_{\times}) \boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}} + (\boldsymbol{R_p} \boldsymbol{e_p})^T \boldsymbol{K_p} [{}^{e}\boldsymbol{\omega}]_{\times} \boldsymbol{R_p} \boldsymbol{e_p}. && (2.44) \\
&= -(\boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}})^T (\boldsymbol{D_p} - \boldsymbol{\Lambda_{d,p}} [{}^{e}\boldsymbol{\omega}]_{\times}) \boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}} && (2.45) \\
&= -(\boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}})^T \boldsymbol{A_{D\Lambda}} \boldsymbol{R_p} \dot{\boldsymbol{e}}_{\boldsymbol{p}}, && (2.46)
\end{aligned}$$

with

$$\boldsymbol{A_{D\Lambda}} = (\boldsymbol{D_p} - \boldsymbol{\Lambda_{d,p}} [{}^{e}\boldsymbol{\omega}]_{\times}). \tag{2.47}$$

The last summand vanished from (2.44) to (2.45) because the rotation of $\boldsymbol{e_p}$ with $\boldsymbol{R_p}$ produces

$$\boldsymbol{R_p} \boldsymbol{e_p} = [\|\boldsymbol{e_p}\|, 0, 0]^T$$

and the first (actually all) diagonal element of $\left(\boldsymbol{K_p} [{}^{e}\boldsymbol{\omega}]_{\times}\right)$ is 0. Since $\dot{V}(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t)$ does not depend on $\boldsymbol{e_p}$ anymore, $\dot{V}(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t)$ is not negative definite. We cannot conclude asymptotic stability using Lyapunov direct method. However, when $\boldsymbol{A_{D\Lambda}} = (\boldsymbol{D_p} - \boldsymbol{\Lambda_{d,p}} [{}^{e}\boldsymbol{\omega}]_{\times})$ is positive definite (pdf), we can show global asymptotic stability of the equilibrium point $(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}) = (\boldsymbol{0}, \boldsymbol{0})$ with the *Barbashin-Krasovskii theorem* [Lav05]. For that we need to show additionally that

(1) $(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}) = (\boldsymbol{0}, \boldsymbol{0})$ is the only solution of $\dot{V}(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t) = 0$ that can stay in the set $\mathcal{Q} = \{(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}) \in \Omega : \dot{V}(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t) = 0\}$, where Ω is the domain of $V(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t)$ i.e. $V : \Omega \to \mathbb{R}$.

(2) V is *radially unbounded*.

For the first condition, referring to (2.46), we see that

$$\dot{V}(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e}_{\boldsymbol{p}}, t) = 0 \Rightarrow \dot{\boldsymbol{e}}_{\boldsymbol{p}} = \mathbf{0},$$

when $\boldsymbol{A}_{\boldsymbol{D\Lambda}}$ is pdf. Consequently,

$$\dot{\boldsymbol{e}}_{\boldsymbol{p}} = \mathbf{0} \Rightarrow \ddot{\boldsymbol{e}}_{\boldsymbol{p}} = \mathbf{0}.$$

Substituting $\ddot{\boldsymbol{e}}_{\boldsymbol{p}} = \dot{\boldsymbol{e}}_{\boldsymbol{p}} = \mathbf{0}$ into the position dynamics (2.38) reveals that $\boldsymbol{e}_p = \mathbf{0}$ as well. Hence, $(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e}_{\boldsymbol{p}}) = (\mathbf{0}, \mathbf{0})$ is the only solution of $\dot{V}(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e}_{\boldsymbol{p}}, t) = 0$ that can stay in $\mathcal{Q}$. Before we show the second condition, we will shortly repeat the definition of a *radially unbounded* function, as stated in [Lav05].

Definition 1. ***Radially unbounded function***
A function $V : \mathbb{R}^n \to \mathbb{R}$ is radially unbounded if and only if

$$\lim_{\|x\| \to \infty} V(x) = \infty.$$

Looking at (2.40), we see that V is the sum of quadratic forms of each state variable. The matrices of the quadratic forms ($\boldsymbol{R}_{\boldsymbol{p}}^T \boldsymbol{\Lambda}_{\boldsymbol{d,p}} \boldsymbol{R}_{\boldsymbol{p}}$, $\boldsymbol{R}_{\boldsymbol{p}}^T \boldsymbol{K}_{\boldsymbol{p}} \boldsymbol{R}_{\boldsymbol{p}}$) are both symmetric and positive definite (see appendix B.3). Their minimal gain is greater than zero, since all of their eigenvalues are greater than zero. Hence,

$$\left\| \begin{bmatrix} \boldsymbol{e}_{\boldsymbol{p}} \\ \dot{\boldsymbol{e}}_{\boldsymbol{p}} \end{bmatrix} \right\| \to \infty \Rightarrow V \to \infty.$$

To summarize, the equilibrium point $(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e}_{\boldsymbol{p}}) = (\mathbf{0}, \mathbf{0})$ of the dynamics in (2.38) is globally asymptotically stable if $\boldsymbol{A}_{\boldsymbol{D\Lambda}} = (\boldsymbol{D}_{\boldsymbol{p}} - \boldsymbol{\Lambda}_{\boldsymbol{d,p}} [{}^{e}\boldsymbol{\omega}]_{\times})$ is positive definite.
When the desired mass of the robot is uniformly in all coordinate directions, i.e. $\boldsymbol{\Lambda}_{\boldsymbol{d,p}} = \lambda \boldsymbol{I}_{\mathbf{3}}$, with λ being a positive scalar, then $\boldsymbol{A}_{\boldsymbol{D\Lambda}}$ is *pdf* regardless of the angular velocity ${}^{e}\boldsymbol{\omega}$, because

$$\boldsymbol{A}_{\boldsymbol{D\Lambda}} = \boldsymbol{D}_{\boldsymbol{p}} - \boldsymbol{\Lambda}_{\boldsymbol{d,p}} [{}^{e}\boldsymbol{\omega}]_{\times} = \boldsymbol{D}_{\boldsymbol{p}} - \lambda \boldsymbol{I}_{\mathbf{3}} [{}^{e}\boldsymbol{\omega}]_{\times} = \boldsymbol{D}_{\boldsymbol{p}} - \lambda [{}^{e}\boldsymbol{\omega}]_{\times}.$$

The symmetric part of $\boldsymbol{A}_{\boldsymbol{D\Lambda}}$ is $\boldsymbol{D}_{\boldsymbol{p}}$ since the matrix $\lambda [{}^{e}\boldsymbol{\omega}]_{\times}$ is skew-symmetric. As $\boldsymbol{D}_{\boldsymbol{p}}$ is chosen positive definite, $\boldsymbol{A}_{\boldsymbol{D\Lambda}}$ is positive definite as well [Joh70].
When the robot's desired mass is different in the coordinate directions, we must calculate ${}^{e}\boldsymbol{\omega}$ to obtain stability constraints. At any time t, the coordinate frame ${}^{e}\chi = \{{}^{e}\boldsymbol{X}, {}^{e}\boldsymbol{Y}, {}^{e}\boldsymbol{Z}\}$ is aligned with the position error $\boldsymbol{e}_{\boldsymbol{p}}$. Figure 2.17 gives a sketch of the involved frames and vectors and Fig. 2.18 illustrates an example. The coordinate frame ${}^{e}\chi$ rotates with angular velocity ${}^{e}\boldsymbol{\omega}$ due to the error derivative $\dot{\boldsymbol{e}}_{\boldsymbol{p}}$.

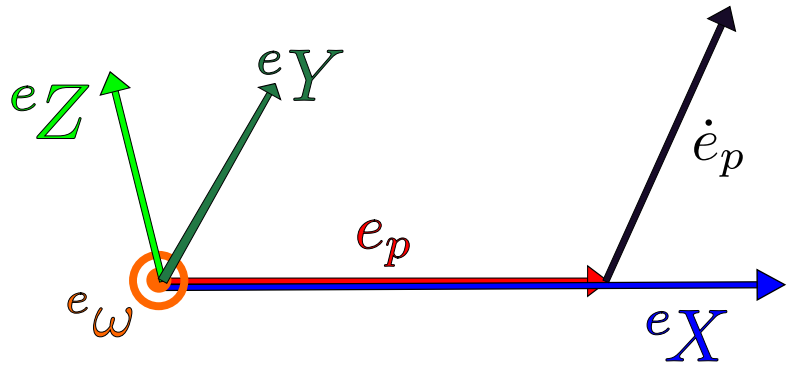


Figure 2.17: Sketch of coordinate frame ${}^{e}\chi = \{{}^{e}\boldsymbol{X}, {}^{e}\boldsymbol{Y}, {}^{e}\boldsymbol{Z}\}$ and its rotation. The frame ${}^{e}\chi$ is obtained by continuously aligning the ${}^{e}\boldsymbol{X}$-axis to the position error $\boldsymbol{e_p}$ direction. $\boldsymbol{e_p}$ and $\dot{\boldsymbol{e}}_{\boldsymbol{p}}$ are in the drawing plane. Since ${}^{e}\boldsymbol{\omega}$ is perpendicular to both $\boldsymbol{e_p}$ and $\dot{\boldsymbol{e}}_{\boldsymbol{p}}$, it points out of the drawing plane.

The rotation axis $\frac{{}^{e}\boldsymbol{\omega}}{\|{}^{e}\boldsymbol{\omega}\|}$ is perpendicular to both $\boldsymbol{e_p}$ and $\dot{\boldsymbol{e}}_{\boldsymbol{p}}$. The angular velocity $\boldsymbol{\omega}$ itself can be calculated with

$$\boldsymbol{\omega} = \frac{\boldsymbol{e_p} \times \dot{\boldsymbol{e}}_{\boldsymbol{p}}}{\|\boldsymbol{e_p}\|^2}. \tag{2.48}$$

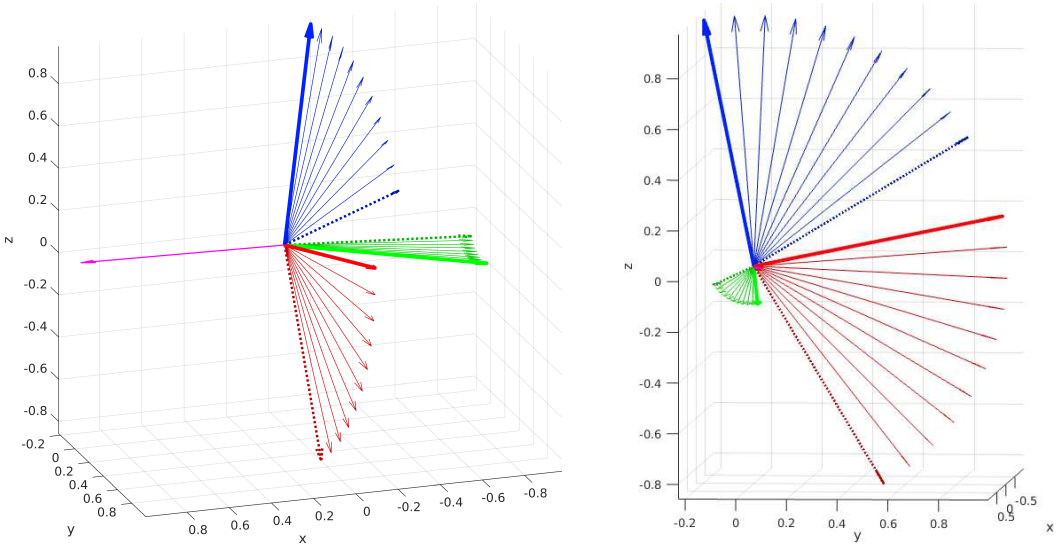
Expressing all vectors in the coordinate frame ${}^{e}\chi$ results in

$$\begin{aligned} {}^{e}\boldsymbol{\omega} &= \frac{\begin{bmatrix} e_{p,[1]} \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} \dot{e}_{p,[1]} \\ \dot{e}_{p,[2]} \\ \dot{e}_{p,[3]} \end{bmatrix}}{(e_{p,[1]})^2} = \frac{1}{(e_{p,[1]})^2} \begin{bmatrix} 0 \\ -e_{p,[1]}\dot{e}_{p,[3]} \\ e_{p,[1]}\dot{e}_{p,[2]} \end{bmatrix} = \frac{1}{e_{p,[1]}} \begin{bmatrix} 0 \\ -\dot{e}_{p,[3]} \\ \dot{e}_{p,[2]} \end{bmatrix} \\ &= \frac{1}{\|\boldsymbol{e_p}\|} \begin{bmatrix} 0 \\ -\dot{e}_{p,[3]} \\ \dot{e}_{p,[2]} \end{bmatrix} \end{aligned} \tag{2.49}$$

where the superscript ${}^{e}\cdot$ has been omitted on the error terms. Note that (2.48) and (2.49) are only valid for $\boldsymbol{e_p} \neq \boldsymbol{0}$. Since we do not rotate the coordinate frame ${}^{e}\chi$ when $\boldsymbol{e_p} = \boldsymbol{0}$, the angular velocity ${}^{e}\boldsymbol{\omega}$ is naturally also zero. In this case, the dynamics in (2.38) constitutes a constant mass-spring-damper system and is asymptotically stable [KB16].

Writing the matrix $\boldsymbol{A_{D\Lambda}}$ from (2.47) in an component-wise structure results in

$$\begin{aligned} \boldsymbol{A_{D\Lambda}} &= (\boldsymbol{D_p} - \boldsymbol{\Lambda_{d,p}} [{}^{e}\boldsymbol{\omega}]_{\times}) \\ &= \begin{bmatrix} D_{p,[1,1]} & \Lambda_{d,p,[1,1]}\, {}^{e}\omega_3 & -\Lambda_{d,p,[1,1]}\, {}^{e}\omega_2 \\ -\Lambda_{d,p,[2,2]}\, {}^{e}\omega_3 & D_{p,[2,2]} & \Lambda_{d,p,[2,2]}\, {}^{e}\omega_1 \\ \Lambda_{d,p,[3,3]}\, {}^{e}\omega_2 & -\Lambda_{d,p,[3,3]}\, {}^{e}\omega_1 & D_{p,[3,3]} \end{bmatrix}. \end{aligned} \tag{2.50}$$



(a) General view (b) View aligned with rotation axis

Figure 2.18: Example of a coordinate frame rotation. The coordinate frame ${}^{e}\chi$ is changed from an initial state (bold, solid red, green, and blue arrows) to a final state (dotted arrows) with intermediate steps (thin and solid arrows). The rotation axis ($\frac{{}^{e}\boldsymbol{\omega}}{\|{}^{e}\boldsymbol{\omega}\|}$) is shown in magenta. When the view is aligned with the rotation axis, the rotation of the coordinate system is clearly visible as a single rotation around this axis.

Positive definiteness is derived from the symmetric part of $\boldsymbol{A_{D\Lambda}}$ [Joh70]. As a multiplication of a matrix with a scalar $\lambda > 0$ does not affect its definiteness we will use

$$\boldsymbol{A^S_{D\Lambda}} = \lambda\frac{1}{2}(\boldsymbol{A_{D\Lambda}} + \boldsymbol{A^T_{D\Lambda}}) \tag{2.51}$$

to evaluate the positive definiteness of $\boldsymbol{A_{D\Lambda}}$. With $\lambda = 2$, (2.51) becomes

$$\boldsymbol{A^S_{D\Lambda}} = \begin{bmatrix} 2D_{p,[1,1]} & {}^e\omega_3\Delta_{\Lambda_1\Lambda_2} & {}^e\omega_2\Delta_{\Lambda_1\Lambda_3} \\ {}^e\omega_3\Delta_{\Lambda_1\Lambda_2} & 2D_{p,[2,2]} & {}^e\omega_1\Delta_{\Lambda_2\Lambda_3} \\ {}^e\omega_2\Delta_{\Lambda_1\Lambda_3} & {}^e\omega_1\Delta_{\Lambda_2\Lambda_3} & 2D_{p,[3,3]} \end{bmatrix}, \tag{2.52}$$

with $\Delta_{\Lambda_i\Lambda_j} = \Lambda_{d,p,[i,i]} - \Lambda_{d,p,[j,j]}$. Since the first entry of the angular velocity ${}^e\omega_1$ is zero, (2.52) simplifies to

$$\boldsymbol{A^S_{D\Lambda}} = \begin{bmatrix} 2D_{p,[1,1]} & {}^e\omega_3\Delta_{\Lambda_1\Lambda_2} & {}^e\omega_2\Delta_{\Lambda_1\Lambda_3} \\ {}^e\omega_3\Delta_{\Lambda_1\Lambda_2} & 2D_{p,[2,2]} & 0 \\ {}^e\omega_2\Delta_{\Lambda_1\Lambda_3} & 0 & 2D_{p,[3,3]} \end{bmatrix}. \tag{2.53}$$

Sylvester's criterion [Gil91] can be used to determine whether $\boldsymbol{A^S_{D\Lambda}}$ (and consequently $\boldsymbol{A_{D\Lambda}}$) is positive definite. For that, we look at the leading principal minors of $\boldsymbol{A^S_{D\Lambda}}$ and arrive at the following conditions for $\boldsymbol{A^S_{D\Lambda}}$ being positive definite:

$$2D_{p,[1,1]} > 0 \tag{2.54}$$

$$4D_{p,[1,1]}D_{p,[2,2]} > ({}^e\omega_3\Delta_{\Lambda_1\Lambda_2})^2 \tag{2.55}$$

$$4D_{p,[1,1]}D_{p,[2,2]}D_{p,[3,3]} > D_{p,[2,2]}({}^e\omega_2\Delta_{\Lambda_1\Lambda_3})^2 + D_{p,[3,3]}({}^e\omega_3\Delta_{\Lambda_1\Lambda_2})^2 \tag{2.56}$$

Condition (2.54) is trivially satisfied when $\boldsymbol{D_p}$ is positive definite and diagonal. Conditions (2.55) and (2.56) are dependent on the angular velocity ${}^{\boldsymbol{e}}\boldsymbol{\omega}$. Whereas condition (2.55) gives a direct upper bound on ${}^e\omega_3$:

$${}^e\omega_3 < \frac{2\sqrt{D_{p,[1,1]}D_{p,[2,2]}}}{\Delta_{\Lambda_1\Lambda_2}},$$

condition (2.56) only states mixed conditions of ${}^e\omega_3$ and ${}^e\omega_2$. As can be derived from (2.55) and (2.56), following circumstances benefits the system's stability:

- small desired inertia differences $\Delta_{\Lambda_1\Lambda_2}$ and $\Delta_{\Lambda_1\Lambda_3}$
- high damping in principal impedance direction $D_{p,[1,1]}$
- slow angular velocity ${}^{\boldsymbol{e}}\boldsymbol{\omega}$.

The dependency of the stability on $D_{p,[2,2]}$ and $D_{p,[3,3]}$ is coupled to the inertia differences $\Delta_{\Lambda_1\Lambda_3}$ and $\Delta_{\Lambda_1\Lambda_2}$, as well as on the angular velocity ${}^{\boldsymbol{e}}\boldsymbol{\omega}$. To state this dependency more clearly, we rewrite condition (2.56) with D_i as a shorter form of $D_{p,[1,1]}$ as

$$f(D_i, \Delta_{\Lambda_i\Lambda_j}, {}^e\omega_i) = 4D_1D_2D_3 - D_2({}^e\omega_2\Delta_{\Lambda_1\Lambda_3})^2 - D_3({}^e\omega_3\Delta_{\Lambda_1\Lambda_2})^2 > 0.$$

Deriving $f(D_i, \Delta_{\Lambda_i\Lambda_j}, {}^e\omega_i)$ partially with respect to D_2 and D_3 gives us

$$\frac{\partial f}{\partial D_2} = 4D_1D_3 - ({}^e\omega_2\Delta_{\Lambda_1\Lambda_3})^2 \tag{2.57}$$

$$\frac{\partial f}{\partial D_3} = 4D_1D_2 - ({}^e\omega_3\Delta_{\Lambda_1\Lambda_2})^2 \tag{2.58}$$

$$\tag{2.59}$$

Hence, when $\frac{\partial f}{\partial D_2}$ and $\frac{\partial f}{\partial D_3}$ are positive, a high value of D_2 and D_3 benefits the stability.

Expressing the angular velocity ${}^e\boldsymbol{\omega}$ as in (2.49) changes (2.54)-(2.56) to

$$2D_{p,[1,1]} > 0 \tag{2.60}$$

$$4D_{p,[1,1]}D_{p,[2,2]} > \frac{1}{\|\boldsymbol{e_p}\|^2}({}^e\dot{e}_{p,[2]}\Delta_{\Lambda_1\Lambda_2})^2 \tag{2.61}$$

$$4D_{p,[1,1]}D_{p,[2,2]}D_{p,[3,3]} > \frac{1}{\|\boldsymbol{e_p}\|^2}\left(D_{p,[2,2]}({}^e\dot{e}_{p,[3]}\Delta_{\Lambda_1\Lambda_3})^2 + D_{p,[3,3]}({}^e\dot{e}_{p,[2]}\Delta_{\Lambda_1\Lambda_2})^2\right), \tag{2.62}$$

which shows the dependence of the position error $\boldsymbol{e_p}$ and its derivative $\dot{\boldsymbol{e}}_{\boldsymbol{p}}$ on the stability.

To summarize the discussion on the stability of the proposed controller, the constraints (2.54)-(2.56) or (2.60)-(2.62) have to be satisfied during all times to guarantee stability. This is trivially the case when the desired inertia is the same in all directions (i.e. $\boldsymbol{\Lambda_{d,p}} = \lambda\boldsymbol{I_3}$). If the desired inertia is not isotropic, then the damping parameters must be chosen such that the conditions are satisfied for an upper bound of ${}^e\boldsymbol{\omega}$ and the desired trajectory $\boldsymbol{x_d}$ has to be adaptively adjusted such that ${}^e\boldsymbol{\omega}$ does not surpass this upper bound (see Sec. B.4 for an example).

Control law

The control law architecture is based on the Cartesian impedance controller from [ASOFH03]. We want to design a control torque $\boldsymbol{\tau_d}$ such that the robot dynamics in (2.13) are changed to our rotational impedance dynamics (2.32). The relation between link velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^7$ and Cartesian velocities $\dot{\boldsymbol{x}} \in \mathbb{R}^6$ is given by the Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{6\times7}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}. \tag{2.63}$$

Taking the derivative of (2.63) and substituting $\ddot{\boldsymbol{q}}$ from the robot model (2.13) produces

$$\begin{aligned}\ddot{\boldsymbol{x}} &= \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\ddot{\boldsymbol{q}} \\ &= \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\bar{\boldsymbol{M}}^{-1}\left(\boldsymbol{\tau_d} + \boldsymbol{\tau}_{\text{ext}} - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{g}\right),\end{aligned} \tag{2.64}$$

where the dependencies on the joint vector $\boldsymbol{q}$ have been omitted. With $\boldsymbol{f_\tau}$ being the new control input, we can choose the control torque $\boldsymbol{\tau_d}$ as

$$\boldsymbol{\tau_d} = \boldsymbol{J}^T \boldsymbol{f_\tau} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{g} - \boldsymbol{\tau}_{\text{ext}} \tag{2.65}$$

to eliminate the effects of the Coriolis, gravity, and external forces on the Cartesian dynamics (2.64):

$$\ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\bar{\boldsymbol{M}}^{-1}\boldsymbol{J}^T \boldsymbol{f_\tau} \tag{2.66}$$

$$\Rightarrow \ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{\Lambda}^{-1}\boldsymbol{F_\tau}. \tag{2.67}$$

The matrix

$$\boldsymbol{\Lambda}(\boldsymbol{x}) = \left(\boldsymbol{J}\bar{\boldsymbol{M}}^{-1}\boldsymbol{J}^T\right)^{-1} \tag{2.68}$$

is also known as the inertia matrix of the operational space [Kha87]. Note that $\boldsymbol{\Lambda}^{-1}$ is only invertible if $\boldsymbol{J}$ has full-rank [SS12]. We obtain the desired dynamics in (2.32) by setting $\boldsymbol{f_\tau}$ to

$$\boldsymbol{f_\tau} = \boldsymbol{\Lambda}\left(\boldsymbol{R}^T\boldsymbol{\Lambda}_d^{-1}\left(-\boldsymbol{K}\boldsymbol{R}\boldsymbol{e} - \boldsymbol{D}\boldsymbol{R}\dot{\boldsymbol{x}} + \boldsymbol{R}\boldsymbol{f}_{\text{ext}}\right) - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) \tag{2.69}$$

as shown below:

$$\begin{aligned}
\ddot{\boldsymbol{x}} &= \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{\Lambda}^{-1}\boldsymbol{f_\tau} \\
&= \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{\Lambda}^{-1}\boldsymbol{\Lambda}\left(\boldsymbol{R}^T\boldsymbol{\Lambda}_d^{-1}\left(-\boldsymbol{K}\boldsymbol{R}\boldsymbol{e} - \boldsymbol{D}\boldsymbol{R}\dot{\boldsymbol{x}} + \boldsymbol{R}\boldsymbol{f}_{\text{ext}}\right) - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) \\
&= \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{R}^T\boldsymbol{\Lambda}_d^{-1}\left(-\boldsymbol{K}\boldsymbol{R}\boldsymbol{e} - \boldsymbol{D}\boldsymbol{R}\dot{\boldsymbol{x}} + \boldsymbol{R}\boldsymbol{f}_{\text{ext}}\right) \\
\Leftrightarrow\ & \boldsymbol{\Lambda}_d\boldsymbol{R}\ddot{\boldsymbol{x}} = -\boldsymbol{K}\boldsymbol{R}\boldsymbol{e} - \boldsymbol{D}\boldsymbol{R}\dot{\boldsymbol{x}} + \boldsymbol{R}\boldsymbol{f}_{\text{ext}} \\
\Leftrightarrow\ & \boldsymbol{\Lambda}_d\boldsymbol{R}\ddot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{R}\boldsymbol{e} + \boldsymbol{D}\boldsymbol{R}\dot{\boldsymbol{x}} = \boldsymbol{R}\boldsymbol{f}_{\text{ext}}.
\end{aligned} \tag{2.70}$$

When $\boldsymbol{R}$ aligns the principal impedance direction $\mathfrak{p}$ to $\boldsymbol{f}_{\text{ext}}$, we can easily limit the external forces to $f_{\max}$ by implementing the velocity saturation law (2.20) in $\mathfrak{p}$ direction[2]. To this end, we set the control force $\boldsymbol{f_\tau}$ to

$$\boldsymbol{f_\tau} = \boldsymbol{\Lambda}\left(\boldsymbol{R}^T\boldsymbol{\Lambda}_d^{-1}\left(-\boldsymbol{f_s} + \boldsymbol{R}\boldsymbol{f}_{\text{ext}}\right) - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right), \tag{2.71}$$

where $\boldsymbol{f_s}$ represents the combination of the spring and damping forces, but is saturated. Referring to (2.33), only the first element $f_{[s,1]}$ needs to be saturated to limit the external forces in steady-state conditions. Hence, we choose $\boldsymbol{f_s}$ as

$$\boldsymbol{f_s} = \begin{bmatrix} D_{[1,1]}\left({}^R\dot{x}_1 - \nu\,{}^R\dot{x}_{[d,1]}\right) \\ \boldsymbol{K}_{[\boldsymbol{2-6}]}\,{}^{\boldsymbol{R}}\boldsymbol{e}_{[\boldsymbol{2-6}]} + \boldsymbol{D}_{[\boldsymbol{2-6}]}\,{}^{\boldsymbol{R}}\dot{\boldsymbol{x}}_{[\boldsymbol{2-6}]} \end{bmatrix}, \tag{2.72}$$

[2] Assuming steady-state conditions.

with

$$
\begin{aligned}
{}^{R}\dot{x}_{[d,1]} &= -\frac{K_{[1,1]}}{D_{[1,1]}}\,{}^{R}e_1 \\
\nu &= \min\left(1, \frac{v_{\max}}{\left\|\dot{x}_{[d,1]}\right\|}\right) \\
v_{\max} &= \frac{f_{\max}}{D_{[1,1]}}.
\end{aligned}
\tag{2.73}
$$

For the experiments in this thesis, a slightly different control law is used. To be exact, we discard the term $\boldsymbol{\dot{J}\dot{q}}$ in (2.71), as it is zero in quasi steady-state conditions and as it would contain noise due to numerical differentiation. Additionally, we add a null space control torque $\boldsymbol{f_\varnothing}$ such that the control torque $\boldsymbol{\tau_d}$ becomes:

$$
\boldsymbol{\tau_d} = \boldsymbol{J^T f_\tau} + \boldsymbol{C\dot{q}} + \boldsymbol{g} - \boldsymbol{\tau}_{\text{ext}} + \boldsymbol{f_\varnothing}, \tag{2.74}
$$

with

$$
\boldsymbol{f_\tau} = \boldsymbol{\Lambda}\left(\boldsymbol{R^T \Lambda_d^{-1}}\left(-\boldsymbol{f_s} + \boldsymbol{R f}_{\text{ext}}\right)\right), \tag{2.75}
$$

and

$$
\boldsymbol{f_\varnothing} = \left(\boldsymbol{I_7} - \boldsymbol{J}^{\#}\boldsymbol{J}\right)\left(K_\varnothing \boldsymbol{e_\varnothing} - D_\varnothing \boldsymbol{\dot{q}}\right). \tag{2.76}
$$

Here, the operator $(\cdot)^{\#}$ refers to the pseudo-inverse, and the null space error $\boldsymbol{e_\varnothing}$ is defined as $\boldsymbol{q_\varnothing} - \boldsymbol{q}$, with $\boldsymbol{q_\varnothing}$ being a taught joint configuration resulting in the desired Cartesian pose $\boldsymbol{x_d}$[3]. Whereas the addition of $\boldsymbol{f_\varnothing}$ to $\boldsymbol{\tau_d}$ does not change the impedance dynamics of the end-effector[4], the elimination of $\boldsymbol{\dot{J}\dot{q}}$ does. As a result, the dynamics are altered to

$$
\boldsymbol{\Lambda_d R\ddot{x}} + \boldsymbol{f_s} = \boldsymbol{R f}_{\text{ext}} + \boldsymbol{\Lambda_d R\dot{J}\dot{q}}, \tag{2.77}
$$

but the effects of $\boldsymbol{\dot{J}\dot{q}}$ are negligible in quasi steady-state conditions, since both $\boldsymbol{\dot{J}}$ and $\boldsymbol{\dot{q}}$ are near zero.

Contact Force Estimation

When a robot enters a contact situation, the contact force normally affects the torques present in the robot's joints. As mentioned in Sec. 1.2.6, much research has been done to identify the torques $\boldsymbol{\tau}_{\text{ext}}$ that arise from external contact forces $\boldsymbol{f}_{\text{ext}}$. For the experiments carried out in this thesis, we use the *generalized momentum* method to estimate these torques [DLM05, DLASHH06]. In clamping scenarios, the external torques $\boldsymbol{\tau}_{\text{ext}}$ arise from contact forces between the robot and the clamped body part, which restricts the robot's motion. As these contact forces

[3]With this null space control torque, the robot stays further away from joint limits.

[4]The dynamics are unchanged when using the inertia-weighted pseudo-inverse as described in [Kha87]

perform active work, the respective external torques $\boldsymbol{\tau}_{\text{ext}}$ can be identified with the generalized momentum method. To infer the contact forces $\boldsymbol{f_c}$ on the contact point $\boldsymbol{p_c}$, we make the same assumptions on contact situations as in [MFDL14], i.e. that only contact forces and no contact torques are transferred during contact. According to the authors of [MFDL14], this leads to more robust estimation results. As a first step, we calculate the contact Jacobian $\boldsymbol{J}_c(\boldsymbol{q}) \in \mathbb{R}^{3\times n}$ that relates the contact point's linear velocity $\dot{\boldsymbol{p}}_{\boldsymbol{c}}$ to the joint velocities $\dot{\boldsymbol{q}}$ as per

$$\dot{\boldsymbol{p}}_{\boldsymbol{c}} = \boldsymbol{J}_c(\boldsymbol{q})\dot{\boldsymbol{q}}.$$

The estimated contact forces $\boldsymbol{f_c}$ are then obtained via pseudo-inversion of the contact Jacobian:

$$\boldsymbol{f_c} = (\boldsymbol{J}_c^T(\boldsymbol{q}))^{\#}\boldsymbol{\tau}_{\text{ext}}. \tag{2.78}$$

For the sake of simplicity, the notions of the (estimated) contact force $\boldsymbol{f_c}$ on the (estimated) contact point and the (estimated) external force $\boldsymbol{f}_{\text{ext}}$ on the end-effector has been used interchangeably for the derivation of the clamping conscious control scheme. However, when imposing the desired impedance dynamics (2.77) or (2.32) on the contact point $\boldsymbol{p_c}$, the contact force $\boldsymbol{f_c}$ must be used, together with the contact Jacobian $\boldsymbol{J}_c$. Furthermore, the Cartesian velocity $\dot{\boldsymbol{x}}$ and error terms $\boldsymbol{e}$ must be mapped to the contact point.

Practical considerations

We partition the workspace of the robot dynamically in two main areas: Areas where human co-workers can be clamped and areas where they cannot. These areas are identified and separated according to Sec. 2.1.1. As explained in Sec. 1.2.2, there have to be made safety precautions even in the non-clamping areas. The danger of human-robot collisions predominantly rise with higher robot speeds and inertias. Hence, the most simple option to reduce the risk of injury is to set a universal speed limit for the robot's end-effector, and ideally for all moving robot parts. More complicated methods also consider the geometry of the end-effector for velocity limits [HHK+12]. For the sake of simplicity, we stay with the former method to reduce the danger in non-clamping areas. As stated above, we employ a velocity saturation policy in the CCIC that is used in potential clamping areas. Depending on the respective body parts, this velocity limit can be substantial lower than in non-clamping areas. Note that in non-clamping areas, it is best to use a different velocity saturation method than the one used in the CCIC. Since the latter also limits the spring force of the impedance to a specific value, it is possible that the resulting actuation force of the robot is not enough to stay precisely on a trajectory in the presence of disturbances.

Force measurement noise

Changes in the principal impedance directions can bring the controlled system to an unstable state, and the stability controller has to bring the system back

to stability. Since the stability controller does not align the principal impedance direction onto the external force direction $\boldsymbol{f}_{\text{ext}}$, the external force does not develop according to the one-dimensional dynamics in (2.33). As a result, the maximal force value $f_{\max}$ can be surpassed temporarily. To reduce the time that the stability controller has to step in, we restrict the changes in the principal impedance direction in the first place. Especially since the external force measurements contain noise, we apply a low-pass filter on these measurements and adapt the principal impedance direction to the filtered signals. This generally introduces a time lag between the external force measurements and the filtered version, also making the principal impedance direction lagging behind. The effects of the signal filtering can be further seen in Sect. 3.1.4.

Obstacle circumvention

Aligning the principal impedance direction $\mathfrak{p}$ to the contact force direction $\boldsymbol{f_c}$ can lead to active obstacle circumvention, as illustrated in Fig. 2.19. When the control point $\boldsymbol{x}$ is in contact with an obstacle, the dynamics get partitioned into directions parallel and perpendicular to $\boldsymbol{f_c}$ (Fig. 2.19 (c)-(e)). The error $\boldsymbol{e}_\perp$ perpendicular to $\mathfrak{p}$ leads to an acceleration of the controlled point $\boldsymbol{x}$ into this direction, since there is no counteracting force (see also (2.33)). The control point thus slides along the contact surface until it either circumvents the obstacle or until $\boldsymbol{f_c}$ lines up with the position error $\boldsymbol{e_p}$.

2.2 Implementation

This section further describes the implementation details of the clamping conscious control pipeline. We begin by reporting how the critical body part dimensions for the clamping identification are obtained in Sec. 2.2.1. Sec. 2.2.2 gives suggestions on how these body part dimensions can be incorporated in collaborative zones. The next sections focus on the clamping identification algorithm. Section 2.2.3 explains the creation of the OBBTree data structure. Further details on how this data structure is traversed in the clamping identification algorithm is outlined in Sec. 2.2.4. The implementation of Early Out Criteria for this traversal is separately described in Sec. 2.2.5. Likewise, an efficient method for implementing the OBB to OBB distance approximation is reported separately in Sec. 2.2.6. Finally, a summary of the entire Clamping Conscious Control pipeline is given in Sec. 2.2.7.

2.2.1 Human Body Part Dimensions

Human body dimensions are varying from person to person. An attempt to analyze various body part dimensions is done in [Til02]. This book lists the respective body part dimensions with respect to *percentile values*. A percentile is a

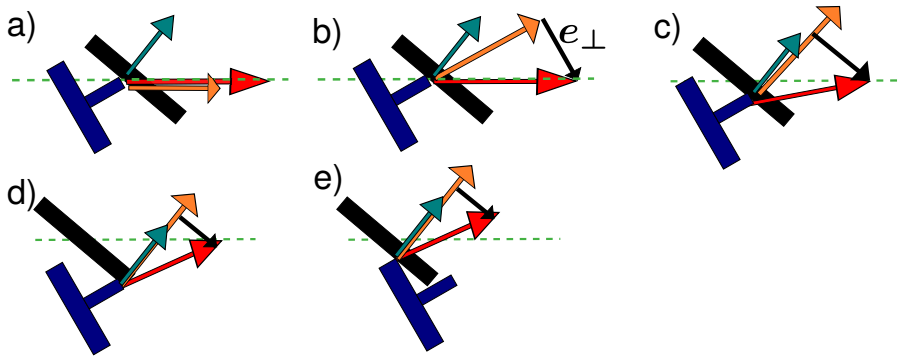


Figure 2.19: Five time instances of an obstacle circumvention scene. The robot's end-effector is shown in blue, an obstacle is sketched in black. The red arrow points to the goal position $\boldsymbol{x_d}$, the orange arrow depicts the current impedance direction $\mathfrak{p}$, the green arrow shows the contact force $\boldsymbol{f_c}$ and the dotted black arrow demonstrates $\boldsymbol{e}_\perp$, the part of the position error $\boldsymbol{e_p}$ perpendicular to $\mathfrak{p}$. The dashed green line is a static line to highlight the robot's movement. When in contact with the obstacle, the principal impedance direction gets aligned step by step to the contact force $\boldsymbol{f_c}$. *a)*: Just before the robot senses the contact situation. The impedance direction is in velocity ($\boldsymbol{e_p}$) direction. *b)*: The principal impedance direction is rotated towards the external force direction. As a result, the error $\boldsymbol{e}_\perp$ increases leading to an acceleration in $\boldsymbol{e}_\perp$ direction. *c)*: The robot's end-effector has slided along the contact surface, changing the direction of the positional error. Additionally, the principal impedance direction is further rotated towards the force direction. These two direction almost line up, leading to a good control over the external force. *d)*: As the end-effector slides along the contact surface, $\boldsymbol{e}_\perp$ is reduced leading to less acceleration in this direction. *e)*: The foremost part of the end-effector has already circumvented the obstacle. The same process is repeated with the new contact point of the end-effector.

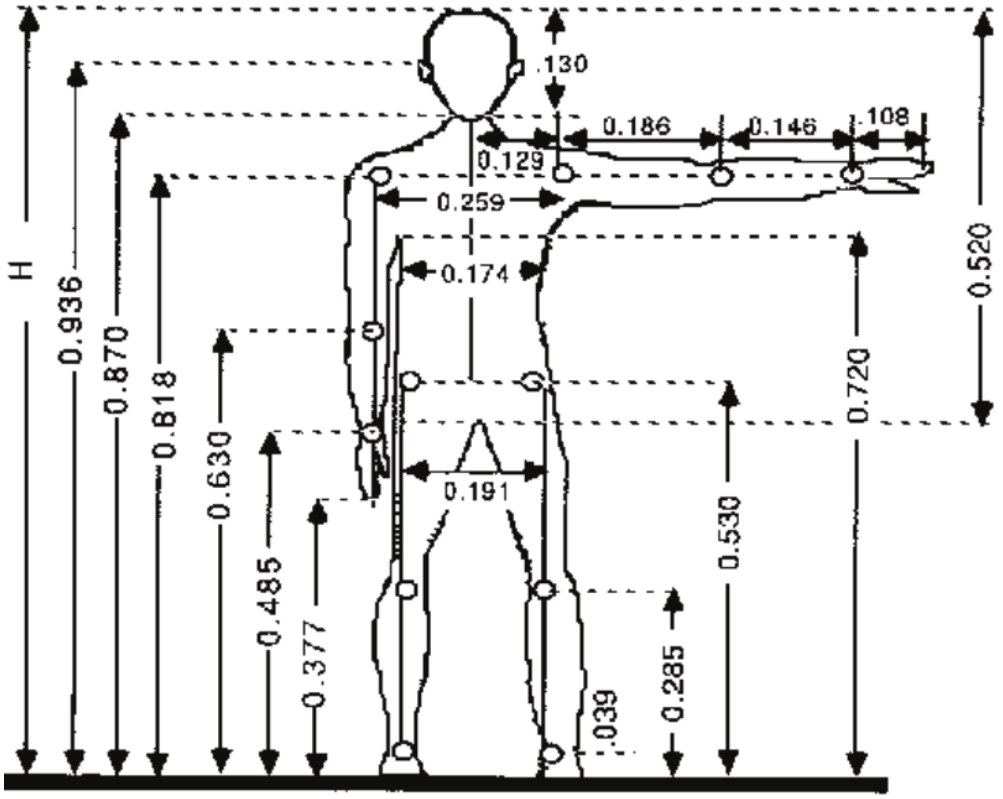


Figure 2.20: Standard human proportions depending on body height *H*. Figure copied from [MT15].

measure in statistics that describes the value for that a specific percentage of the study participants has a lower or identical measured value. E.g. the 99 percentile of male body height[5] is 1.92 meters, meaning that 99 percent of the male subjects have a body height equal or smaller than 1.92 meters. The book especially states various body part dimensions of the 1, 50, and 99 percentile of the US population[5], covering 98 percent of the total US population and thus leaving out the smallest and biggest 1 percent.

Another point of reference for body part dimensions is shown in Fig. 2.20, where certain body part dimensions are stated, relative to the body height. However, the shown body part dimensions describe only a fraction of the available data in [Til02]. Therefore, the measurements in [Til02] are taken for this thesis.

2.2.2 Collaborative Zones

Each collaborative zone contains the minimal and maximal dimensions (size_{min} and size_{max}) of the body parts that can be in this zone. These values are sup-

[5]Age 20-65 years, United States population

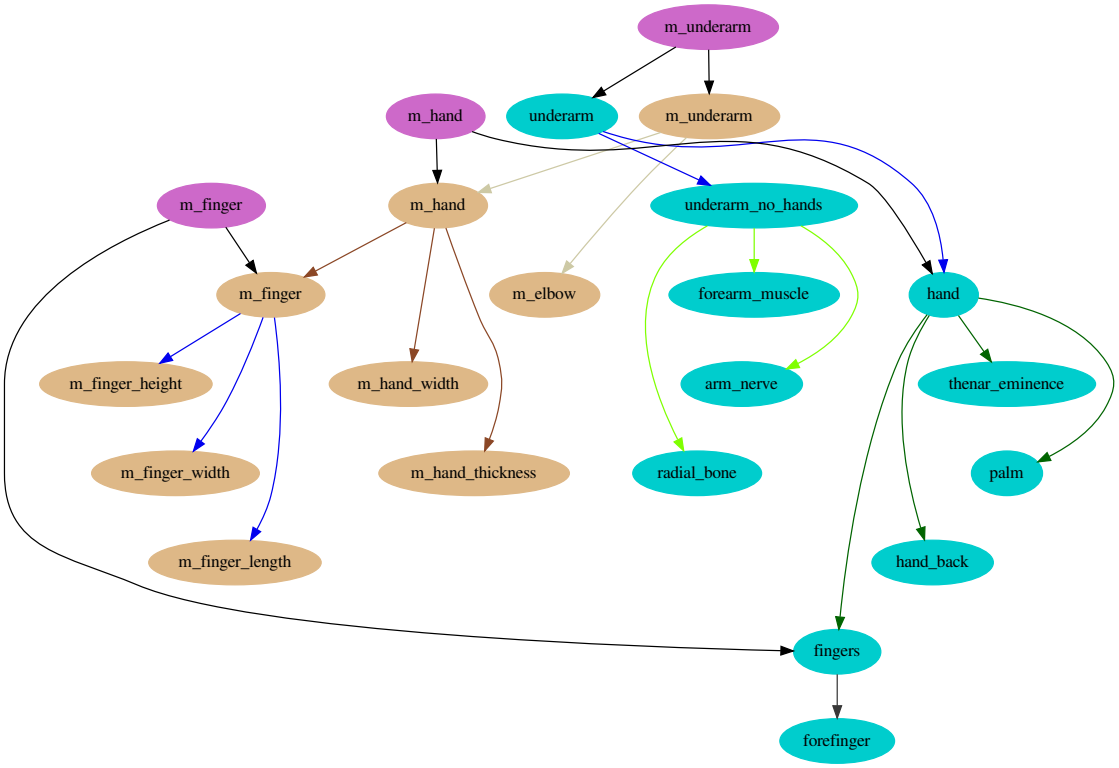


Figure 2.21: For the collaborative zones, body part dimensions (in light brown) are clustered, as well as force limits (turquoise). The body part dimensions are obtained from [Til02], the force limits from [ISO16]. They are connected with each other to form configuration items (purple) for the collaborative zones. The arrows have different colors to better identify parent-child relationships from the clustering. The body part dimensions are split into data from men and women, whereas the force data is regardless of gender. This figure shows only a subset of the male configuration items (prefixed with "m_").

plemented with a maximal force value f_{max} that is maximally permitted for these body parts (taken from [ISO16]). To allow easy configuration of the collaborative zones, the body part dimensions and the force values are clustered and then connected to each other. Figure 2.21 gives an excerpt of these combined configuration items. To configure a collaborative zone, we must specify the geometric extent and pose of the zone in the robot's workspace and add one or more configuration items to it. Depending on the intended customization of the collaborative setup, several approaches are proposed for this configuration.

No customization: If the collaborative setup should work for any co-worker and for any task, the configuration item *entire_body* can be placed over the entire robot's workspace. Additionally, a safety margin can be added to the body part

dimensions (the original data leaves out the smallest and biggest 1 percent of the US population, see Sec. 2.2.1).

Co-worker customization: If the group of co-workers is known, each co-worker could create a configuration file with his body dimensions. The co-worker could change the respective dimensions manually to more accurate ones. If only one co-worker collaborates with the robot during a task, the robot can use solely the configuration file of this co-worker. But, if multiple co-workers are in the robot's workspace at the same time, the robot should use the most conservative dimensions of their configuration files.

Camera guided: Depending on the abilities of the camera and processing system, several ideas could be implemented. First, the camera could detect which co-worker is in which collaboration zone, and load the appropriate configuration file for each collaboration zone. Furthermore, the visual system itself could create dynamic collaboration zones, as regions where the co-workers - or their body parts - currently are. In addition, the configuration files could also be created by the visual system, analyzing for each coworker the minimal and maximal dimension of each body part.

2.2.3 OBBTree Creation

We approximate each robot link with a separate OBBTree, thus, for every robot link, one OBBTree is created. The creation process is a bottom up procedure and was inspired by the point-sphere tree creation of [SSLeS14]. An overview can be seen in Fig. 2.23. The geometries that are approximated can be represented in multiple formats, the only condition is that it must be possible to sample surface points on them. For our simulations and experiments, we use generated surface meshes. The first step in the OBBTree creation process is to sample points from these surface meshes with poisson-disc sampling ([CCS12]) to obtain uniformly distributed points. The sampling process is done twice, once with a minimal point to point distance of 2 mm, and once with 8 mm. Consequently, we have a maximum resolution of 2 mm for the geometries, which is a sufficient value for natural shapes and our use case of human-robot contacts. The points of the 2 mm sampling operation are clustered with k-means, where the points of the 8 mm sampling operation serve as initial cluster centers. For the EOC, it is beneficial that points are grouped that are close together and that have similar normals. Therefore, the k-means algorithm also considers the normals of the points. To adjust the importance ratio of close points to similar normals, either the location vectors, or the normal vectors can be scaled before clustering. Figure 2.24 presents schematically how the lowest cluster can look like when normals have higher importance than the points' location (Fig. 2.24a), and the other way round (Fig. 2.24b). After clustering, an OBB is created for each cluster that comprises every point in this cluster.

To fit the OBB tightly on the cluster points, we use an algorithm working with the convex hull of the point set [Kor15]. To speed up the fitting process for large point sets, we use instead a more approximative method that samples several OBB rotations and keeps the smallest resulting OBB. Afterwards, each OBB is assigned a list of its cluster points, as well as the cluster's center point. The normals of the cluster points are quantized before they are saved in form of a histogram in the OBB node. This is more memory efficient, but predominantly, the OBBTree traversal works faster when there is only a discrete set of normal vectors.

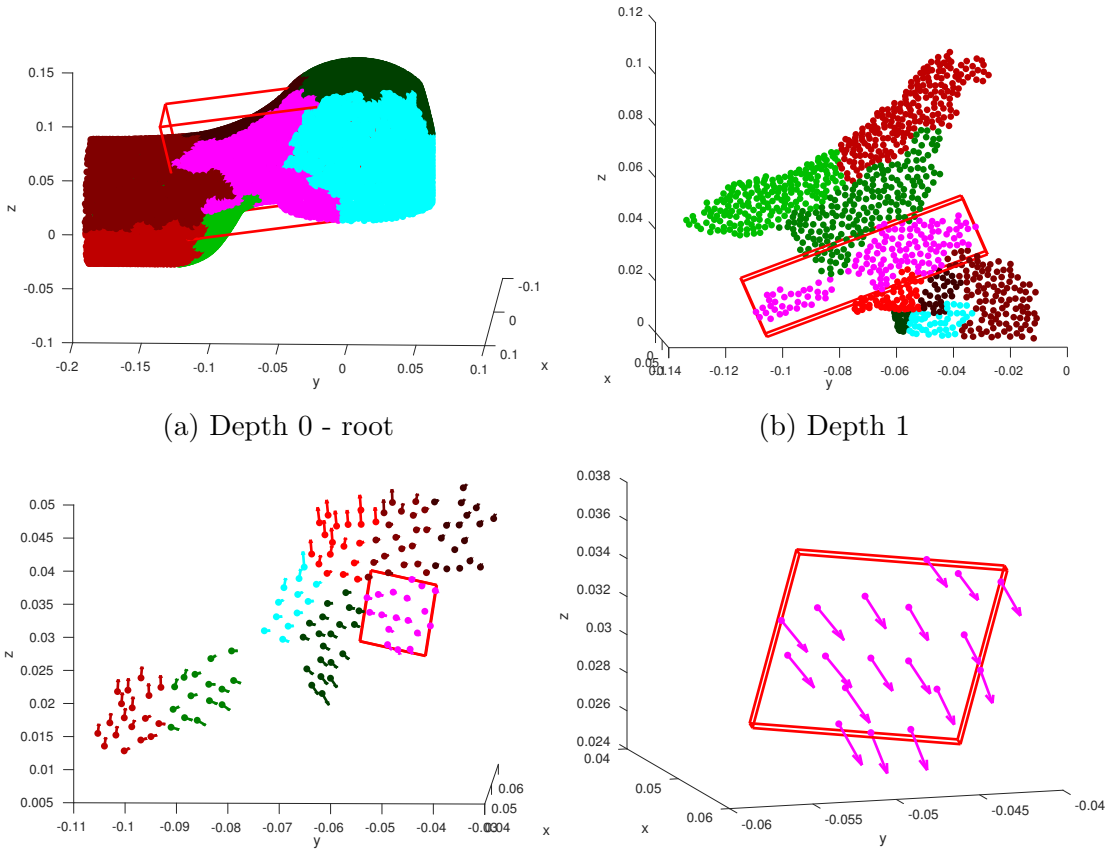
The next steps consists of iteratively clustering the just created OBB nodes. Each of these nodes is represented with their cluster's center point that has been assigned to them. The number of new center points is dependent on the desired branching factor or the depth of the final OBBTree. Since we want the OBBTrees to have equal depth across all links and environments, we choose the number of these new center points adaptively. After the OBB nodes have been clustered, a bigger OBB is created that comprises all assigned points of the clustered OBBs. Analogously to the previous clustering step, this newly created OBB is assigned a list of its OBB child nodes, a concatenated list of their sample points and a histogram of these sample points' normals. This clustering scheme is repeated until the root node of the OBBTree is created. Such a root node is shown exemplarily in Figure 2.22. Traversing it to its leaf nodes also shows the contained clusters of the sample points.

In each cluster iteration, the importance of the point locations with respect to their normals is adjusted. This is needed, because the distances between the clustered points increase with each iteration, whereas the lengths of their normals stay the same, since they are normalized after each iteration. To counteract this change in cluster point distances, the normals are multiplied by a constant greater than one before clustering, and are normalized again after clustering.

For large environments, their sampled points are more numerous than the sampled points for each link. To keep equal depth and branching factors across all OBBTrees, we can increase the sampling radius for the environment. This in turn decreases the resolution for the environment, making the results of the OBBTree traversals less accurate. Instead, we choose to pre-partition the sampled points into multiple sets, for which a OBBTree is created each, following the above procedure.

2.2.4 OBBTree Traversal

As pointed out in Sec. 2.1.3, the identification of potential clamping situations is done by traversing the OBBTrees representing the robot and the environment. This section explains the OBBTree traversal in more detail. The algorithm requires an OBBTree Tree^A for each robot link and at least one OBBTree Tree^B for



(a) Depth 0 - root

(b) Depth 1

(c) Depth 2

(d) Depth 3 - leaf

Figure 2.22: OBB hierarchy and clustering results for link 2 and a total tree depth of 3. Starting from the root node *(a)*, each figure displays the points contained in the node of the respective depth, as well as how these points are clustered for the next depth level. In every figure, the pink cluster is chosen to be visualized in the next depth level. One can also see their respective OBBs in red, as well as the normals of the sampled points in *(c)* and *(d)*.

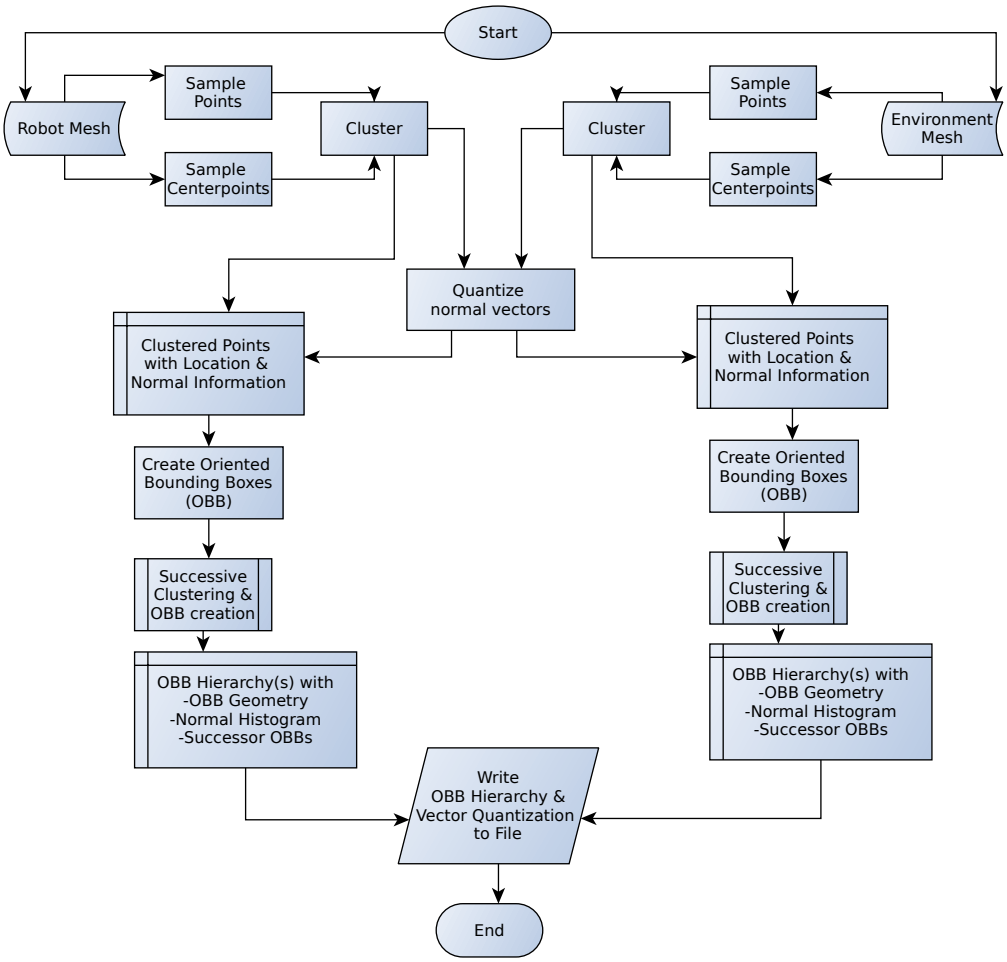
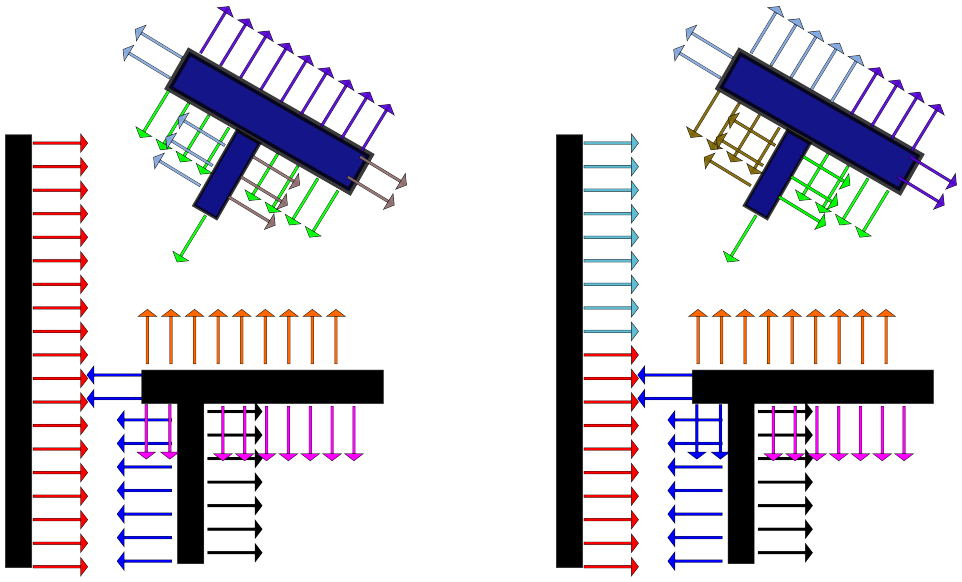


Figure 2.23: Overview of the OBBTree creation process.



(a) Importance on normals (b) Importance on location

Figure 2.24: Different importance of locations and normals in the clustering operation. The example scene consists of the environment in black and the robot end-effector in blue. The arrows represent the normals of the sample points that are clustered. They are displayed with the same color when they belong to the same cluster. In *(a)*, the normals of the sampled points are given a higher importance than the locations in the clustering operation. In *(b)*, the locations are more important than the normals, forming different clusters than in *(a)*.

the environment. All OBBTrees must have the same depth. The traversal algorithm is summarized in algorithm 1. Since the algorithm should identify possible clamping situations per robot link, we start an OBBTree traversal for each robot link Tree^A with each environment Tree^B. In OBBTreeTrav the root nodes of Tree^A and Tree^B are checked for EOC. If an EOC is found, the same Tree^A root node has to be checked with the next OBBTree Tree^B of the environment. Only if all checks with every Tree^B results in an EOC, clamping can be ruled out on this link (line 15). If during such a check no EOC is found, the traversal continues with the child nodes of Tree^A and Tree^B (line 12). The traversal of the child nodes is recursive. As long as no EOC is found between the child nodes, their child nodes are processed. This depth-first traversal ends when the leaf nodes of the OBBTrees are reached and still no EOC has been found along the traversal path (line 23). This means that the currently examined leaf nodes can establish a clamping situation, they constitute a *dangerous leaf node pair*. This information is passed to the recursion layers above and is finally setting the danger flag of the respective link (lines 13, 14 and 4). Consequently, if there is a potential clamping situation on a link, the minimal amount of necessary EOC checks is equal to the depth of the OBBTrees. If there is no clamping situation, this minimal amount is equal to the number of environment OBBTrees. In general, more EOC checks are necessary because often clamping cannot be ruled out by processing only the root nodes, and because there might be an EOC along the traversal path. The latter makes it necessary to check at least one other sibling node. The maximal amount of EOC checks is the same for whether there is a potential clamping situations or not. In both cases all nodes of Tree^A have to be compared with each node of all environment OBBTrees ($\{\text{Tree}^B\}$) when all but the last processed leaf node rules out a clamping situation, but their parent nodes don't. Hence, the maximal amount $n_{\max}$ of EOC checks is

$$n_{\max} = n_{\text{env}} \sum_{i=0}^{d} b^{2i},$$

assuming equal depths d and branching factors b across all OBBTrees, with n_{env} being the amount of environment OBBTrees.

Sorting and caching of nodes

To identify a potential clamping situation, a single *dangerous OBB leaf node pair* suffices. The computation time required to confirm a potential clamping situation is thus dependent on when such a leaf node pair is found during the OBBTree traversal. To speed up the traversal routine, we incorporated a sorting and caching scheme in our clamping identification algorithm. The extended algorithm is shown in algorithm 2. The arrays cLeaf^A and cLeaf^B cache the last dangerous OBB leaf node pair for each link (line 34f.). When a link is checked for clamping situations, this cached OBB leaf pair is processed first (line 8). In usual cobot applications,

Algorithm 1 Clamping identification. Evaluates if there is a possible clamping situation between a set of OBBTrees $\{\text{Tree}^A\}$, each approximating a robot link, and another set of OBBTrees $\{\text{Tree}^B\}$, representing the OBBTrees of the environment. The trees are transformed relative to the world coordinate system by $\{\boldsymbol{T^A}\}$ and $\{\boldsymbol{T^B}\}$ respectively. For the sake of conciseness, the selection of the correct transformation for each OBB node is not portrayed. The function IDENTIFYCLAMPING returns a boolean for each link, expressing if clamping is possible. It uses the functions OBBTREETRAV and OBBNODETRAV that traverse OBBTrees and OBB nodes. The former returns false if clamping is impossible. The latter returns the EOC that has been found, or none.

```
1: function IDENTIFYCLAMPING({Tree^A}, {T^A}, {Tree^B}, {T^B})
2:     for Tree^A in {Tree^A} do                          ▷ iterate over all robot links
3:         linkID ← Tree^A.linkID
4:         danger[linkID] ← OBBTreeTrav(Tree^A, T^A, {Tree^B}, {T^B})
5:     return danger                                      ▷ true if clamping is possible
6:
7: function OBBTREETRAV(Tree^A, T^A, {Tree^B}, {T^B})
8:     for Tree^B in {Tree^B} do          ▷ iterate over all environment OBBTrees
9:         if earlyOut(Tree^A, T^A, Tree^B, T^B) is none then
10:            for Node^A in Tree^A.childs do                  ▷ Go one depth down
11:                for Node^B in Tree^B.childs do
12:                    eo ← OBBNodeTrav(Node^A, T^A, Node^B, T^B)
13:                    if eo is none then              ▷ No EOC has been found
14:                        return true
15:    return false
16:
17: function OBBNODETRAV(Node^A, T^A, Node^B, T^B)
18:    eo ← earlyOut(Node^A, T^A, Node^B, T^B)                          ▷ Sec. 2.2.5
19:    if eo is none and ¬Node^A.isLeaf() then
20:        for Node^A in Node^A.childs do
21:            for Node^B in Node^B.childs do
22:                eo ← OBBNodeTrav(Node^A, T^A, Node^B, T^B)      ▷ recursive
23:    return eo
```

the robot does not move fast. If the clamping identification algorithm runs with a high rate, it is reasonable to assume that the respective links or OBBs do not move far from their previous pose within one cycle. Consequently, dangerous OBB leaf pairs tend to stay dangerous. The minimum amount of EOC checks is thus reduced to only one for each robot link in ideal potential clamping scenarios. The same paradigm - what is dangerous is likely to stay dangerous - also motivates to sort the remaining OBB nodes. An OBB node whose successors delivered a dangerous OBB leaf in the last iteration is likely to deliver a dangerous OBB leaf in the next iteration, too. Even if the last cached OBB leaf node cLeaf^A or cLeaf^B is not dangerous anymore, its siblings might be. Therefore, traversing a queue of OBB nodes that has been sorted according to their last clamping evaluations, reduces the amount of necessary EOC checks in potential clamping situations. Unfortunately, the caching and sorting does not improve the algorithm's performance when there is no potential clamping situation since in this case all nodes of the respective depths have to be processed.

2.2.5 Early Out Criteria

During the OBBTree traversal, the children of a node are only traversed if there is no EOC applicable to this node. If there is an EOC for a node, or a node pair, it means that this node (or pair) does not lead to a clamping situation. As stated in Sec. 2.1.3, there are distance and velocity based EOC.

Distance Early Out Criteria

Given the two distance dependent prerequisites (2.1) and (2.2) for a clamping situation, a clamping situation can be ruled out if one of these prerequisites is not met. Therefore, the current branch of the OBBTree traversal is dismissed if either

$$d_{\min} > \text{size}_{\max} \tag{2.79}$$

or

$$d_{\max} < \text{size}_{\min}. \tag{2.80}$$

These conditions still stay valid when a distance approximation $\tilde{d}_{\min} < d_{\min}$ is used instead of $d_{\min}$, or $\tilde{d}_{\max} > d_{\max}$ instead of $d_{\max}$. An efficient implementation on how to calculate $\tilde{d}_{\min}$ and $\tilde{d}_{\max}$ is further described in Section 2.2.6.

Velocity Early Out Criteria

There are several EOC that are dependent on the relative velocity $\dot{\boldsymbol{x}}_{\boldsymbol{AB}}$ between the two OBBs A and B. Since in our case, the EOC checks are always between a potential moving OBB A from the robot and a static OBB B from the environment, this relative velocity $\dot{\boldsymbol{x}}_{\boldsymbol{AB}}$ is equal to the velocity $\dot{\boldsymbol{x}}$ of the moving OBB A. The first velocity dependent EOC checks whether OBB A moves towards OBB B. Referring

Algorithm 2 Clamping identification with sorting and caching. The environment OBBTrees are given in a queue $\{\boldsymbol{T^B}\}^S$ instead of a set. Additionally, the children nodes within the OBBTree structures are also stored as a queue. The function $putFirst(e, q)$ puts the element e first in queue q, such that it gets retrieved first when traversing the queue.

```
1: cLeaf^A, cLeaf^B ← ∅
2: danger ←zeros(#links)
3: linkID
4: procedure IDENTIFYCLAMPINGSORT({Tree^A}, {T^A}, {Tree^B}^S, {T^B}^S)
5:     for Tree^A in {Tree^A} do                          ▷ iterate over all robot links
6:         linkID ← Tree^A.linkID
7:         if danger[linkID] then
8:             danger[linkID] ← OBBNodeTravSort(                          ▷ fast eval.
                   cLeafs^A[linkID], T^A, cLeafs^B[linkID], T^B)
9:         else
10:            danger[linkID] ← OBBTreeTravSort(Tree^A, T^A, {Tree^B}^S, {T^B}^S)
11:
12: function OBBTREETRAVSORT(Tree^A, T^A, {Tree^B}^S, {T^B}^S)
13:     for Tree^B in {Tree^B}^S do                                ▷ queue traversal
14:         if earlyOut(Tree^A, T^A, Tree^B, T^B) is none then
15:             for Node^A in Tree^A .childs do                    ▷ queue traversal
16:                 for Node^B in Tree^B .childs do
17:                     eo ← OBBNodeTravSort(Node^A, T^A, Node^B, T^B)
18:                     if eo is none then
19:                         putFirst(Node^B, Tree^B .childs)                ▷ sort
20:                         putFirst(Node^A, Tree^A .childs)
21:                         putFirst(Tree^B, {Tree^B}^S)
22:                         return true
23:     return false
24:
25: function OBBNODETRAVSORT(Node^A, T^A, Node^B, T^B)
26:     eo ← earlyOut(Node^A, T^A, Node^B, T^B)                    ▷ Sec. 2.2.5
27:     if eo is none and ¬Node^A.isLeaf() then
28:         for cNode^A in Node^A .childs do                       ▷ queue traversal
29:             for cNode^B in Node^B .childs do
30:                 eo ← OBBNodeTravSort(cNode^A, T^A, cNode^B, T^B)
31:     else if eo is none then
32:         putFirst(cNode^B, Node^B .childs)                      ▷ sort
33:         putFirst(cNode^A, Node^A .childs)
34:         cLeaf^A[linkID] ← cNode^A                  ▷ cache last dangerous leaf pair
35:         cLeaf^B[linkID] ← cNode^B
36:     return eo
```

to Fig. 2.25, OBB A and OBB B are heading for a collision if the conservative displacement vector $\boldsymbol{p_{AB}}$ has a positive projection on $\dot{\boldsymbol{x}}$, i.e.:

$$\boldsymbol{p_{AB}} \cdot \dot{\boldsymbol{x}} > 0. \tag{2.81}$$

It is essential that the vector $\boldsymbol{p_{AB}}$ is used in (2.81) instead of the vector $\boldsymbol{c_{AB}} = \boldsymbol{x_B} - \boldsymbol{x_A}$, expressing the displacement between the center points of the OBBs. Otherwise, the OBB constellation in e.g. Fig. 2.25a would not be classified as a potential colliding case. The conservative displacement vector $\boldsymbol{p_{AB}}$ is created by projecting the OBBs onto the normed velocity $\dot{\boldsymbol{x}}_{\boldsymbol{N}} = \frac{\dot{\boldsymbol{x}}}{\|\dot{\boldsymbol{x}}\|}$ and by adding these projections to $\boldsymbol{c_{AB}}$:

$$\begin{aligned} \boldsymbol{p_{AB}} &= \boldsymbol{x_B} + \sum_i |\beta_i \boldsymbol{b_i} \cdot \dot{\boldsymbol{x}}_{\boldsymbol{N}}| \dot{\boldsymbol{x}}_{\boldsymbol{N}} - \left(\boldsymbol{x_A} - \sum_i |\alpha_i \boldsymbol{a_i} \cdot \dot{\boldsymbol{x}}_{\boldsymbol{N}}| \dot{\boldsymbol{x}}_{\boldsymbol{N}} \right) \\ &= \boldsymbol{c_{AB}} + \dot{\boldsymbol{x}}_{\boldsymbol{N}} \sum_i |\beta_i \boldsymbol{b_i} \cdot \dot{\boldsymbol{x}}_{\boldsymbol{N}}| + |\alpha_i \boldsymbol{a_i} \cdot \dot{\boldsymbol{x}}_{\boldsymbol{N}}|. \end{aligned} \tag{2.82}$$

Reversing (2.81), we get the first velocity EOC, termed *colliding direction* EOC:

$$\boldsymbol{p_{AB}} \cdot \dot{\boldsymbol{x}} < 0. \tag{2.83}$$

The next two EOC depend on the normed velocity $\dot{\boldsymbol{x}}_{\boldsymbol{N}}$ as well as on the normals of the points that are represented by the OBBs. Given the OBB constellation in Fig. 2.26a, only specific sides of the OBBs A and B can enter into collisions. Concerning the moving OBB A, these sides are those with the normals $\boldsymbol{n_1^A}$ and $\boldsymbol{n_2^A}$. The other sides cannot collide with anything static because their normals are opposed to the velocity direction. Therefore, a **moving** OBB node does not need to be further traversed when all its points' normals $\boldsymbol{n_i^A}$ are opposed to the velocity direction such that for all i

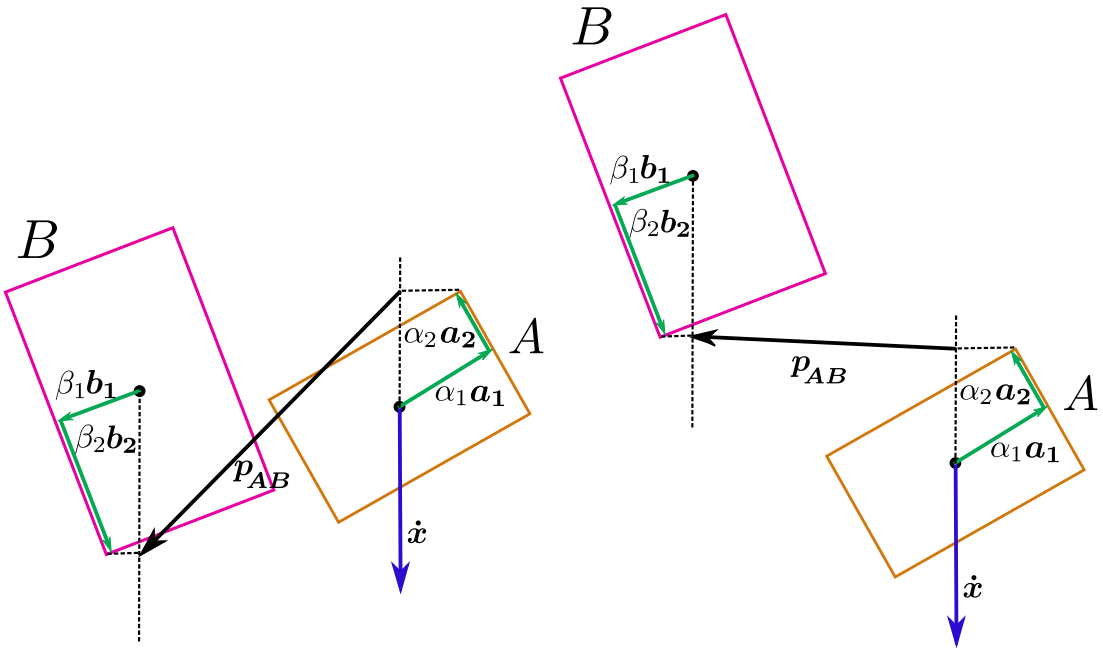
$$\forall i (\boldsymbol{n_i^A} \cdot \dot{\boldsymbol{x}} < 0). \tag{2.84}$$

A similar EOC can be stated for the static OBB B. Here, only the points can enter into collisions that have normals opposed to the velocity direction ($\boldsymbol{n_3^B}$ and $\boldsymbol{n_4^B}$). Hence, a *static* OBB node can be dismissed if

$$\forall i (\boldsymbol{n_i^B} \cdot \dot{\boldsymbol{x}} > 0). \tag{2.85}$$

To refer to these two EOC, (2.84) is termed *link normal* EOC and (2.85) is termed *environment normal* EOC.

Applying the link and environment normal EOC without any modifications during the OBBTree traversal leads to high computational effort. First, the velocity of the link, represented by the OBBTree, needs to be expressed in the same coordinate system as the sample points. Then, the dot product in (2.84) or (2.85)



(a) OBBs in colliding direction (b) OBBs not in colliding direction

Figure 2.25: Illustration of the *colliding direction* EOC. In *(a)*, the EOC (2.83) is not satisfied because the OBBs can potentially collide. In *(b)*, OBB A moves away from OBB B, making a collision impossible.

has to be calculated for each sample point's normal. With the OBBTrees that we are using, having often more than 10^4 sample points, this is an expensive check. Therefore, we quantize the normal vectors of the sample points during the creation of the OBBTrees and save in each node a histogram h of the contained normals. Then, after the OBBTree creation process, we determine for each *code* vector[6] $\boldsymbol{v_i}$ the indices j and k that satisfy following conditions:

$$\begin{aligned} \boldsymbol{v_i} \cdot \boldsymbol{v_j} &> 0 \\ \boldsymbol{v_i} \cdot \boldsymbol{v_k} &< 0. \end{aligned} \tag{2.86}$$

This allows us to query for each code vector the other code vectors that are in the same or the opposite half-sphere as the query vector. Figure 2.26b illustrates such a query operation. This data structure of quantized normals accelerates the check of the link and environment normal EOC. In the beginning of the OBBTree traversal, the velocity of the respective link $\dot{\boldsymbol{x}}$ is transformed into the coordinate system of the OBBTree. Then, the nearest code vector $\boldsymbol{v_i}$ to the transformed velocity is searched. Using the indices j and k from (2.86), the EOC checks (2.84) and (2.85) can be simplified to:

$$\forall q : h(q) > 0 (q \in \{k\}) \tag{2.87}$$

and

$$\forall q : h(q) > 0 (q \in \{j\}). \tag{2.88}$$

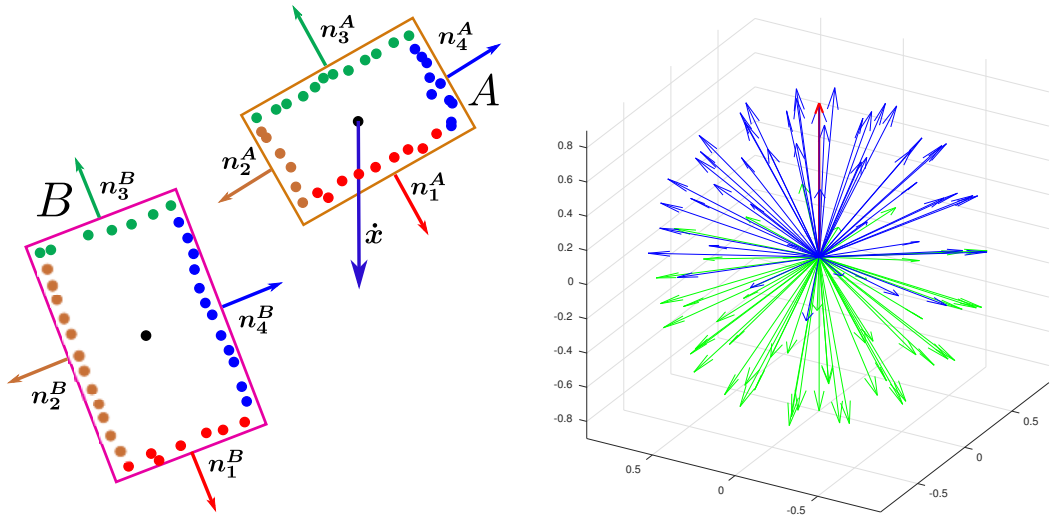
Instead of calculating several dot products, this version of the link and environment EOC only needs to check whether all indices q that have a non-zero entry in the histogram h, are also contained in the set of indices $\{k\}$ (or $\{j\}$). Since we approximate the velocities of each point within the OBBTrees (robot links) with the linear velocity $\dot{\boldsymbol{x}}$ of the respective link, this method is only accurate when the angular velocities of the links are small. Otherwise, points apart from the axis of rotation have a non-negligible different linear velocity than $\dot{\boldsymbol{x}}$. At the expense of higher computational loads, multiple linear velocities can be computed for distributed points on the respective robot link, though.

2.2.6 Efficient OBB Distance Approximation

As stated in [GLM96] and elaborated in [Huy08], the collision test between two OBBs is simplified when calculating the OBB projections in the coordinate system of one of the boxes. Let A be the base OBB, then B is translated relative to A by $\boldsymbol{t}$ and rotated by $\boldsymbol{{}^B_A R}$. The unit vectors $\boldsymbol{a_j}$ from (2.4) have consequently the simple form

$$\boldsymbol{a_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ \boldsymbol{a_2} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \ \boldsymbol{a_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{2.89}$$

[6] I.e. for each vector in the codebook that was used for the quantization



(a) Normal vectors of OBBs (b) Vector quantization

Figure 2.26: In *(a)*, an OBB pair constellation is shown where each OBB consists of four leaf nodes whose sample points are displayed. All sample points within one leaf node have the same normal, which is drawn in the same color as the points. In *(b)*, a codebook of 100 unit vectors is illustrated. During the creation of the OBBTree, every sample point's normal gets assigned to one of these code vectors. For the EOC checks, one can retrieve the vectors $\boldsymbol{v_j}$ (blue) and $\boldsymbol{v_k}$ (green) according to (2.86), given the query vector $\boldsymbol{v_i}$ (red).

With ${}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}$ expressing the rotation from B to A, the unit vectors $\boldsymbol{b_j}$ expressed in frame ${}^{A}\chi$ are the columns of ${}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}$:

$$\boldsymbol{b_1} = \begin{bmatrix} {}^{A}_{B}R_{[1,1]} \\ {}^{A}_{B}R_{[2,1]} \\ {}^{A}_{B}R_{[3,1]} \end{bmatrix}, \; \boldsymbol{b_2} = \begin{bmatrix} {}^{A}_{B}R_{[1,2]} \\ {}^{A}_{B}R_{[2,2]} \\ {}^{A}_{B}R_{[3,2]} \end{bmatrix}, \; \boldsymbol{b_3} = \begin{bmatrix} {}^{A}_{B}R_{[1,3]} \\ {}^{A}_{B}R_{[2,3]} \\ {}^{A}_{B}R_{[3,3]} \end{bmatrix}. \tag{2.90}$$

The 15 separation axes that have to be checked for OBB $\leftrightarrow$ OBB collisions are the 6 face normals of OBB A and B, as well as any combination of $\boldsymbol{a_j} \times \boldsymbol{b_k}$ [GLM96]. Since the distance calculations in (2.5)- (2.8) use the same projections than the collision queries, the same axes have to be checked. To calculate the distances efficiently, we exploit the structure of the OBBs' representations as in [GML00], which lead to different implementations for different axes. In the following, the summands of the projected distances d_i (2.5) and d_i^+ (2.7) are simplified depending on the projection axis.

Face normals of A

When the projection axis $\boldsymbol{l}$ is a face normal of A

$$\boldsymbol{l} = \boldsymbol{a_i},$$

then the OBB A itself is projected on this axis (cf. (2.4)) as

$$r^A = \sum_j |\alpha_j \boldsymbol{a_j} \cdot \boldsymbol{a_i}| = \alpha_i, \tag{2.91}$$

because the face normals $\boldsymbol{a_j}$ and $\boldsymbol{a_i}$ are perpendicular to each other for $i \neq j$. With $\bar{\boldsymbol{R}}$ being the matrix that contains the absolute values of ${}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}$, the projection of OBB B onto $\boldsymbol{l}$ simplifies to

$$r^B = \sum_j |\beta_j \boldsymbol{b_j} \cdot \boldsymbol{a_i}| = \beta_1 \bar{R}_{[i,1]} + \beta_2 \bar{R}_{[i,2]} + \beta_3 \bar{R}_{[i,3]}. \tag{2.92}$$

The projection of the OBBs' translation $|\boldsymbol{t} \cdot \boldsymbol{l_i}|$ is simply

$$|\boldsymbol{t} \cdot \boldsymbol{a_i}| = t_i. \tag{2.93}$$

Face normals of B

Similar simplifications can be done when $\boldsymbol{l} = \boldsymbol{b_i}$:

$$r^A = \sum_j |\alpha_j \boldsymbol{a_j} \cdot \boldsymbol{b_i}| = \alpha_1 \bar{R}_{[1,i]} + \alpha_2 \bar{R}_{[2,i]} + \alpha_3 \bar{R}_{[3,i]} \tag{2.94}$$

$$r^B = \sum_j |\beta_j \boldsymbol{b_j} \cdot \boldsymbol{b_i}| = \beta_i \tag{2.95}$$

$$|\boldsymbol{t} \cdot \boldsymbol{l}| = |\boldsymbol{t} \cdot \boldsymbol{b_i}| \tag{2.96}$$

Equation (2.96) cannot be further simplified, since no a priori assumptions can be made about the translation vector $\boldsymbol{t}$ or the relative rotation ${}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}$.

Cross products of edges

The equations for the projected distances (2.5) and (2.7) are only valid for projection axes of unit length. Since the cross product $\boldsymbol{a_i} \times \boldsymbol{b_j}$ is in general not of unit length, the projection axes must be normalized:

$$\boldsymbol{l} = \frac{1}{\|\boldsymbol{a_i} \times \boldsymbol{b_j}\|} \boldsymbol{a_i} \times \boldsymbol{b_j}.$$

As (2.5) and (2.7) are linear combinations of $\boldsymbol{l}$, the normalizing factor

$$\lambda = \frac{1}{\|\boldsymbol{a_i} \times \boldsymbol{b_j}\|} \tag{2.97}$$

can be factorized:

$$d = \lambda \left(|\boldsymbol{t} \cdot \boldsymbol{l}| - r^A - r^B \right) \tag{2.98}$$

$$d^+ = \lambda \left(|\boldsymbol{t} \cdot \boldsymbol{l}| + r^A + r^B \right) \tag{2.99}$$

To shorten the notation of the following simplifications, we introduce shifted indices:

$$\begin{aligned} [i \gg 1] &= [(i \,\%3) + 1] \text{ and} \\ [i \gg 2] &= [((i+1)\%3) + 1], \end{aligned} \tag{2.100}$$

where % is the modulo operator and i can be replaced by any index. The simplified summands in (2.98) and (2.99) are then

$$r^A = \sum_k |\alpha_k \boldsymbol{a_k} \cdot (\boldsymbol{a_i} \times \boldsymbol{b_j})| = \alpha_{[i \gg 1]} \bar{R}_{[i \gg 2, j]} + \alpha_{[i \gg 2]} \bar{R}_{[i \gg 1, j]} \tag{2.101}$$

$$r^B = \sum_k |\beta_k \boldsymbol{b_k} \cdot (\boldsymbol{a_i} \times \boldsymbol{b_j})| = \beta_{[j \gg 1]} \bar{R}_{[i, j \gg 2]} + \beta_{[j \gg 2]} \bar{R}_{[i, j \gg 1]} \tag{2.102}$$

$$|\boldsymbol{t} \cdot (\boldsymbol{a_i} \times \boldsymbol{b_j})| = |t_{[i \gg 2]} \, {}^A_B R_{[i \gg 1, j]} - t_{[i \gg 1]} \, {}^A_B R_{[i \gg 2, j]}|. \tag{2.103}$$

Derivations of (2.101)-(2.103) can be found in [GML00] and [Huy08].

To approximate the minimum distance $\tilde{d}_{\min}$ (2.6) and the maximum distance $\tilde{d}_{\max}$ (2.8), a system of perpendicular separation axes $\boldsymbol{l}_i$ is required. Any system of perpendicular separation axes will produce an approximated minimum distance $\tilde{d}_{\min}$ less or equal to the real minimum distance $d_{\min}$. Since we want to approximate the minimum distance as close as possible, we choose the biggest projected distance d_i as the first separation axis $\boldsymbol{l}_1$. If this separation axis is a face normal of OBB A or B, the remaining separation axes $\boldsymbol{l}_2$ and $\boldsymbol{l}_3$ are simply the other face normals of OBB A or B, because they are naturally perpendicular to each other. In the case

of $\boldsymbol{l}_1$ being a cross product $\boldsymbol{a_i} \times \boldsymbol{b_j}$, the second perpendicular axis $\boldsymbol{l}_2$ can be chosen to be either $\boldsymbol{a_i}$ or $\boldsymbol{b_j}$. However, the third axis $\boldsymbol{l}_3$ has to be calculated under the constraints of being perpendicular to both $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$. Without loss of generality we take $\boldsymbol{l}_2 = \boldsymbol{a_i}$ as second separation axis. Then, $\boldsymbol{l}_3$ can be obtained with

$$\boldsymbol{l}_3 = \boldsymbol{l}_1 \times \boldsymbol{l}_2 = (\boldsymbol{a_i} \times \boldsymbol{b_j}) \times \boldsymbol{a_i}. \tag{2.104}$$

Since this axis is generally not contained in the previous 15 separation axes, the projected distances d_i and d_i^+ for this axis must be calculated additionally. As with the other separation axes, an efficient implementation is possible through

$$r^A = \sum_k |\alpha_k \boldsymbol{a_k} \cdot ((\boldsymbol{a_i} \times \boldsymbol{b_j}) \times \boldsymbol{a_i})| = \alpha_{[i\gg 1]} \bar{R}_{[i\gg 1,j]} + \alpha_{[i\gg 2]} \bar{R}_{[i\gg 2,j]} \tag{2.105}$$

$$r^B = \sum_k |\beta_k \boldsymbol{b_k} \cdot ((\boldsymbol{a_i} \times \boldsymbol{b_j})| = \frac{1}{\lambda^2}\beta_j + \bar{R}_{[i,j]} \left(\beta_{[j\gg 1]} \bar{R}_{[i,j\gg 1]} + \beta_{[j\gg 2]} \bar{R}_{[i,j\gg 2]}\right), \tag{2.106}$$

with λ from (2.97). The projection of the translation can be simplified to

$$|\boldsymbol{t} \cdot ((\boldsymbol{a_i} \times \boldsymbol{b_j}) \times \boldsymbol{a_i})| = |t_{[i\gg 1]} \, {}^A_B R_{[i\gg 1,j]} + t_{[i\gg 2]} \, {}^A_B R_{[i\gg 2,j]}|. \tag{2.107}$$

For derivations of (2.105)-(2.107) see Sec. A.2 in the appendix. Having calculated the projected distances d_i and d_i^+, the approximations for the minimal and maximal distance $\tilde{d}_{\min}$ and $\tilde{d}_{\max}$ are obtained with (2.6) and (2.8). As summary, the general layout of the distance approximation algorithm is displayed in algorithm 3.

2.2.7 Summary of the Clamping Conscious Control Pipeline

Having now described all individual parts of the Clamping Conscious Control pipeline, we highlight their interdependencies in this section. The following is best understand when looking at the schemata in Fig. 2.27. The starting point of the pipeline is the identification of possible clamping situations. This is done by traversing the OBBTrees of the robot, wich are updated according to the robot configuration $\boldsymbol{q}$, and of the environment as explained in Sec. 2.1.3 and Sec. 2.2.4. The outcome of this process depends, among others, on the dimensions $\text{size}_{\min}$ and $\text{size}_{\max}$ of the body parts that are in the robot's workspace. These are provided by pre-configured collaborative zones (Sec. 2.1.1/ Sec. 2.2.2). The CCIC (Sec. 2.1.6) is influenced by the outcome of this evaluation in multiple ways. If clamping is not possible, the end-effector of the robot can be controlled according to the current task specifications. To avoid high *dynamic* contact forces, it is advisable to limit the velocity though (Sec. 1.2.2). If clamping is possible, regardless of whether a contact has been sensed yet, the velocity is saturated according to (2.72) incorporating the maximal admissible force $f_{\max}$ provided by the collaborative zone. Additionally, the gain matrices of the desired impedance dynamics (2.32)/ (2.77)

Algorithm 3 Distance approximation. Calculates the minimum distance and maximum distance approximations $\tilde{d}_{\min}$ and $\tilde{d}_{\max}$, given the dimensions of two OBBs ($\{\alpha_i\}, \{\beta_i\}$), and their relative transformation ($\boldsymbol{t}, {}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}$).

function DISTANCEAPPROX($\{\alpha_i\}, \{\beta_i\}, \boldsymbol{t}, {}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}$)
  $\{\boldsymbol{a_i}\} \leftarrow baseCoordinateSystem()$ ▷ Eq. (2.89)
  $\{\boldsymbol{b_i}\} \leftarrow cols({}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R})$ ▷ Eq. (2.90)
  $\{\boldsymbol{l_i}\} \leftarrow separationAxes(\{\boldsymbol{a_i}\}, \{\boldsymbol{b_i}\})$ ▷ 15 separation axes
  $\bar{\boldsymbol{R}} \leftarrow |{}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}|$ ▷ Absolute value of each entry
  for $i = 1 : 15$ **do**
    $[r_i^A, r_i^B, tl_i, \lambda] \leftarrow project(\boldsymbol{l_i}, \{\alpha_i\}, \{\beta_i\}, \boldsymbol{t}, {}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}, \bar{\boldsymbol{R}})$
    $d_i \leftarrow \lambda(tl_i - r_i^A - r_i^B)$ ▷ Eq. (2.5) / (2.98)
  $i_{min} \leftarrow indexMin(\{d_i\})$
  if $i_{min} > 6$ **then**
    $l_{16} \leftarrow separationAxis16(i_{min})$ ▷ e.g. Eq. (2.104)
    $[r_{16}^A, r_{16}^B, tl_{16}, \lambda] \leftarrow project(\boldsymbol{l_{16}}, \{\alpha_i\}, \{\beta_i\}, \boldsymbol{t}, {}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}, \bar{\boldsymbol{R}})$
    $d_{16} \leftarrow \lambda(tl_{16} - r_{16}^A - r_{16}^B)$ ▷ Eq. (2.6)
  $v \leftarrow i_{min}$
  $[j, k] \leftarrow perpendicularIndices(v)$ ▷ $\Rightarrow \boldsymbol{l_v} \perp \boldsymbol{l_j} \perp \boldsymbol{l_k} \perp \boldsymbol{l_v}$
  $\tilde{d}_{\min} \leftarrow minDistApprox(d_v, d_j, d_k)$ ▷ Eq. (2.6)
  for $i = [v, j, k]$ **do**
    $d_i^+ \leftarrow \lambda(tl_i + r_i^A + r_i^B)$ ▷ Eq. (2.7)/ (2.99)
  $\tilde{d}_{\max} \leftarrow maxDistApprox(d_v^+, d_j^+, d_k^+)$ ▷ Eq. (2.8)
  return $[\tilde{d}_{\min}, \tilde{d}_{\max}]$
function PROJECT($\boldsymbol{l_i}, \{\alpha_i\}, \{\beta_i\}, \boldsymbol{t}, {}^{\boldsymbol{A}}_{\boldsymbol{B}}\boldsymbol{R}, \bar{\boldsymbol{R}}$)
  $\lambda \leftarrow \|\boldsymbol{l_i}\|$
  switch $\boldsymbol{l_i}$ **do**
    case Face normal of A
      $[r_i^A, r_i^B, tl_i] \leftarrow [Eq.(2.91), Eq.(2.92), Eq.(2.93)]$
    case Face normal of B
      $[r_i^A, r_i^B, tl_i] \leftarrow [Eq.(2.94), Eq.(2.95), Eq.(2.96)]$
    case Axes cross product
      $[r_i^A, r_i^B, tl_i] \leftarrow [Eq.(2.101), Eq.(2.102), Eq.(2.103)]$
    case $\boldsymbol{l_{16}}$
      $[r_i^A, r_i^B, tl_i] \leftarrow [Eq.(2.105), Eq.(2.106), Eq.(2.107)]$
  return $[r_i^A, r_i^B, tl_i, \lambda]$

can be adapted to e.g. make it easier to push the robot back. If there is a potential clamping situation and a contact is sensed (thresholding $\boldsymbol{\tau}_{\text{ext}}$), the contact point $\boldsymbol{p_c}$ is estimated[7]. Until contact is lost, the contact point is controlled and not the end-effector, as otherwise one cannot control the contact forces directly. Having estimated the contact point $\boldsymbol{p_c}$ and the contact Jacobian $\boldsymbol{J_c}$, the contact force $\boldsymbol{f_c}$ can be calculated with (2.78). To simplify the steady-state force limiting, the coordinate system is rotated such that the force only acts in the principal impedance direction $\mathfrak{p}$. The steady-state contact force is then limited by saturating the actuation force $\boldsymbol{f_s}$ through the velocity saturation law (2.72). However, imposing this desired behavior requires multiple feed-forward terms, including the Cartesian mass matrix $\boldsymbol{\Lambda}$ of the robot, external torques $\boldsymbol{\tau}_{\text{ext}}$ and contact forces $\boldsymbol{f_c}$ and some more. The exact control law is given in (2.65) and (2.69) for the idealized case and in (2.74)-(2.76) for the control law used in the experiments.

[7] As we are not monitoring the scene or tracking the human coworkers, we infer the contact point from the static geometry of the scene and the trajectory of the robot. See e.g. Fig. 3.30.

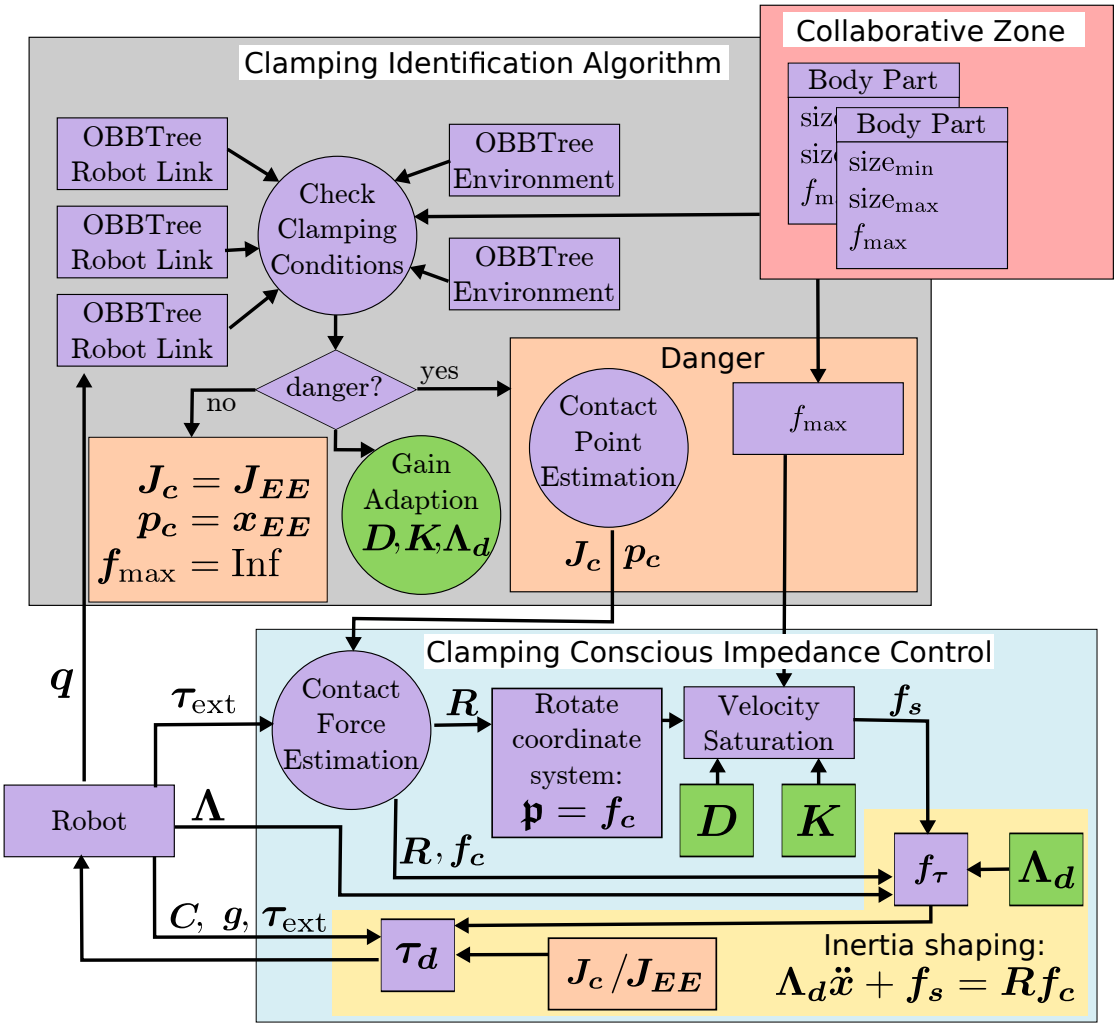


Figure 2.27: Overview of the Clamping Conscious Control pipeline. For the sake of clarity, only the most important entities are displayed.

Chapter 3

Evaluation

This chapter analyses the concepts and implementation of the proposed clamping conscious control pipeline through simulations and experiments. The results are discussed and the shortcomings are reported. To most of them, suitable remedies are proposed as topics of future work.

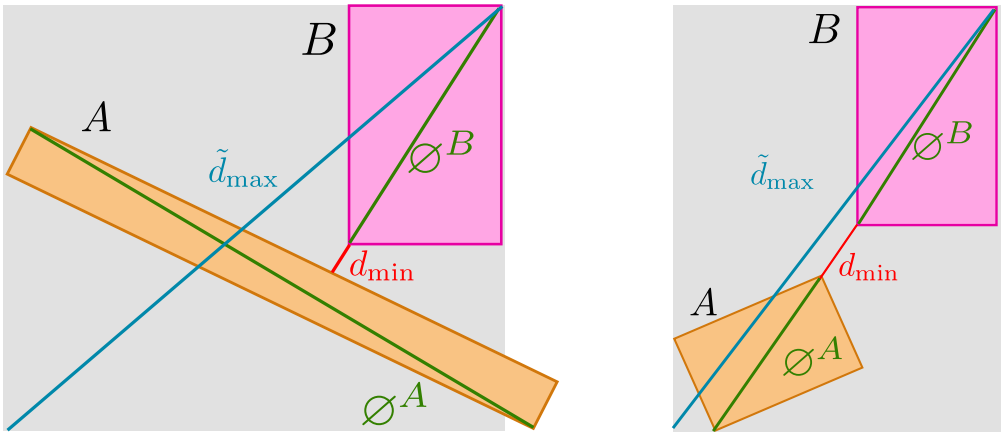
3.1 Simulations

The following simulations were performed on a desktop computer, equipped with an Intel Core i7-2600 processor (4×3.4GHz), 8 GB of RAM, and running Ubuntu 16.04 LTS. The simulations are either written in C++ or in MATLAB.

3.1.1 OBB to OBB Distance Approximation

To evaluate the accuracy and the computational efficiency of our SAT based approximation approach, we compare it to the collision library *flexible collision library (fcl)* [PCM12]. To this end we generate random OBB pairs, whose minimum and maximum distance is calculated or approximated. Each side length of each OBB is an *independent and identically distributed (i.i.d.)* variable on the interval $[0, 50]$[1]. Analogously, each OBB is translated from the origin by another i.i.d. variable with range $[0, 100]$. Finally, each OBB is rotated randomly around a random axis. This procedure produces generally both intersecting and non-intersecting OBB pairs. The minimum and maximum distance of each OBB pair (OBB A and B)is then approximated with our SAT approach as explained in Sec. 2.1.4 and Sec. 2.2.6. As comparison, we calculate the minimum distance $d_{\min}$ of the same OBB pair with fcl. The distance query in fcl is computed with the GJK algorithm, together with an iterative procedure to find the minimum distance of the resulting Minkowski sum to the origin. Since this algorithm returns the exact minimum distance, it is used as ground truth. Although it is possible

[1]The unit is dropped, since it can be any unit of length.



(a) Constellation 1 (b) Constellation 2

Figure 3.1: Comparison of our maximum distance approximation $\tilde{d}_{\max}$ and $d_{\overline{\max}}$ as in (3.1). In the OBB constellation *(a)*, $d_{\overline{\max}}$ is far bigger than $\tilde{d}_{\max}$. In constellations where $d_{\min}$ is a corner to corner distance, $d_{\overline{\max}}$ can be a better approximation than $\tilde{d}_{\max}$. In both shown constellations, $\tilde{d}_{\max}$ is calculated by taking the edges of OBB B as projection axes.

to extend the GJK algorithm to return the maximum distance $d_{\max}$[2], we simply calculate the distances from every corner of OBB A to every corner of OBB B. Since the maximum distance thereof is also the maximum distance between the OBBs, we take this maximum distance as ground truth (see Sec. A.1 for an explanation). Additionally, we compare our maximum distance approximation $\tilde{d}_{\max}$ with a coarse, but fast approximation $d_{\overline{\max}}$ that is easily obtained when already having calculated the minimum distance $d_{\min}$:

$$d_{\overline{\max}} = d_{\min} + \varnothing^A + \varnothing^B, \tag{3.1}$$

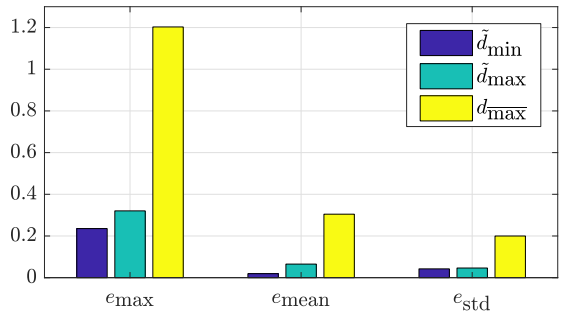
with $\varnothing^A$ ($\varnothing^B$) being the edge to edge diagonal of OBB A (OBB B). A visual comparison of $d_{\overline{\max}}$ and $\tilde{d}_{\max}$ is shown in Fig. 3.1.

Results

To compare the accuracy of the distance approximations, we calculate how much the approximated quantity x differs from the ground truth x_{gt} with

$$e = \left| \frac{x - x_{gt}}{x_{gt}} \right|.$$

[2]The maximum distance is the point of the Minkowski sum farthest away from the origin.



x	$e_{\max}$	e_{mean}	e_{std}
$\tilde{d}_{\min}$	0.235	0.0193	0.0424
$\tilde{d}_{\max}$	0.321	0.0652	0.0463
$d_{\overline{\max}}$	1.203	0.305	0.120

Figure 3.2 & Table 3.1: Accuracy of distance approximations for 10000 samples.

Multiple (n) error evaluations e_i are then aggregated and expressed with

$$e_{\max} = \max_i e_i \tag{3.2}$$

$$e_{\mathrm{mean}} = \frac{1}{n}\sum_i e_i \tag{3.3}$$

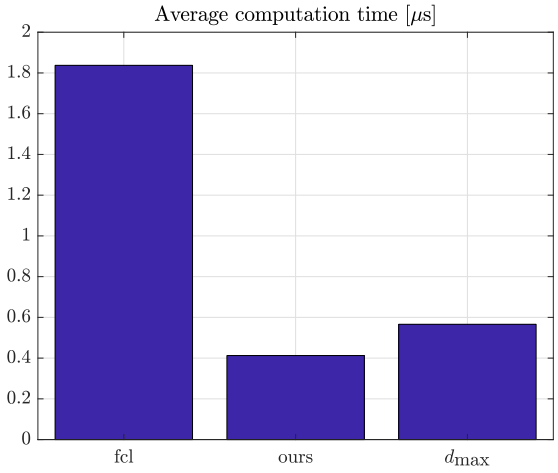
$$e_{\mathrm{std}} = \sqrt{\frac{1}{n-1}\sum_i (e_i - e_{\mathrm{mean}})^2}, \tag{3.4}$$

representing the maximum and mean error, as well as the standard deviation thereof. Figure 3.2 and Tab. 3.1 reports the accuracy evaluation of the approximations $\tilde{d}_{\min}$, $\tilde{d}_{\max}$ and $d_{\overline{\max}}$. Colliding OBB constellations have been excluded for this analysis. Our minimum distance approximation algorithm reliably detects these cases and returns the correct value of $\tilde{d}_{\min} = 0$. The computation times that are needed on average for a single OBB to OBB distance evaluation are shown in Fig. 3.3 and Tab. 3.2. Our SAT based approximation approach is more than 4 times faster than the GJK implementation in fcl. Even the ground truth computation for the maximum distance $d_{\max}$ exceeds the time needed for calculating both $\tilde{d}_{\min}$ and $\tilde{d}_{\max}$ with our SAT approach. The average error of $\tilde{d}_{\min}$ is less than 2 percent on minimum distance calculations. For maximum distance calculations, the average error of our SAT based approach is almost 5 times less than the ad hoc implementation 3.1 based on the minimum distance.

3.1.2 Clamping Identification

To evaluate various parameters of our clamping identification algorithm, described in Sec. 2.2.4, we recorded a trajectory of the Panda robot [EMI17] for 10 seconds. Approximately every 1 ms, the current joint angles and the elapsed time is saved, leading to a total of 9328 entries[3]. The recorded trajectory is visualized in Fig. 3.4. The green environment is composed of different shapes and

[3]After recording the data, the respective thread sleeps for 1 ms. Hence, the effective record rate is less than 1 kHz.

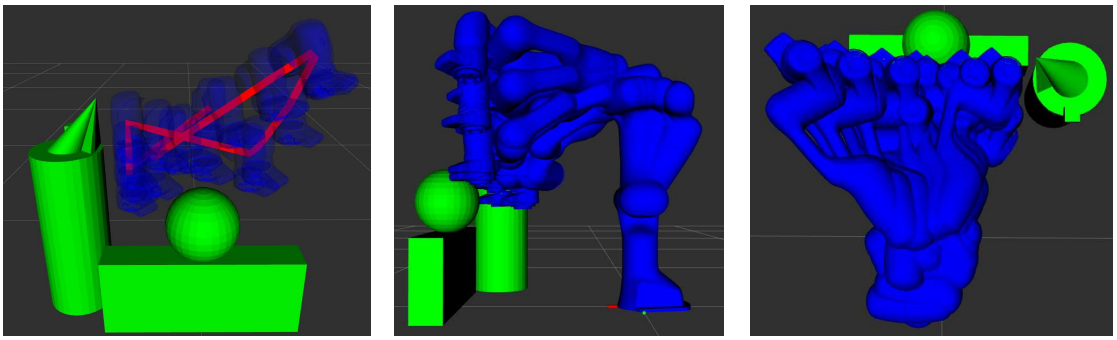


	fcl	ours	$d_{\max}$
time [μs]	1.84	0.41	0.57

Figure 3.3 & Table 3.2: Mean computation time of distance approximations averaged on 10000 samples. The computation time includes the computation of $\tilde{d}_{\min}$ and $\tilde{d}_{\max}$ for *our* method; for *fcl* it includes the computation of $d_{\min}$ and $d_{\overline{\max}}$. The computation time for the ground truth maximum distance is labeled as $d_{\max}$.

placed such that the robot does not collide with it. For every parameter set of our clamping identification algorithm, we replay the robot's trajectory and measure the time needed to determine whether there is a potential clamping situation or not. We also compare the results with a state of the art minimum distance computation from *fcl*[PCM12]. This implementation constructs a hierarchy of OBBs and *Rectangular Swept Spheres (RSSs)* [LGLM00] for a given mesh. Distance queries are then solved by traversing this hierarchy, and by application of the GJK and *Expanding Polytope Algorithm (EPA)* [VDB01]. Since its computation time is dependent on the complexity of the mesh, we use simplified meshes of 1000 faces per robot link and environment. An analysis of the accuracy of this approximation is given in Sec. 3.1.3.

Firstly, we only consider the distance EOC in our algorithm. We simulate two similar scenarios, both with the same environment and robot trajectory as in Fig. 3.4, but with different body parts in the collaborative zones. Both collaborative zones extend over the entire robot workspace, the first contains the element *m_head*, the other contains the element *m_abdominal_region*. Their force limits, as well as their minimum and maximum dimensions $\text{size}_{\min}$ and $\text{size}_{\max}$ are listed in Tab. 3.3. For both scenarios, we evaluate different parameters for the OBBTree creation, namely the depth $d_{\clubsuit}$ of the OBBTree, and the influence of the normals during clustering. Concerning the OBBTree traversal, we analyze the effects of using the caching and sorting scheme, as introduced in Sec. 2.2.4. The necessary OBB to OBB distance calculations are once performed by our SAT based approximation approach, and once with the GJK algorithm as described in Sec. 3.1.1. In



(a) Front view (b) Side view (c) Top view

Figure 3.4: Robot trajectory used for the clamping identification algorithm. There is no collision with the environment, however, especially the end-effector comes close to the environment. In *(a)*, the trace of the trajectory is shown in red. The robot starts near the bottom left vertex of the trajectory and finishes at the line-strip's end. For the sake of clarity, only the last two links are shown in this view. The remaining views display all robot links.

the following, the reported computation times refer to the time needed to evaluate the clamping situation for a single robot configuration. This evaluation is composed of an evaluation for every link, such that the reported computation times are the sums of the computation times of the individual links.

Results

Figure 3.5 and Fig. 3.6 show that for both collaborative zones, the caching and sorting scheme leads to a significant drop in average computation time across all tested OBBTrees. When approximating the robot and the environment with OBB-Trees of depth 9, the average computation time for the *m_head* scenario drops from 0.9 ms to 0.2 ms, and from 4.4 ms to 0.9 ms in the *m_abdominal_region* scenario. This drop is due to the reduced number of necessary EOC checks. As can be seen in Fig. 3.7b, the number of total EOC checks drops approximately by the same factor as the computation time. The fastest mean computation time is obtained with an OBBTree of depth 6 in the *m_head* scenario, but with depth of 9 in the *m_abdominal_region* scenario. However, for *m_head*, the OBBTree with depth 9 is almost as fast as the depth 6 tree. The exact mean computation times of fast parameter combinations is shown in Tab. 3.4, together with the average amount of EOC checks per clamping evaluation. It can be seen that when using our SAT approach, more EOC checks are necessary, compared to the GJK implementation. Since the minimum distance approximation $\tilde{d}_{\min}$ from the SAT approach is always equal or smaller than the real minimum distance $d_{\min}$, less OBBTree branches can be potentially dismissed according to the minimum distance EOC. The faster computation time of the minimum distance approximation

Table 3.3: Excerpt of body limits for the configuration of collaborative zones. The values are obtained according to Sec. 2.2.1 and Sec. 2.2.2.

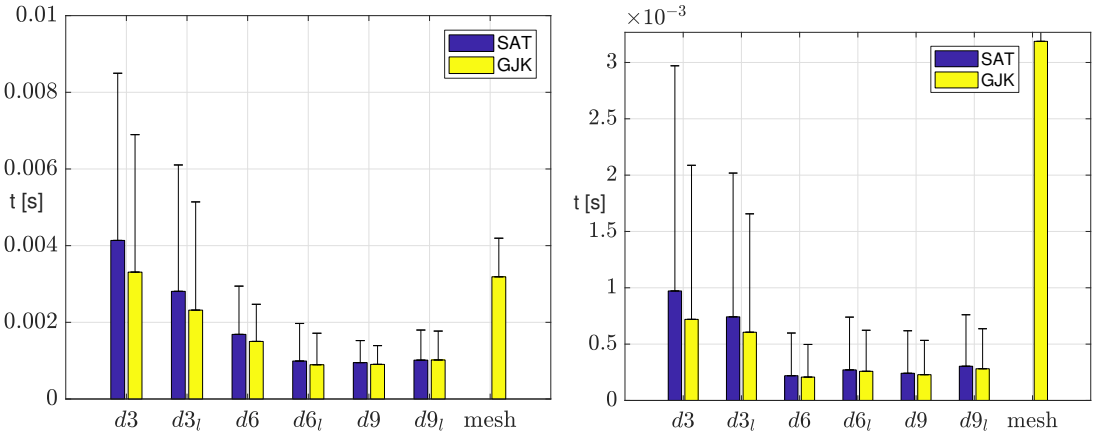
body part	size_{min} [m]	size_{max} [m]	f_{max} [N]
m_head	0.142	0.251	65
m_abdominal_region	0.201	0.429	110
m_hand	0.013	0.117	140
m_above_legs	0.013	0.523	65
m_legs	0.076	0.197	130

Table 3.4: Selection of fastest mean computation times t_{avg} and the respective average amount of EOC checks per robot configuration. The label identifies a unique set of the configuration parameters $\{d_{\clubsuit}$, sort , OBB to OBB $\}$.

body part	label	$d_{\clubsuit}$	sort	OBB to OBB	t_{avg} [ms]	n_{avg}
m_head	1	6	yes	GJK	0.21	64
m_head	2	6	yes	SAT	0.22	82
m_head	3	9	yes	GJK	0.23	69
m_head	4	6_l	yes	GJK	0.26	77
m_abdominal_region	3	9	yes	GJK	0.9	251
m_abdominal_region	5	9	yes	SAT	1.2	393
m_abdominal_region	4	6_l	yes	GJK	2.1	538

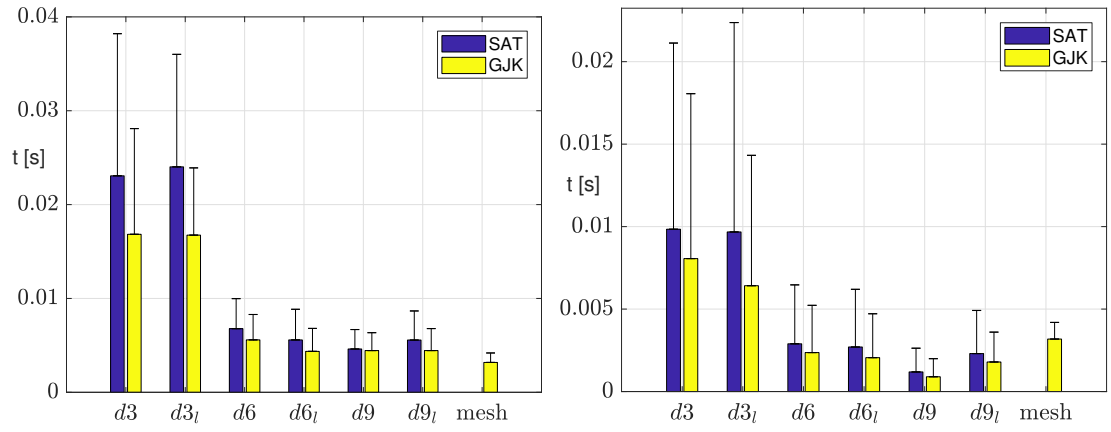
with SAT does not compensate for the higher amount of necessary EOC checks in our simulations, especially since the OBBTree traversal involves additional computational overhead. Although our SAT based approach calculates on average a better maximum distance approximation $\tilde{d}_{\text{max}}$ than the ad-hoc approximation $d_{\overline{\text{max}}}$ in (3.1), this advantage is not of high importance in the simulated scenarios. For the *m_abdominal_region* scenario for example, no early-out check lead to the dismissal of a branch due to the maximum distance when using the OBBTree with depth 9 that has the best performance.

Taking both simulation scenarios into account, an OBBTree depth $d_{\clubsuit}$ of 9 is the most performant one. In our setup, the OBBTrees with $d_{\clubsuit} = 9$ have on average a branching factor b of 2.07, compared to 2.99 for $d_{\clubsuit} = 6$ and 8.96 for $d_{\clubsuit} = 3$. There is no clear trend whether the integration of the sample point normals during the clustering processes is beneficial to the computation time. However, for $d_{\clubsuit} = 9$, using the OBBTree with normal information is slightly faster than using its counterpart for both examined collaborative zones.



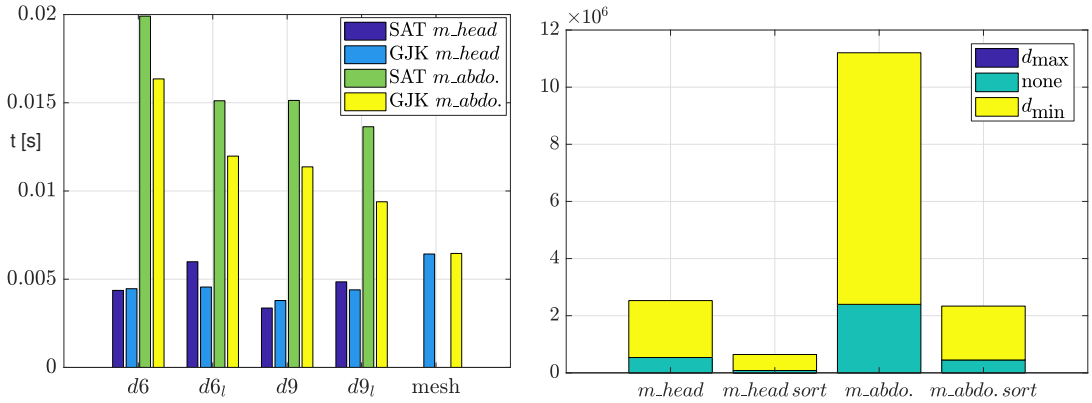
(a) No sorting or caching, see algorithm 1 (b) Sorting and caching, see algorithm 2

Figure 3.5: Computation time analysis with *m_head* as critical body limit. The blue and yellow bars represent the average time needed for the clamping identification of a single robot configuration. The black bars on top of them show the standard deviation of the averaged time samples. For the blue bars, the distance between two OBBs is approximated using our SAT approach (algorithm 3), the GJK algorithm is used for the yellow bars. The values on the abscissa define the depth $d_{\clubsuit}$ of the OBBTree, where the subscript $\cdot_l$ denotes that only the point locations are used for clustering. The reference method of fcl is displayed at the entry *mesh*.



(a) No sorting or caching, see algorithm 1 (b) Sorting and caching, see algorithm 2

Figure 3.6: Computation time analysis with *m_abdominal_region* as critical body limit. Bars and abscissa analogously to Fig. 3.5.



(a) Maximum computation time (b) Number of early-out checks

Figure 3.7: In *(a)*, the maximum computation times to evaluate the clamping situation is displayed. All timings refer to the algorithm that incorporates the sorting and caching scheme (except for the fcl implementation "mesh"). In *(b)*, the number of EOC checks is itemized for depth $d_{\clubsuit} = 9$ using the GJK algorithm. The color of the bars represent the outcome of the EOC checks. For $d_{\max}$, condition 2.80 is true, for $d_{\min}$ condition 2.79. Only in the case of *m_head* with the sorting and caching extension, there are 84 $d_{\max}$ EOC, which is too little to be seen on the graph.

Since the amount of necessary EOC checks is highly dependent on the robot-environment configuration, the computation time of the clamping identification algorithm varies much during the trajectory. This can be seen by inspecting the standard deviations of the time measurements in Fig. 3.5 and Fig. 3.6. Since the sorting and caching scheme only accelerates the computation in potential clamping situations, the maximum time to rule out possible clamping can still be high. Figure 3.7a reports the maximum computation time for the simulated scenarios. The fastest of our tested methods needs maximally almost 10 ms to evaluate the clamping situation in the *m_abdominal_region* scenario. Figure 3.8 and Fig. 3.9 visualize the outcome of the clamping identification for each of the collaborative zone setups for the entire trajectory. In both figures, the outcome of the comparative method from *fcl* is displayed on top. On the bottom, the results are shown for parameter set 3 (see Tab. 3.4) of our algorithm. Here, most clamping identifications are due to the fast evaluation of the cached *dangerous leaf node pairs*. In Fig. 3.8, the danger evaluation of ours and the comparative method matches well. However, our method evaluates the possible clamping scenarios more conservatively, as shown representatively in Fig. 3.10. The possible clamping situations begin maximally 51 time steps earlier or end maximally 51 time steps later than for the comparative method. In the scenario illustrated in Fig. 3.9, the more conservative evaluation of clamping situations is well-marked for the second robot link. Between time steps 5690 and 9261, our method identifies a clamping situation,

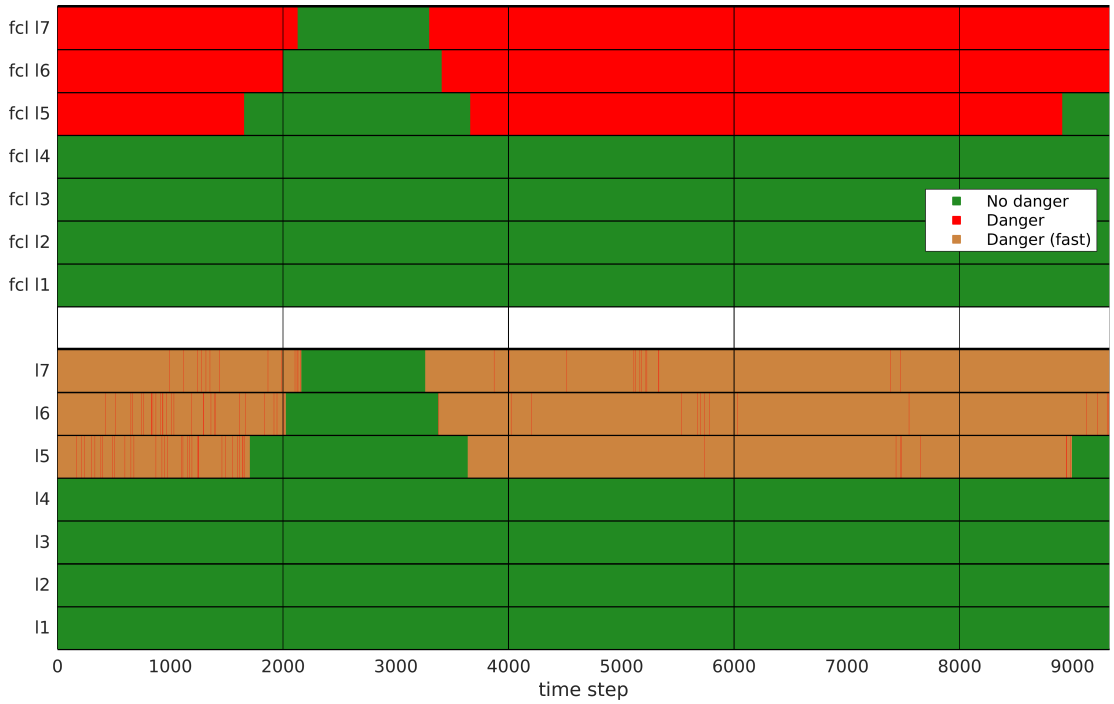


Figure 3.8: Clamping evaluation results of the scenario with *m_head* in the collaborative zone. Each horizontal bar represents the clamping evaluation for a single robot link. The 7 bars on top display the result for the comparative *fcl* method, the 7 bars below display the results of our algorithm with parameter set 3. Red and brown color indicate that a clamping situation is possible. The difference is that brown is used when the evaluation of the cached *dangerous leaf node pair* results in a possible clamping identification. The brown bars of links *l5*-*l7* are interrupted with red stripes that are best visible electronically and zoomed in.

whereas the comparative method does not. In this case, our OBB approximations of the involved geometries lead to a smaller minimum distance calculation that do not trigger the distance based EOC. A comparison of the accuracy between our OBB approximations and the simplified mesh, used for the fcl implementation, is given in Sec. 3.1.3.

Different parameter sets of our algorithm also lead to slightly different clamping identification results. For some parameter sets, a potential clamping situation begins earlier or ends later, compared to other parameter sets. Figure 3.11 illustrates the clamping identification results for a selection of parameter sets in the ending of a clamping situation. The times between the endings of the potential clamping situations differ no more than 15 time steps across the parameter sets. When two parameter sets use different depths $d_{\clubsuit}$ (e.g. set 3 and 4), these differences are bigger than when using the same depth $d_{\clubsuit}$ but different OBB to OBB distance calculations (e.g. set 3 and 5). The reason is that OBBTrees with different depths $d_{\clubsuit}$

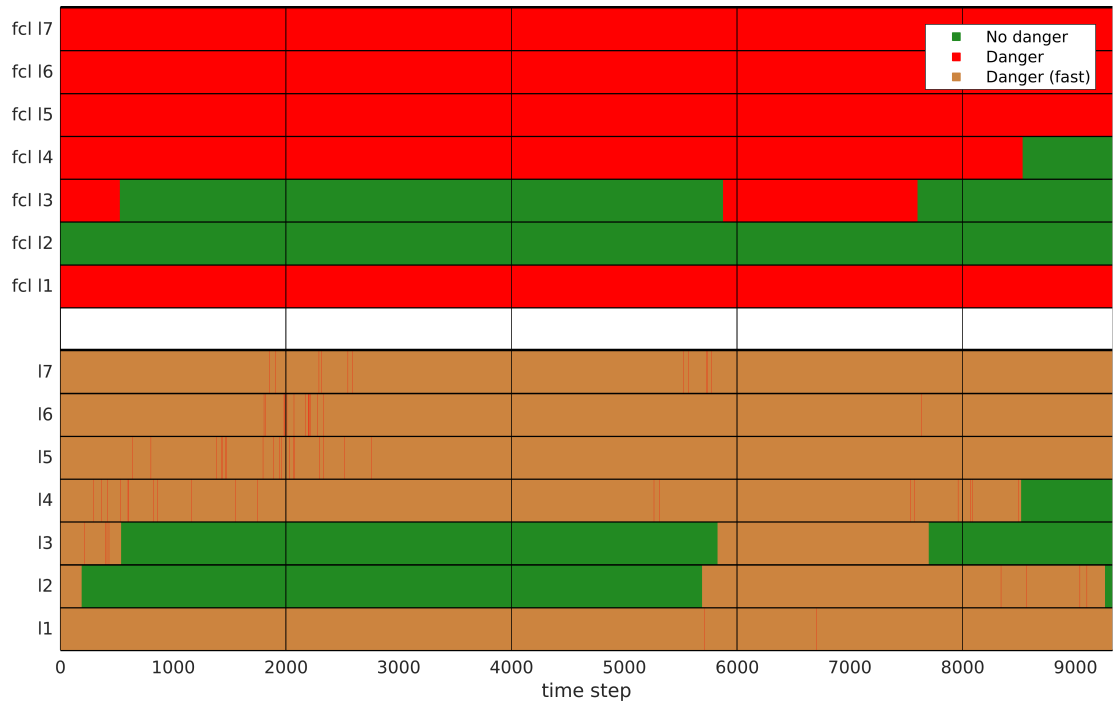


Figure 3.9: Clamping evaluation results of the scenario with *m_abdominal_region* in the collaborative zone. See Fig. 3.8 for further explanations.

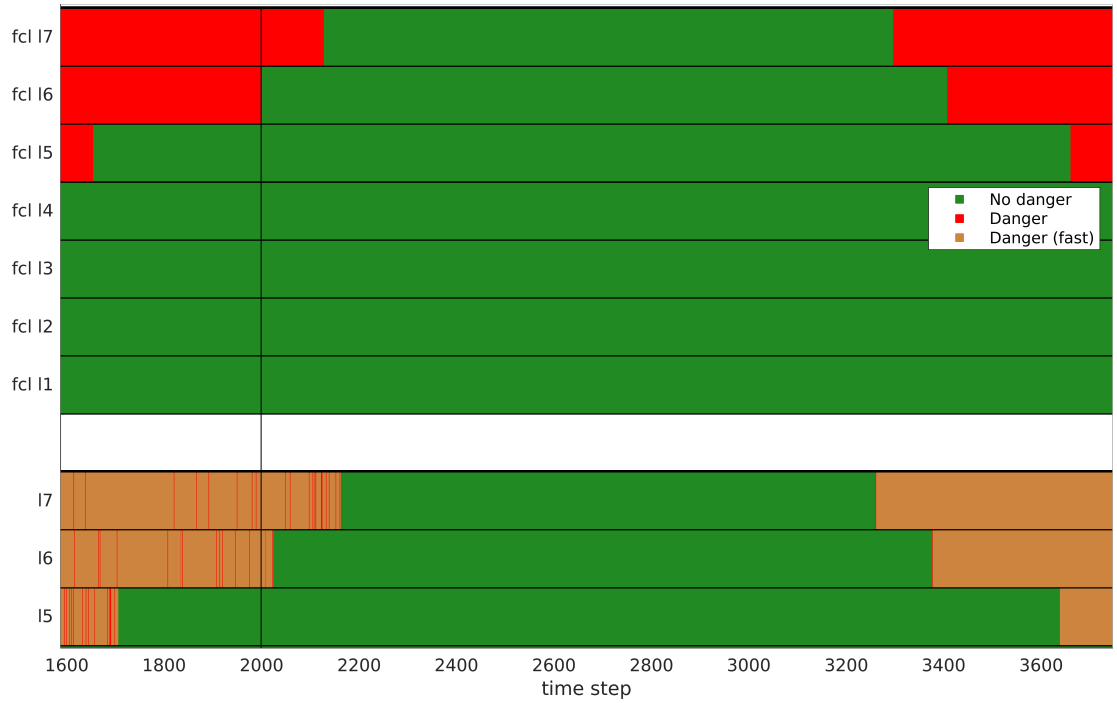


Figure 3.10: Clamping evaluation for the *m_head* scenario. Zoomed in extract of Fig. 3.8.

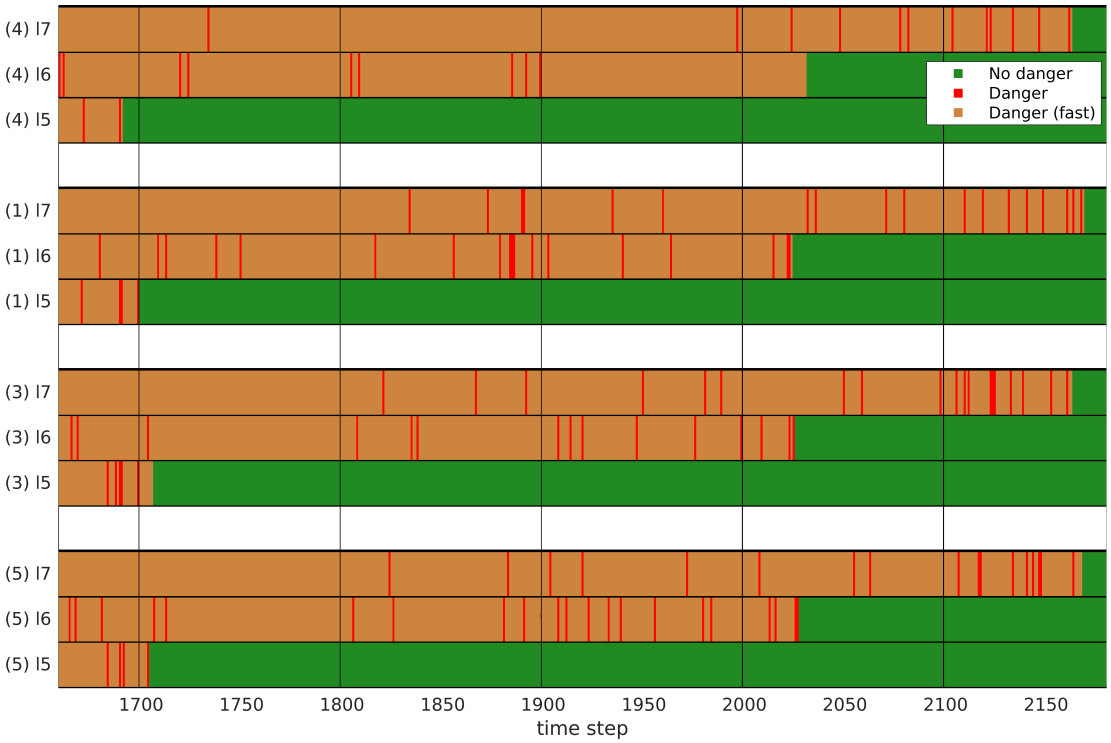


Figure 3.11: Clamping identification results for different parameter sets in the *m_head* scenario. For each parameter set, only the links 5 to 7 are displayed because the remaining links show identical results. The parameter set that is used is written to the left of the respective link name in parentheses. See Tab. 3.4 for their parameters.

are composed of completely different OBBs. Especially when regarding only the cached leaf node pairs, differences in the OBBs of these leaf nodes lead to different distance computations. As can be seen in Fig. 3.12, the OBB of a child node is not always contained in the OBB of the parent. If the clamping identification algorithm considers all OBBs along the path from the root node to the leaf, it is more likely that somewhere along the path the branch is correctly dismissed for a non-clamping situation.

Velocity Early Out Criteria

After having evaluated the parameters of the distance based EOC, we include the velocity based EOC to analyze the same trajectory of the robot again. They are checked in following order: First comes the *colliding direction* (2.83) EOC, then the *link normal* (2.84) and *environment normal* 2.85 EOC. Finally, the *distance* EOC, as detailed above, is inspected. Figure 3.13 gives an impression how the *colliding direction* EOC (2.83) affects the traversal of the OBBTrees' root nodes. The gray arrows represent the conservative displacement vectors $\boldsymbol{p_{AB}}$ from the

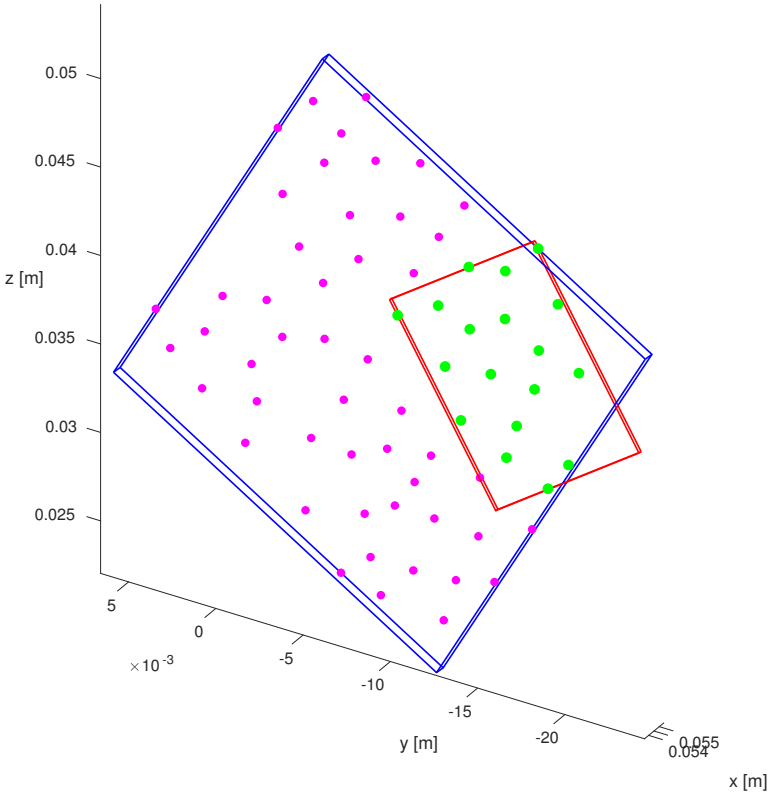


Figure 3.12: An artifact of our clustering method is that child OBBs (in red) are not always fully contained in their parent's OBB (in blue). This is because we fit the parent's OBB to the sample points (magenta and green) and not to the children's OBBs.

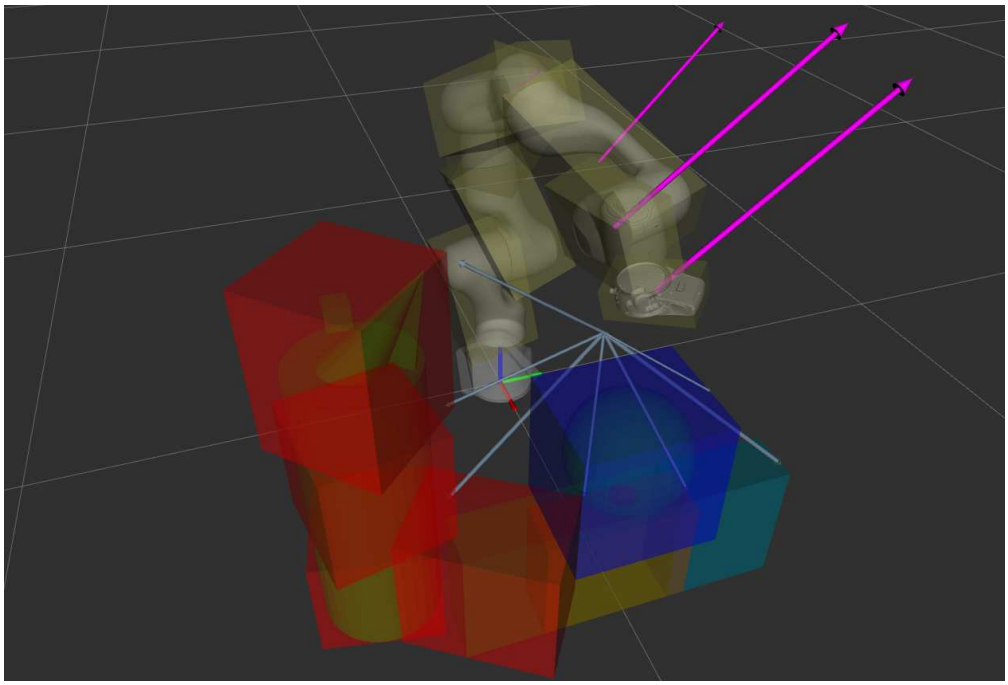


Figure 3.13: Illustration of the *colliding direction* EOC as per (2.83). The conservative displacement vectors $\boldsymbol{p_{AB}}$ from link 7 to the OBBTrees of the environment are shown in gray. When they form a negative dot product with the velocity of the respective link (pink arrows), the OBBTrees are not traversed any further (red boxes). Blue boxes are in potential clamping / colliding direction. The orange and turquoise boxes are actually red and blue respectively. The change in color is due to rendering them transparently on the green environment.

currently considered link (link 7) to the OBBs of the environment. If they project negatively on the linear velocity $\dot{\boldsymbol{x}}$ (pink arrows) of the link's center, the respective OBBTrees (red boxes) are dismissed in the further traversal.

Figure 3.14 displays the effects of the velocity based EOC on the clamping identification results for the *m_head* scenario. The bottom graph shows the results when no velocity based EOC is used. Adding the ***c**olliding direction* EOC (middle graph) and additionally the *link and environment* ***n**ormal* EOC (top graph) leads to less potential clamping cases. Whereas the identification of the beginning of potential clamping situations is governed by the distance EOC, the endings of these situations are earlier identified by including the velocity based EOC. The first potential clamping situation for the 5th link finishes for example 1193 time steps earlier when considering the velocity EOC.

The clustering approach of the OBBTrees also influences the *link and environment normal* EOC. When sample points with similar normals are grouped together

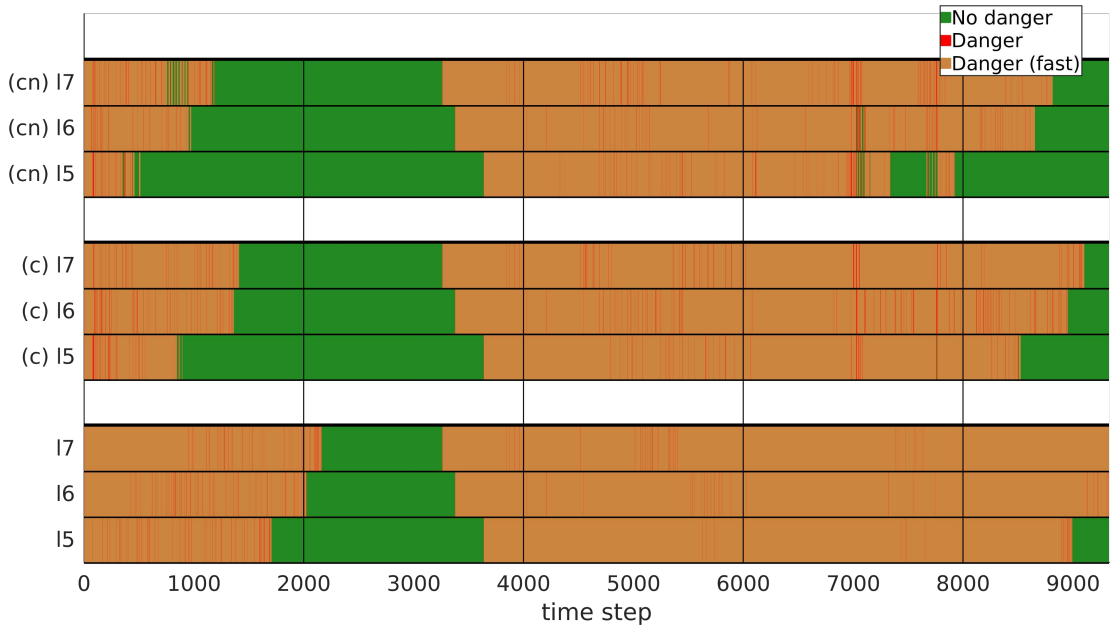


Figure 3.14: Clamping identification outcome for the *m_head* scenario. For the bottom graph, only distance EOC are used (see Fig. 3.8). The ***c**olliding direction* EOC is added for the middle graph, as well as on the top graph together with the *link and environment* ***n**ormal* EOC. Only links $5-7$ are displayed since the outcome for the remaining links is identical. Best viewed electronically and zoomed in.

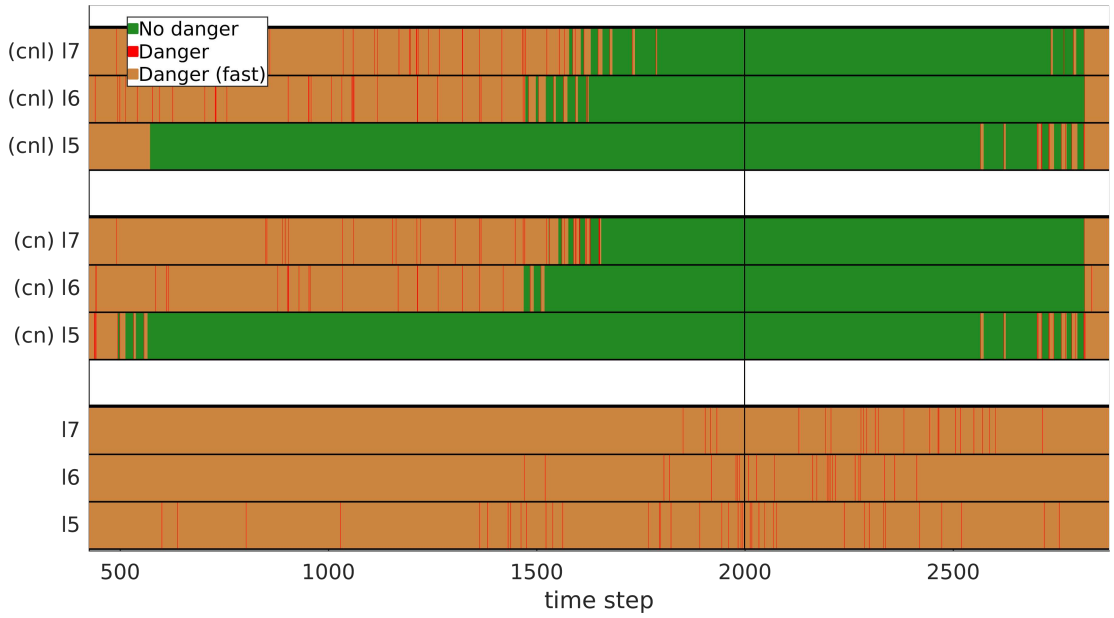
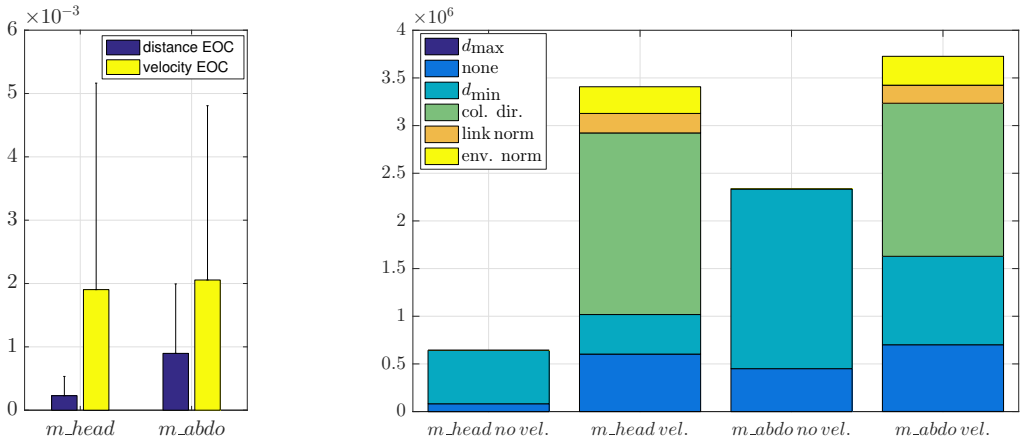


Figure 3.15: Clamping identification outcome for the *m_abdominal_region* scenario (zoomed in). For the bottom graph, only the distance EOC are considered. For both the middle and the top graph, the ***c**ollision direction* and *link and environment **n**ormal* EOC is considered as well. The middle graph shows the clamping identification result when normal information is used during the clustering process at the OBBTree creation[a]. For the top graph, on the contrary, only localization information was used during clustering. Best viewed electronically and zoomed in.

[a]To be exact, parameter set 3 from Tab. 3.4 is used.

in one OBB, they can be easier dismissed compared to OBBs containing points with different normals. This is representatively illustrated for the *m_abdominal_region* scenario in Fig. 3.15. The dangerous clamping situations are sooner over for OBBTrees with $d_{\clubsuit} = 9$ (middle graph), than for OBBTrees that do not consider normal information during clustering ($d_{\clubsuit} = 9_l$, top graph).
One artifact of employing the velocity based EOC is the induced jitter in the clamping identification signals, as can be seen in Fig. 3.14 and Fig. 3.15. This is mainly caused by the fluctuations in the links' velocities, which are calculated by numerical derivation.

More EOC checks lead in general to more branch dismissals. However, one cannot determine a priori whether the increased dismissal rate results in faster or slower execution times. On the one hand, more EOC checks can dismiss a branch earlier than it would has been with fewer checks. On the other hand, when a dangerous leaf node pair is found with few EOC checks, more checks can possibly identify such a leaf node pair as safe, making further OBBTree traversal necessary.



(a) Mean computation time and standard deviation

(b) Number of EOC checks with the respective result

Figure 3.16: Comparison between using only distance EOC and using also velocity EOC. In *(a)*, the mean computation times and standard deviations for the clamping identification algorithm are compared. *(b)* lists the outcomes of the EOC checks. The bars with "*no vel.*" use only distance EOC.

Especially with the sorting and caching scheme, the more EOC, the likelier the cached dangerous leaf node pair will get dismissed. As portrayed in Fig. 3.16a, including the velocity EOC checks slows down the clamping identification algorithm for the tested trajectory. For both the *m_head* and *m_abdominal_region* scenario, more branches have to be traversed. This can be seen in Fig. 3.16b, as there are more EOC checks that result in EOC "*none*" when considering the velocity EOC. However, rates near to 500 Hz can still be obtained. The time needed on average for a single EOC check affects the total execution time of the clamping identification algorithm. With 5.5 μs on average, the *colliding direction* EOC is the most expensive to analyze, followed by the *distance* EOC with 4.2 μs (GJK) and 2.8 μs (SAT)[4]. The *link and environment normal* EOC is the fastest with 0.28 μs.

3.1.3 Accuracy Comparison of OBBTree vs. FCL's Mesh-Based Method

As stated in Sec. 3.1.2, we use simplified meshes for the comparative distance evaluation, since otherwise the needed computation time would make the algorithm unfeasible. We measure the quality of the simplified mesh by taking the *Hausdorff Distance* between the simplified and the original mesh. To see where the simplified mesh differs much from the original one, we compute the Haus-

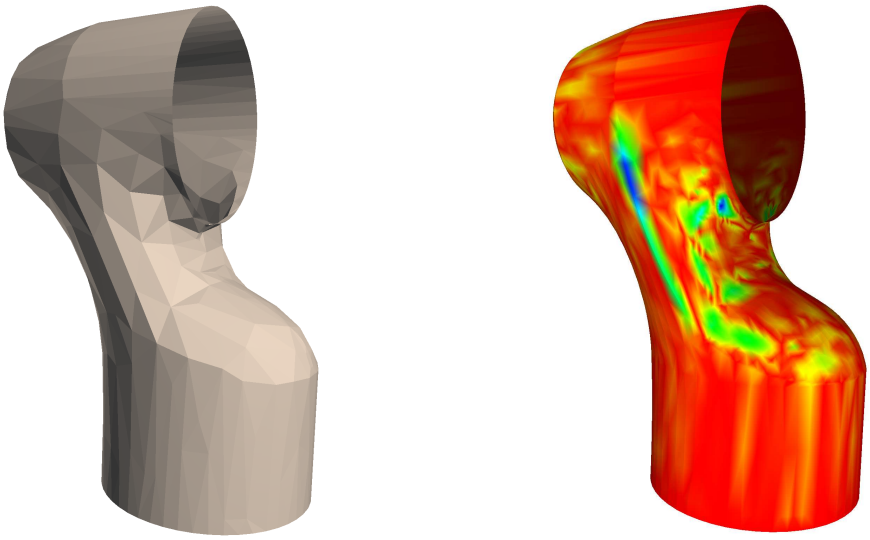
[4]The reported computation times are higher than in Tab. 3.2 because there is additional overhead like function calling. Also note that the computation times for the velocity based EOC are trajectory and scenario dependent. Values are reported for the *m_head* scenario.

dorff Distance separately for a selection of sampled points. These distances are visualized in Fig. 3.17b for the first link. Red areas indicate that the Hausdorff Distance in these areas is low, signifying a good mesh approximation. Blue areas on the other side indicate a poor mesh approximation. Comparing this error map with the simplified mesh in Fig. 3.17a, one sees that the approximation error is especially high on areas where the simplified mesh exhibit strong discontinuities. To compare this approximation error quantitatively with our OBBTree approximations, we calculate a similar error measure for the OBBTrees by taking the minimum distance from each sample point to the enclosing OBB of the respective leaf node. The mean value of these distances is reported separately for each robot link in Fig. 3.18. Since the leaf node OBBs differs depending on the clustering algorithm, the errors are separately displayed for the clustering method integrating the normals of the sample points, and for the method only considering the points' locations. Across all links and environment OBBs, the clustering method with normals has lower approximation errors than its counterpart. Especially for the environment OBBTrees, the normals of the sample points seems to help finding clusters that can better be approximated with OBBs. The mean errors from these approximations are also 3.8 to 24.9 times less than the mean approximation errors from the mesh simplification concerning the robot links. For the environment, the approximation errors of the OBBTrees fluctuate more. A box shaped object (*env1*) can be better approximated with OBBs than a sphere shaped one (*env7*). Consequently, some of the OBBTrees of the environment are better approximated with OBBs than with a simplified mesh.

Although, on average, the OBBTree approximations for the robot links are more accurate than the mesh simplifications, this is not true for the maximum approximation errors. As shown in Fig. 3.19, it varies from link to link which approximation strategy is more accurate. However, the environment mesh is best approximated by the simplified mesh. The good approximation performance of the mesh simplification for the environment is because the original mesh contains only 14 percent more faces than the simplified one. Therefore, the simplified mesh resembles strongly the original one.
The maximum approximation errors for the OBBTrees are close to the sampling radius used for the OBBTree creation (see Sec. 2.2.3). Thus, the accuracy of the OBB approximation and the accuracy of the initial sampling are compatible with each other.

3.1.4 Clamping Conscious Impedance Control

In this section we describe several simulations that analyze the impedance dynamics in (2.31) for different environment setups and different parameters. The differential equations in the simulations are simply integrated by using the *forward Euler method*. The simulation parameters that are constant across the different



(a) Simplified mesh of link 1 (b) Error map of simplification

Figure 3.17: Example of the mesh simplification for link 1. The mesh in *(a)* is reduced to 1000 faces. The Hausdorff distance between the simplified and the original map is projected onto the original mesh in *(b)*. Blue areas indicate high distances, expressing large approximation errors. Red areas indicate a small Hausdorff Distance.

simulations are summarized in Tab. 3.5. To keep the illustrations simple, the simulations are in 2-D.

Simulation 1: single line obstacle

First, we consider a scenario where a control point p is prevented from reaching its goal position $\boldsymbol{x_g}$ by a line obstacle. The resulting trajectories of the control point for different control methods is illustrated in Fig. 3.20. At $t = t_0$, the control point p with mass m starts at position $\boldsymbol{x_0}$ with velocity $\dot{\boldsymbol{x}}_\mathbf{0}$. It follows the dynamics of (2.31) with the principal impedance direction $\mathfrak{p}$ aligned with its velocity. The velocity of the control point is saturated by the velocity saturation law (2.20). When point p is in contact with the environment, the environment exerts a force $\boldsymbol{f}_{\text{ext}}$ on the control point p normal to its surface[5]. This force is dependent on the stiffness of the environment k_e, as well as on the penetration distance $d_\perp$ of p into the environment, perpendicular to the environment's surface:

$$\boldsymbol{f}_{\text{ext}} = k_e * d_\perp. \tag{3.5}$$

From the first contact onwards, the different control methods lead to different trajectories of the control point. The parameters of the implemented controllers

[5]As there is only a single control point, we use here $\boldsymbol{f}_{\text{ext}}$ instead of $\boldsymbol{f_c}$.

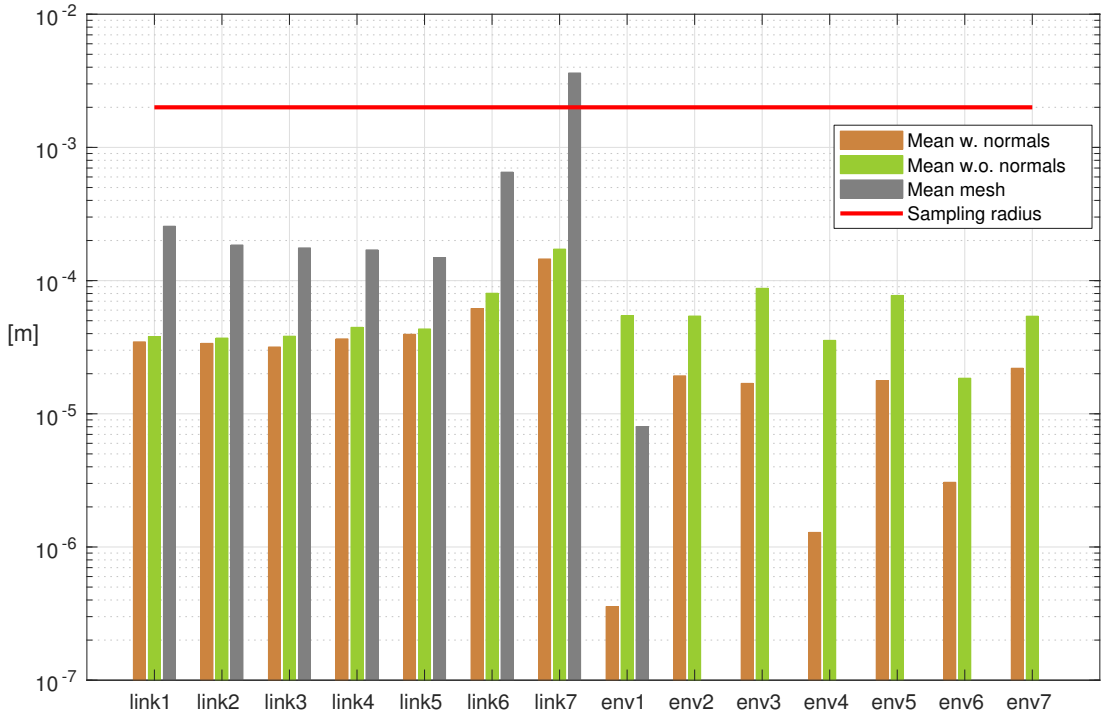


Figure 3.18: The minimal distances between sample points and their enclosing OBBs are averaged for every link and every part of the environment. The brown bars represent these distances when the OBBs were created by including the points' normals in the clustering step. The green bars show the distances when these normals were not considered during clustering. The gray bars report the Hausdorff Distance between points of the simplified mesh and the original one. There are seven green and brown bars shown for the environment since the environment mesh is split up into seven parts for which an OBBTree is created each. As a reference, the sampling radius is also shown that is used to sample the points from the original mesh to create the leaf nodes of the OBBTrees.

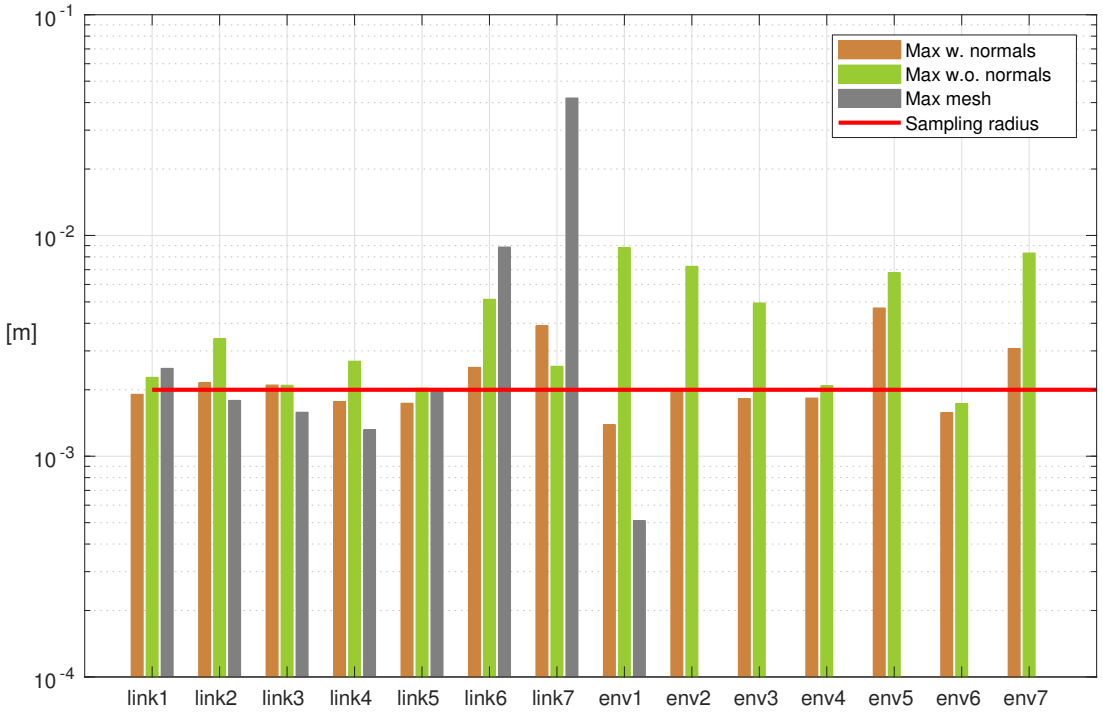


Figure 3.19: Maximum approximation errors. See Fig. 3.18 for a detailed description.

are summarized in Tab. 3.6. All but the *"vel. dir."* controller change the principal impedance direction $\mathfrak{p}$ according to the external force $\boldsymbol{f}_{\text{ext}}$. This update of the principal impedance direction $\mathfrak{p}$ can either be instantaneous, or gradually with the update law:

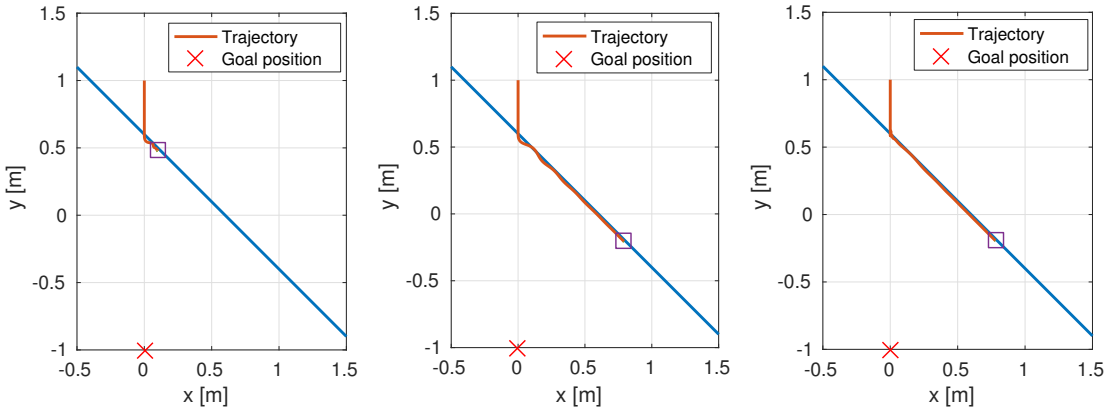
$$\mathfrak{p}_{i+1} = \frac{(1-g_u)\mathfrak{p}_i + g_u\mathfrak{p}_d}{\|(1-g_u)\mathfrak{p}_i + g_u\mathfrak{p}_d\|}, \tag{3.6}$$

where g_u is a non-negative gain and $\mathfrak{p}_d$ is the desired principal impedance direction. The update gain g_u of the exponential moving average in (3.6) is controlled via

$$g_u(t) = g_{\min} + \sigma(t - t_c)(g_{\max} - g_{\min}), \tag{3.7}$$

with $\sigma(t)$ being a sigmoid function that goes from 0.007 to 0.99 within 1 second, starting from the time of collision t_c. The update gain g_u is thus bounded by $g_{\min}$ and $g_{\max}$ and smooths high changes of $\mathfrak{p}$ due to collisions.

In the simulated scenario, the control point collides with the environment at $t = 0.47$ s. The norm of the external force $\boldsymbol{f}_{\text{ext}}$ that is experienced by the control point is reported in Fig. 3.21 for different controllers. All force trajectories surpass the maximal force $f_{\max}$ shortly after the collision. The force tracking impedance controller *"force track"* from [LB08] exhibit the least amount of force overshoot. Consequently, when controlled with this controller, the point p penetrates the



(a) velocity direction (b) force direction (c) force tracking

Figure 3.20: Simulation trajectories with one obstacle. The square represents the controlled point p that wants to reach the goal position $\boldsymbol{x_g}$ illustrated as red cross. An obstacle, which is shown as blue line, obstructs the control point from reaching its goal. The obstacle is modeled as a solid plate connected with a spring to its equilibrium position as in Fig. 2.12. In *(a)*, the principal impedance direction is left at the pre-collision velocity direction. In *(b)* and *(c)*, the principal impedance direction aligns to the external force direction. In *(c)*, the controller tries additionally to track the maximum force value $f_{\max}$, as in [LB08].

Table 3.5: Common parameters of the CCIC simulations. The mass, stiffness and damping matrices of the control point is denoted as $\boldsymbol{M}$, $\boldsymbol{K}$ and $\boldsymbol{D}$. The simulation rate is equal to the control rate, with time steps t_s.

t_s [ms]	$f_{\max}$ [N]	k_e $\left[\frac{\text{N}}{\text{m}}\right]$	$\boldsymbol{M}$ [kg]	$\boldsymbol{K}$ $\left[\frac{\text{N}}{\text{m}}\right]$	$\boldsymbol{D}$ $\left[\frac{\text{Ns}}{\text{m}}\right]$	$\boldsymbol{x_0}$ [m]	$\dot{\boldsymbol{x}}_{\boldsymbol{0}}$ $\left[\frac{\text{m}}{\text{s}}\right]$	$\boldsymbol{x_g}$ [m]
1	100	10^4	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 200 & 0 \\ 0 & 10^3 \end{bmatrix}$	$\begin{bmatrix} 89.4 & 0 \\ 0 & 200 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1 \end{bmatrix}$

Table 3.6: Control parameters of the CCIC simulation.

label	$\mathfrak{p}_d$	gradually	$g_{\min}$	$g_{\max}$	k_f	k_v
vel. dir.	$\dot{\boldsymbol{x}}_{\mathbf{0}}$	-	-	-	-	-
force dir.	$\boldsymbol{f}_{\text{ext}}$	no	-	-	-	-
c.1 force dir.	$\boldsymbol{f}_{\text{ext}}$	yes	0.001	0.005	-	-
c.2 force dir.	$\boldsymbol{f}_{\text{ext}}$	yes	0.001	0.01	-	-
c.3 force dir.	$\boldsymbol{f}_{\text{ext}}$	yes	0.001	0.025	-	-
force track [a]	$\boldsymbol{f}_{\text{ext}}$	yes	0.001	0.005	$0.1\frac{1}{\text{N}}$	$1 \times 10^{-5}\frac{s}{\text{N}}$
force track 2 [b]	$\boldsymbol{f}_{\text{ext}}$	yes	0.001	0.005	$0.01\frac{1}{\text{N}}$	$5 \times 10^{-6}\frac{s}{\text{N}}$

[a] Implements force tracking according to [LB08] in the principal impedance direction. The constant stiffness k_0 is taken to be $K_{[1,1]}$. For details, see [LB08].

[b] See footnote *a*. This controller is only used for the multiple-line obstacle simulation

environment the least, as visible in Fig. 3.20c. However, this controller lead to a steady-state force error of 4.8 N. The remaining controllers that also change the principal impedance direction $\mathfrak{p}$ to the external force $\boldsymbol{f}_{\text{ext}}$ on the other hand, reach a steady-state force equal to $f_{\max} = 100$ N, as set by the velocity saturation law (2.20). The development of their principal impedance directions $\mathfrak{p}$ is displayed in Fig. 3.22. The higher the update gain limit $g_{\max}$ of the respective controllers, the faster $\mathfrak{p}$ reaches the direction of the environment's normal vector.

Simulation 2: single and multiple line obstacle with friction and noise

To adapt the ideal simulation above more to the real world case, we introduce friction and force measurement noise. The friction is modeled by Coulomb friction of the form

$$\boldsymbol{f}_{\boldsymbol{\mu}} = \mu \frac{\boldsymbol{f}_{\text{ext}}}{\|\boldsymbol{f}_{\text{ext}}\|} \operatorname{sgn}(\dot{\boldsymbol{x}}) \boldsymbol{n}_{\parallel}, \tag{3.8}$$

with $\boldsymbol{f}_{\boldsymbol{\mu}}$ being the force due to friction, μ being the friction coefficient and $\boldsymbol{n}_{\parallel}$ is a unit vector representing the tangent of the line surface. This friction force $\boldsymbol{f}_{\boldsymbol{\mu}}$ is added to the spring force $\boldsymbol{f}_{\text{ext}}$ of the environment (3.5) and is then subject to measurement noise:

$$\hat{\boldsymbol{f}}_{\text{ext}} = (\boldsymbol{f}_{\text{ext}} + \boldsymbol{f}_{\boldsymbol{\mu}})(1 + \varepsilon). \tag{3.9}$$

Here, ε is an i.i.d. variable taken from a normal distribution $\mathcal{N}$ with mean 0 and standard deviation σ^2, imitating Gaussian white noise. In the following simulations, the environment exerts the force $(\boldsymbol{f}_{\text{ext}} + \boldsymbol{f}_{\boldsymbol{\mu}})$ on the control point when they are colliding. However, the control point senses this force as $\hat{\boldsymbol{f}}_{\text{ext}}$ with $\mu = 0.1$ and $\sigma^2 = 0.2$ due to simulated force measurement noise. Figure 3.23 reports the forces that are exchanged between p and the environment in normal direction.

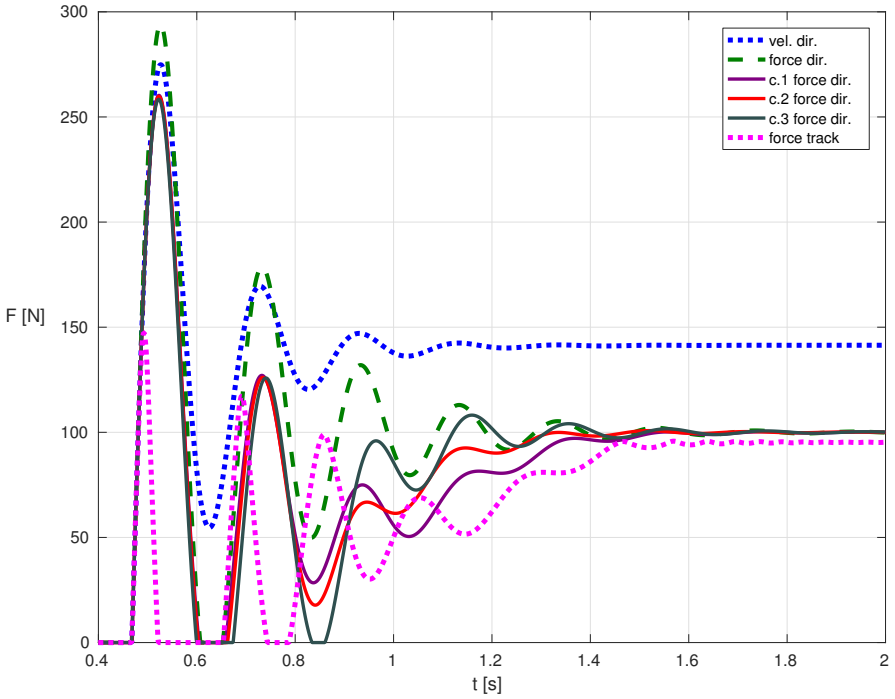


Figure 3.21: Norm of external forces $\boldsymbol{f}_{\text{ext}}$ due to contact with the single line environment.

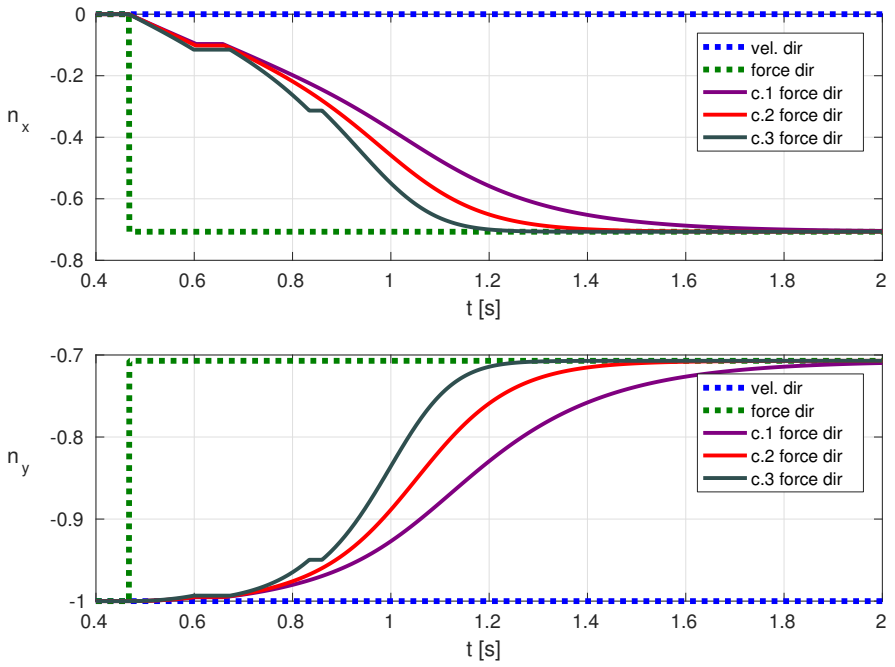


Figure 3.22: Development of the principal impedance direction $\mathfrak{p} = [n_x, n_y]^T$.

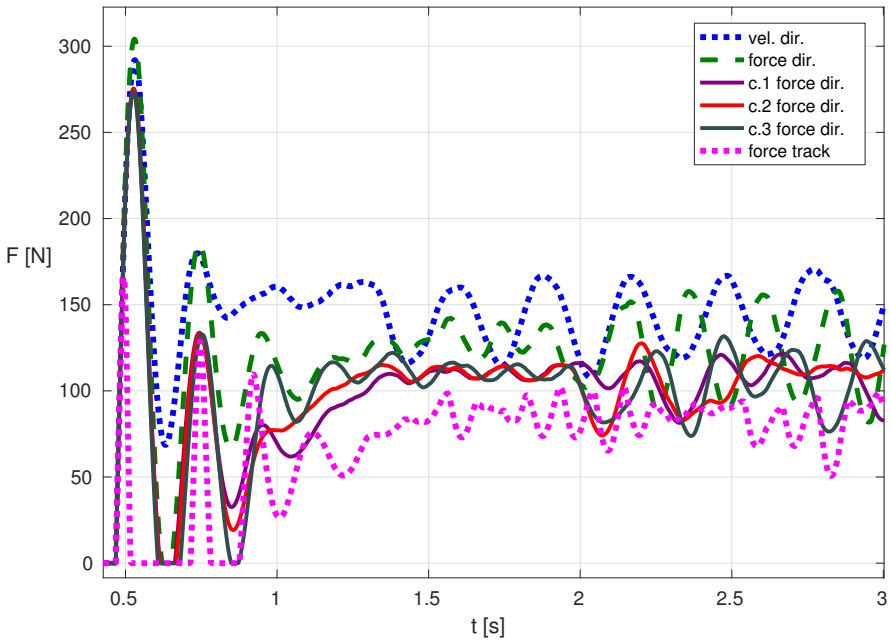
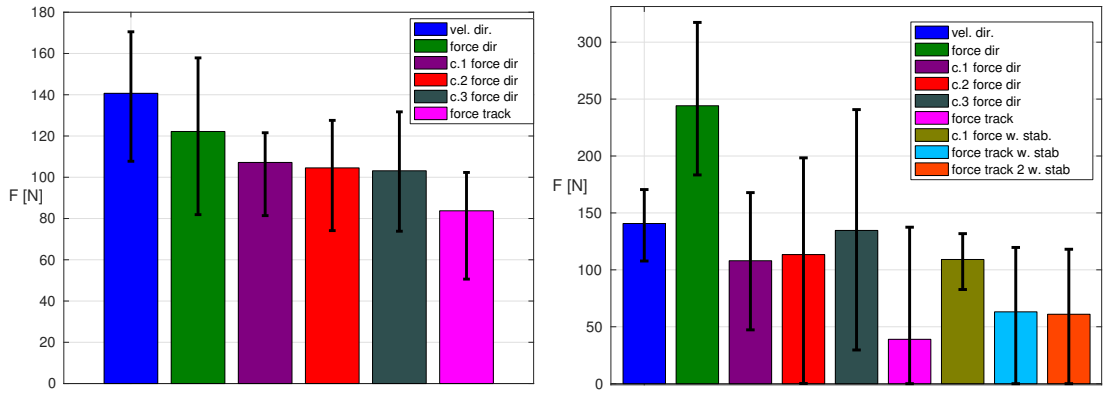


Figure 3.23: Norms of external forces $\|\boldsymbol{f}_{\text{ext}}\|$ caused by contact with the single line environment with friction and Gaussian white noise in the force measurements. Note that the graph shows $\|\boldsymbol{f}_{\text{ext}}\|$, not $\|\hat{\boldsymbol{f}}_{\text{ext}}\|$.

These forces are comparable to those of the simulation without friction and noise (Fig. 3.21) up to approximately $t = 0.8$ s. Afterwards, they do not reach a steady state, but are oscillating around their former steady state value. The controllers that gradually update the principal impedance direction $\mathfrak{p}$ lead to more irregular oscillations, but with smaller amplitude. As shown in Fig. 3.24a, the "*c.1 force dir*" controller induces the least amount of force oscillations between $t = 2$ s and $t = 3$ s.

Apart from friction and force measurement noise, the above simulations deal only with interaction forces in a constant direction. In human-robot collaboration scenarios however, interaction forces do often change their directions. When clamping a human body part, the irregular shape and flexibility of the body part may lead to changing contact force directions. To better model situations where the contact force changes its direction, a second line object is appended to the first line object in a different angle. It is placed such that the control point p is trapped at the transition between these two obstacles (see Fig. 3.25). Consequently, the contact force acting on point p changes frequently its direction from $\boldsymbol{n_b} = \frac{1}{\sqrt{2}}[1, 1]^T$ to $\boldsymbol{n_r} = [0, 1]^T$ and vice versa. As a result, the controllers that align the principal impedance direction $\mathfrak{p}$ to the measured contact force $\hat{\boldsymbol{f}}_{\text{ext}}$, change $\mathfrak{p}$ accordingly. Compared to the case with the single line obstacle, the norm of the external forces $\|\boldsymbol{f}_{\text{ext}}\|$ oscillate more, as can be seen in Fig. 3.26. Only the "*vel. dir.*" controller

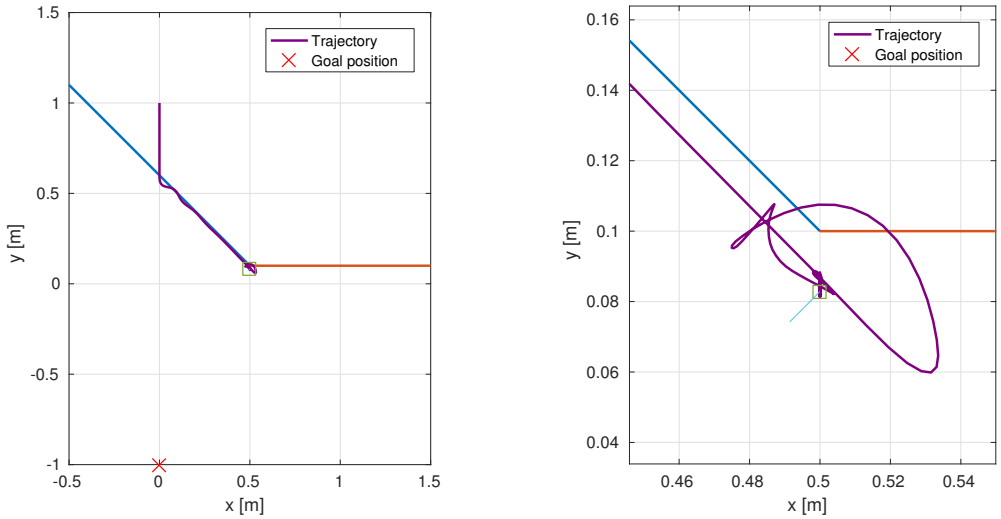


(a) single line obstacle (b) multiple-line obstacle

Figure 3.24: The colored bars report the mean force value for each controller between $t = 2$ s and $t = 3$ s. The thin black bars indicate the interval of the force values in the same time period. The bigger these bars, the bigger are the amplitudes of the force oscillations for the respective controllers.

has the same force trajectory as in Fig. 3.23, because the control point does not reach the transition point to the second line obstacle under this control. To reduce the oscillations, we pair the "*c.1. force dir.*" and the "*force track (2)*" controllers with the stability observe and control scheme as explained in Sec. 2.1.6: When the part of the position error $\boldsymbol{e_p} = \boldsymbol{x} - \boldsymbol{x_g}$ perpendicular to $\mathfrak{p}$ reaches a threshold, $\mathfrak{p}_d$ is set to $\boldsymbol{e_p}$. As illustrated in Fig. 3.27, this control modification indeed reduces the oscillations in the resulting contact force $\boldsymbol{f}_{\text{ext}}$. However, the biggest force overshoots coming from the dynamic collisions with the line obstacles, are not lowered by this modification. In the top graph of Fig. 3.27 for example, the peak at $t = 0.5$ s represents the collision with the first line obstacle and the one at $t = 1.3$ s with the second. Figure 3.24b compares the mean force values, as well as the minimum and maximum force values of the different controllers after the dynamic impacts (from $t = 2$ s to $t = 3$ s). Whereas the force track controllers with the stability observe and control scheme possess the least amount of force overshoot, their mean value is far below $f_{\max} = 100$ N, which should be tracked. This comes mainly from the still considerable amount of force oscillations, leading even to complete contact losses. Among the tested controllers, "*c.1 force w. stab*" tracks $f_{\max}$ the best with a mean force value of 109 N and ranging from 83 N to 132 N between $t = 2$ s and $t = 3$ s.

A similar performance to the "*c.1 force w. stab*" control scheme can be obtained by reducing the update gain g_u when the error $\boldsymbol{e_p}$ perpendicular to $\mathfrak{p}$ reaches a threshold. When g_u is small, the entire control scheme resembles impedance control with constant mass, damping and stiffness matrices. Figure 3.28 compares the external forces $\boldsymbol{f}_{\text{ext}}$ and the update gains g_u of this control scheme ("*c.1 force*



(a) Trajectory with second line object (b) Zoomed in trajectory

Figure 3.25: This simulation scene consists of two line obstacles. The blue line obstacle has a surface normal of $\boldsymbol{n_b} = \frac{1}{\sqrt{2}}[1, 1]^T$, the normal of the red obstacle is $\boldsymbol{n_r} = [0, 1]^T$. When the blue square p is left (right) to $x_1 = 0.5$ m, it is subject to the external force caused by the blue (red) obstacle.

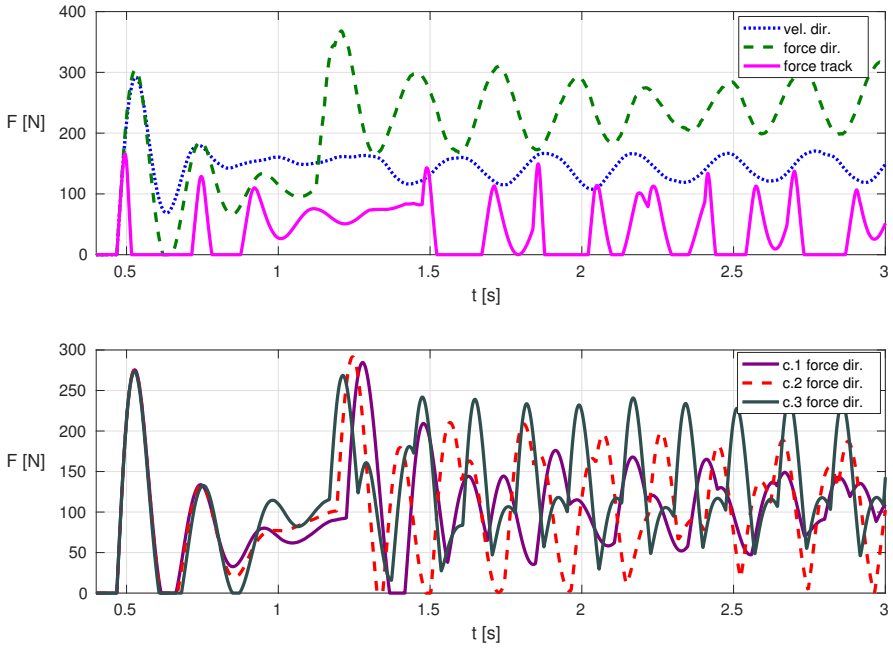


Figure 3.26: Norms of external forces $\|\boldsymbol{f}_{\text{ext}}\|$ caused by contact with the multiple-line environment. For the sake of clarity, the forces are split into two graphs.

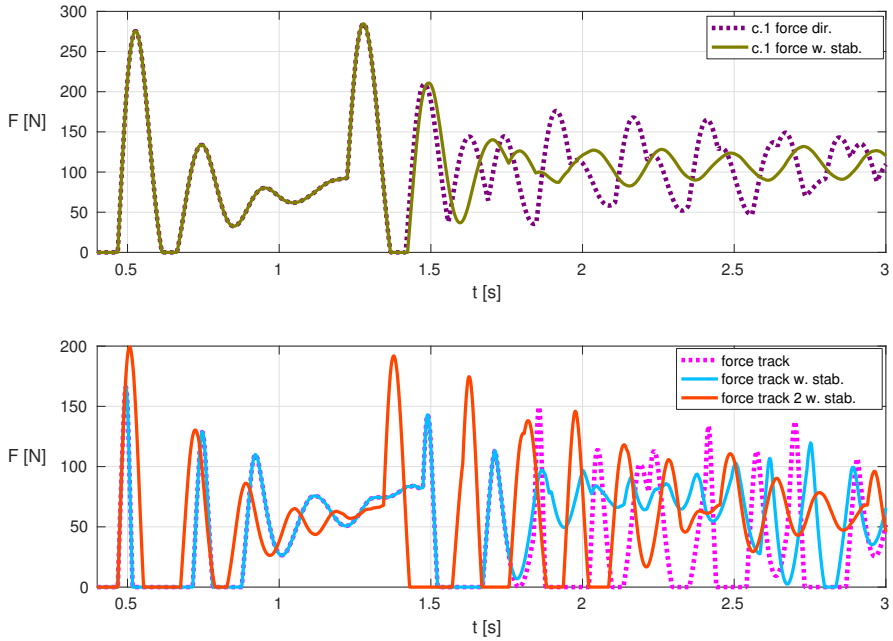


Figure 3.27: Effect of the stability control scheme on $\|\boldsymbol{f}_{\text{ext}}\|$. The force trajectories without the stability control scheme are the same as in Fig. 3.26 and serve as comparison.

w. gain c.") and of the "*c.1 force w. stab*" controller. Both controllers trigger their change in comportment at the same time ($t = 1.34$ s), using identical instability detection thresholds. Whereas the "*c.1 force w. stab*" controller changes the principal impedance direction $\mathfrak{p}$ to the position error $\boldsymbol{e_p}$ during this phase, the "*c.1 force w. gain c.*" controller leaves $\mathfrak{p}$ quasi-constant. After the oscillations in the position error $\boldsymbol{e_p}$ perpendicular to $\mathfrak{p}$ have fallen below a threshold, both control schemes update $\mathfrak{p}$ again to follow the sensed external force $\hat{\boldsymbol{f}}_{\text{ext}}$ with increasing update gains g_u. The time of changing back to this initial comportment is different for both controllers, since the evolution of $\boldsymbol{e_p}$ differs.

3.2 Experiments

The experiments are performed with the robotic platform *Panda* [EMI17] from Franka Emika. It is a 7 DoF lightweight robot arm with a payload of 3 kg, designed for human-robot collaboration. The links are elastically coupled with the motors, enabling high compliance and torque sensing. To control the robot, we use the Franka Control Interface (FCI) [EMI18a] and the respective Robotic Operation System (ROS) integration. Through the ROS Control interface, we can send torque commands to the robot at a rate of 1 kHz. The code (C++) is executed on the same machine as the simulations (see Sec. 3.1).

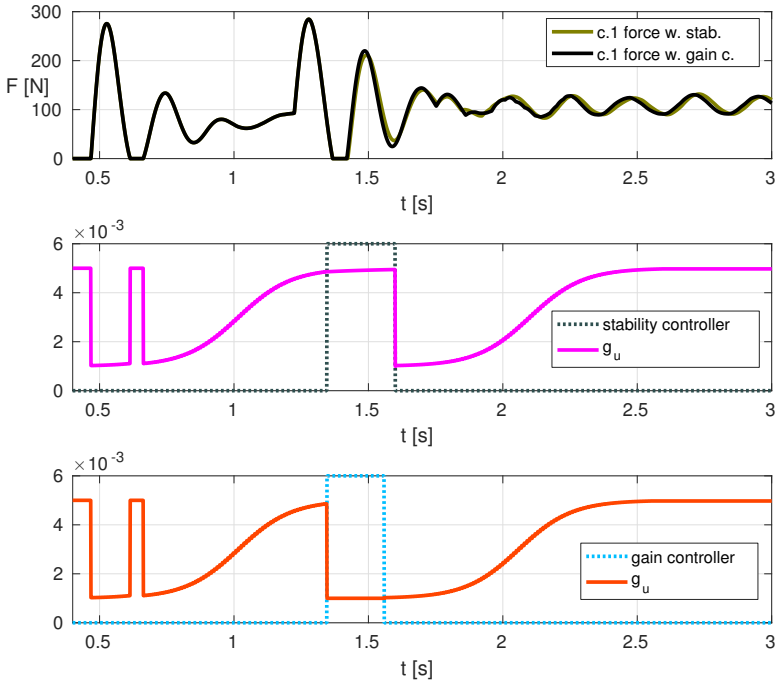


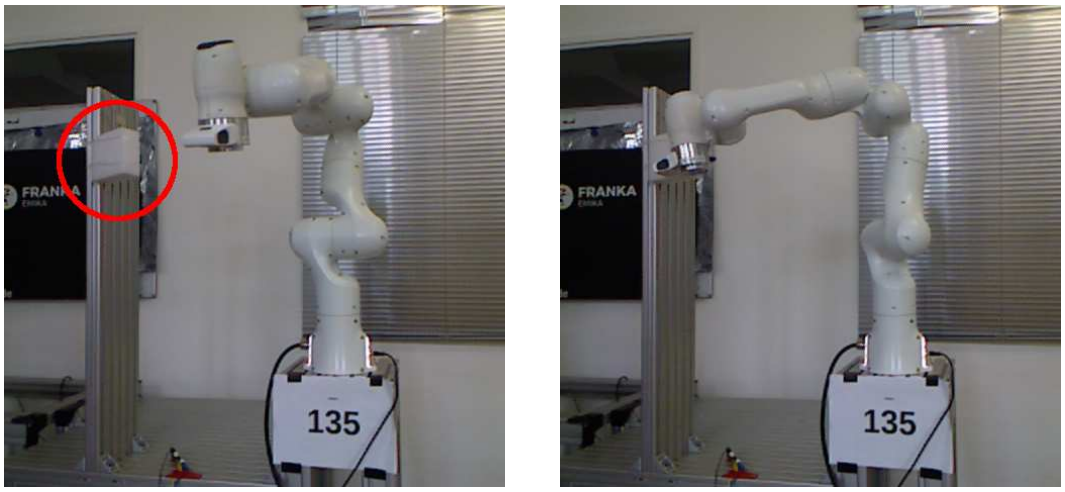
Figure 3.28: The top graph compares the norms of the external forces $\|\boldsymbol{f}_{\text{ext}}\|$ of the controllers. The middle graph shows, besides g_u, the phase (high) when the stability control (e.g. $\mathfrak{p}_d = \boldsymbol{e_p}$) is active for the "*c.1 force w. stab*" controller. Analogously, the bottom graph shows the update gain g_u and the phase (high) when "*c.1 force w. gain c.*" resembles quasi-constant impedance control (i.e. $g_u = g_{\min}$).

3.2.1 Clamping Identification

The first experiment demonstrates the performance of our clamping identification algorithm. To better analyze the accuracy of the algorithm, we use a foam block of known dimensions as a substitute of a human body part. This foam block is attached to a wall-like environment whose pose is known to the robot. The wall is modeled as a single cuboid, from which 60893 points are sampled for the OBBTree creation. Figure 3.38 illustrate this cuboid and the sample points. Obviously, a simple body like a cuboid can be represented much easier than with our OBBTree structures. However, to be able to transfer the results of the experiments to more complicated environment geometries, the simple cuboid body is treated identically as more complex bodies. Concerning the foam block, the robot does not know its pose, only its minimal and maximal dimensions $\text{size}_{\min}$ and $\text{size}_{\max}$. The setup is shown in Fig. 3.29. To guarantee a control rate of 1 kHz, we execute robot control and clamping identification on two separate threads. For this experiment, the robot is commanded a trajectory such that its end-effector[6] crashes into the foam block from the side (negative x-direction of the world coordinate system as displayed in Fig. 3.38a). The clamping identification algorithm ($d_{\clubsuit} = 9$, using SAT) surveils the robot's motion simultaneously, however, as detailed in Sec. 3.1.2, with a potentially slower rate. The results of the clamping identification algorithm is displayed in Fig. 3.30. At $t = 7.4$ s, a potential clamping situation is identified for the end-effector. As a result, the velocity of the end-effector is reduced such that potential collisions are less harmful. At $t = 7.56$ s, the collision with the foam block is sensed, triggered by $\|\boldsymbol{f_c}\|$ passing a threshold. Inferring from the course of $\|\boldsymbol{f_c}\|$, the actual collision begins approximately at $t = 7.54$ s. In the time between having identified a potential clamping situation ($t = 7.4$ s) and the collision itself, the end-effector traveled 4.4 cm further, demonstrating the conservative nature of the clamping identification algorithm. As the algorithm is unaware of the foam block's pose, it uses worst-case assumptions.
As soon as a collision is detected, the robot is put into zero gravity mode and is thus retracting from the collision due to the contact force $\boldsymbol{f_c}$. This is the reason why the clamping identification signal falls off again at $t = 7.66$ s: The velocity based EOC are activated. Slight movements and noise in the velocity signal $\dot{\boldsymbol{x}}$ of the contact point lead to chattering of the clamping identification signal at $t = 8.22$ s. Especially since $\dot{\boldsymbol{x}}$ is near zero, the velocity direction changes strongly with changing $\dot{\boldsymbol{x}}$. This chattering can be reduced by labeling situations with velocities below a certain threshold as not dangerous.

In a second experiment, we place the foam block orthogonally to the wall environment (see Fig. 3.31a). The robot's end-effector is commanded to hit the foam block in a straight line (Fig. 3.31b). For this experiment, the clamping identification algorithm uses identical OBBTrees as in the previous experiment, but the

[6] As no gripper is mounted in this setup, link 7 is the end-effector.



(a) No clamping situation (b) Clamping situation

Figure 3.29: A white block of foam ($14.1 \times 11.3 \times 4.8$ cm) is attached to a wall-like obstacle (red circle). It is placed such that one of its sides touches the edge of the wall-like obstacle, resulting in a slight diagonal pose of the foam block (relative to the world coordinate system as shown Fig. 3.38a). The minimal and maximal dimensions size_{min} and size_{max} of the foam block are registered in a collaborative zone, covering the entire robot's workspace.

GJK method instead of SAT. Since the collision scenario is created such that the worst case pose of the foam block is its actual pose[7], we expect the clamping identification signal to trigger more closely to the actual time of contact than in the previous experiment. Figure 3.32 shows the respective signals for this setup. A possible clamping situation is indicated at $t = 2.59$ s, whereas the actual contact occurs 40 ms later at $t = 2.63$ s. During this time, the end-effector traveled 5 mm further, revealing that in this case our clamping identification algorithm is accurate to 5 mm[8].

3.2.2 Contact Force in Quasi Steady-State Conditions

As the main goal of the CCIC (Sec. 2.1.6) is to limit the contact forces in clamping situations, we evaluate in the next experiment if the CCIC reaches this goal in real life conditions. Being mostly interested in the quasi steady-state contact forces, we use the modified control law (2.74)-(2.76). The values of the

[7]For this experiment we only consider poses of the foam block where an entire side is in contact with the wall environment. Thus, the largest relevant dimension of the foam block is the length of its biggest side.

[8]This accuracy evaluation is also affected by the time delay of the $\boldsymbol{\tau}_{\text{ext}}$ estimation. As $\boldsymbol{\tau}_{\text{ext}}$ is filtered by the robot control system, it is slightly delayed. On top of that, the end-effector has to penetrate the foam block to generate $\boldsymbol{\tau}_{\text{ext}}$.

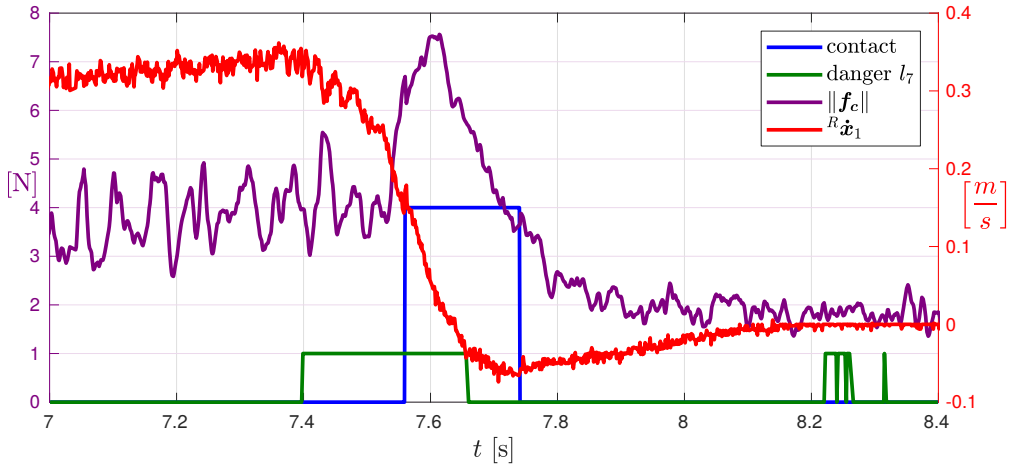
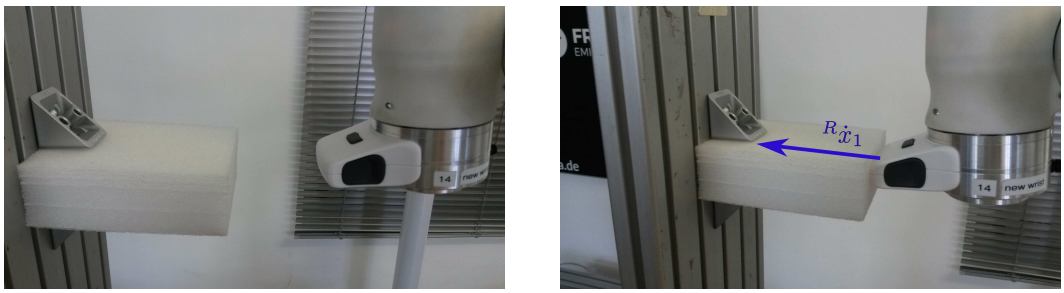


Figure 3.30: Timing of clamping identification signals. Possible clamping situations on the end-effector are indicated by high phases of *danger* l_7. The *contact* signal is triggered by the contact force $\boldsymbol{f_c}$ reaching a threshold of 6 N. As $\boldsymbol{f_c}$ is dependent on the contact point $\boldsymbol{p_c}$, the OBBTree of the end-effector is traversed to find the closest OBB's center point to the wall obstacle. For simple colliding geometries, this gives accurate enough results. For the sake of clarity, we only show one component of the contact point's velocity. However, since ${}^R\dot{x}_1$ is expressed in the frame aligned with the initial position error, it contains the most important information (see also Fig. 3.31b). The scale for the velocity ${}^R\dot{x}_1$ is on the right, for the force $\boldsymbol{f_c}$ on the left.



(a) Initial end-effector pose (b) End-effector colliding with foam block

Figure 3.31: The end-effector starts from its initial pose in *(a)* to collide with the foam block in *(b)*. The velocity of this straight movement is ${}^R\dot{x}_1$.

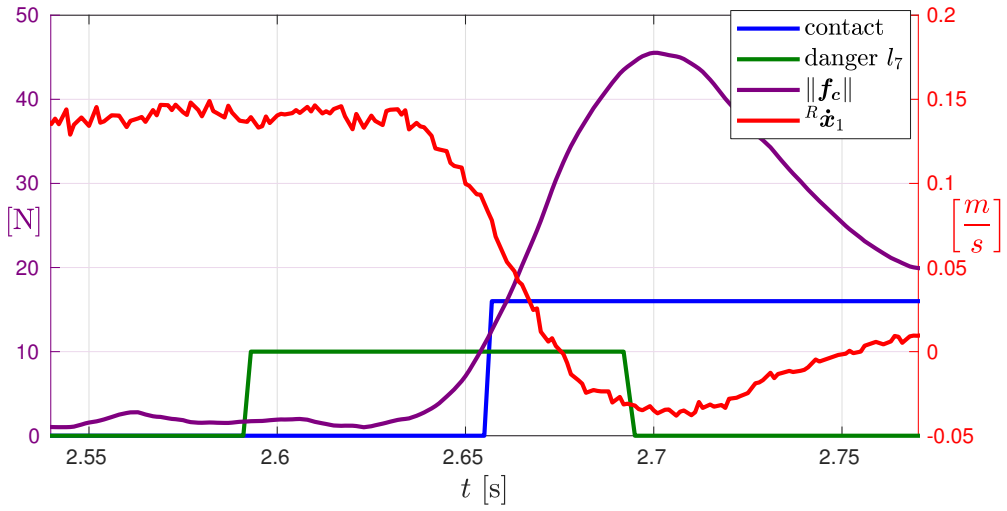


Figure 3.32: Clamping identification signals of 2nd foam block experiment. For a description of the signals, see Fig. 3.30. In contrast to the first experiment, the robot is not put into zero gravity mode, leading to higher contact forces $\boldsymbol{f_c}$.

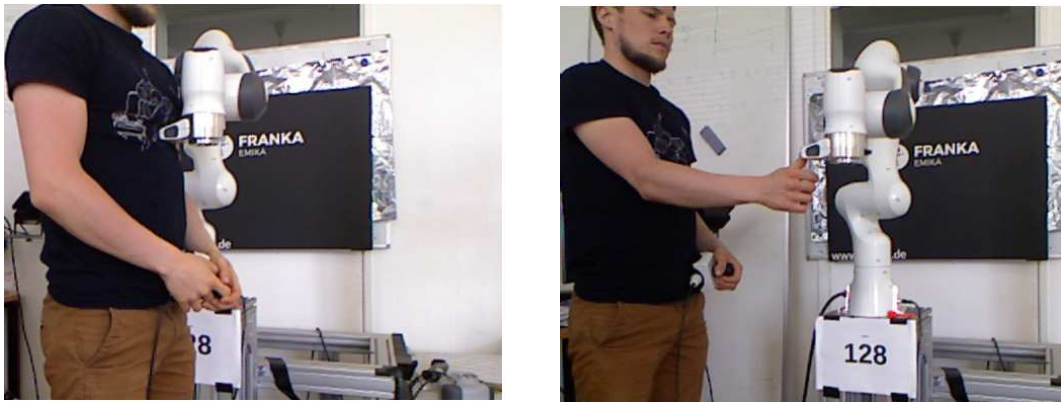
Table 3.7: Parameters of the CCIC control law.

$\boldsymbol{\Lambda_d}$ [kg]	$\boldsymbol{K}$ $\left[\frac{\text{N}}{\text{m}}\right]$	$\boldsymbol{D}$ $\left[\frac{\text{Ns}}{\text{m}}\right]$	$K_\varnothing$ $\left[\frac{\text{N}}{\text{rad}}\right]$	$D_\varnothing$ $\left[\frac{\text{Ns}}{\text{m}}\right]$
$\text{diag}\begin{pmatrix}10,10,10\\10,10,10\end{pmatrix}$	$\text{diag}\begin{pmatrix}600,1500,1500\\2000,2000,2000\end{pmatrix}$	$\text{diag}\begin{pmatrix}155,245,245\\198,198,198\end{pmatrix}$	5	4.47

matrices in (2.75)/(2.71) and scalars in (2.76) that are used in this experiment are summarized in Tab. 3.7.

The general setup of the experiment is as follows. The end-effector of the robot is commanded to alternate between two desired poses $\boldsymbol{x_d}$. Before the second pose is reached, a human subject positions itself in the end-effector's trajectory as shown in Fig. 3.33. The contact forces are then measured and analyzed.

During non-contact phases, the rotation matrix $\boldsymbol{R}$ in the control law (2.75) is chosen such that the principal impedance direction $\mathfrak{p}$ is pointing in initial error direction $\boldsymbol{e_p} = \boldsymbol{x_p} - \boldsymbol{x_{[d,p]}}$. As reported in Tab. 3.7, the compliance in $\mathfrak{p}$ direction is higher than in its perpendicular directions. Since there is no contact point, the pose of the end-effector is controlled and the contact Jacobian is taken to be the Jacobian of the end-effector $\boldsymbol{J_{EE}}$. In contact phases, $\mathfrak{p}$ is updated to follow a low pass filtered version of $\boldsymbol{f_c}$ as per (3.6) with $g_u = 0.001$. To limit the velocity and more importantly the contact forces, we apply the velocity saturation law (2.72) in principal impedance direction with $f_{\max} = 50$ N.



(a) Chest impact (b) Thumb impact

Figure 3.33: Setup for contact force experiment. In *(a)*, the robot's end-effector collides with the human chest. In *(b)*, the human stops the robot by pressing against the end-effector with his thumb. The contact point $\boldsymbol{p_c}$ for the contact force estimation (2.78) is taken to be the point laying furthest to the front in velocity direction[a].

[a]As the movement of the end-effector is predominantly in negative y-direction (of the world coordinate system), the OBBTree of the end-effector is traversed to find the OBBNode center with the smallest y-coordinate.

As shown in Fig. 3.34, the contact force $\boldsymbol{f_c}$ stays far below the limit of 50 N. For the chest impact, the peak contact force due to transient oscillations is 17.8 N. After 0.6 s, $\boldsymbol{f_c}$ stays in the range between 7.9 and 9.6 N until the end of the contact phase. The middle plot of Fig. 3.34 illustrates the principal impedance direction $\mathfrak{p}$ during the contact. The contact force $\boldsymbol{f_c}$ is mostly acting in negative y-direction of the world coordinate system, expressed by the low-passed filtered force component $\bar{f}_{c,2}$. Consequently, $\mathfrak{p}$ points mainly in the same direction ($\mathfrak{p}_2 \approx -1$). As the contact force aligns well with the initial position error $\boldsymbol{e_p}$ direction, $\mathfrak{p}$ changes only slightly during the contact.
Similar results are obtained for the thumb contact case. Here, the transient force oscillations at the beginning of the contact are weaker since the arm of the human subject dampens the collision more compared to the chest impact. When the human subject presses against the end-effector such that the robot is pushed back, the contact force $\boldsymbol{f_c}$ surpasses its former steady-state value. In this phase, peak contact force values of 27 N are obtained. This increased contact force is mainly caused by the damping term $\boldsymbol{DR\dot{x}}$ of the dynamics (2.32)/(2.77).

The discrepancy of more than 80 % between the desired steady state force and the one obtained in the experiment shows a bad performance of our control law (2.74)-(2.76). There are multiple possible reasons why our control law does not impose the desired dynamics in (2.77). Firstly, there are inaccuracies in the

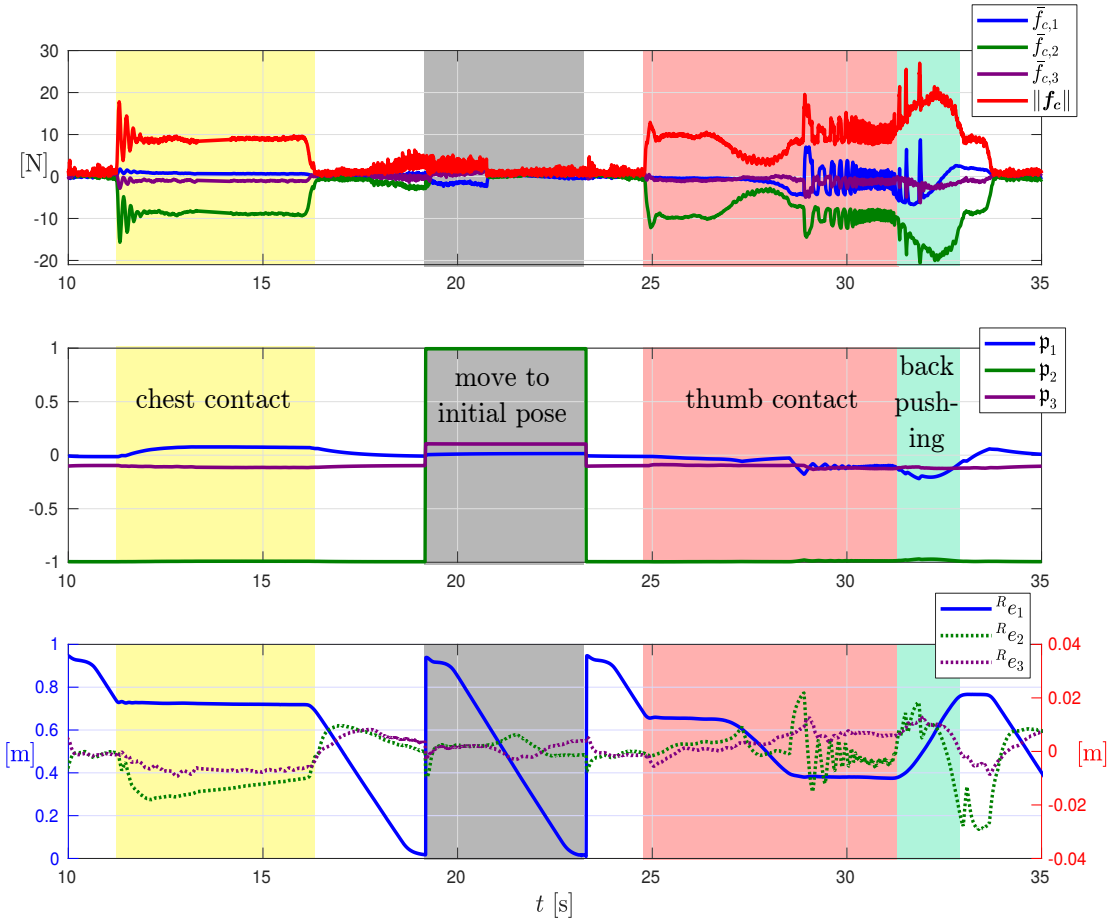


Figure 3.34: Results of the contact force experiment using CCIC. The top graph displays the components of the low-pass filtered version of $\boldsymbol{f_c}$, as well as the norm of $\boldsymbol{f_c}$ itself. The middle graph shows the components of the principal impedance direction $\mathfrak{p}$ that follow $\bar{\boldsymbol{f}}_{\boldsymbol{c}}$ when there is a contact. The bottom graph shows the positional error expressed in the rotated coordinate frame (the x-axis is aligned with $\mathfrak{p}$). The scale for the first component $^R e_1$ is shown in blue on the left, for the second and third component in red on the right. Combining $^R e_1$ with $K_{[1,1]}$ from Tab. 3.7, we see that the spring force $K_{[1,1]}\,{}^R e_1$ is thresholded in all contact scenarios by the desired force limit of 50 N. The positional errors perpendicular to $\mathfrak{p}$ are rising when there is a change in the contact situation, leading to perpendicular motions.

modeled robot dynamics, e.g. we do not model joint friction. Secondly, our control law requires exact measurements of the contact force $\boldsymbol{f_c}$ and external torque $\boldsymbol{\tau}_{\text{ext}}$. The estimation of $\boldsymbol{\tau}_{\text{ext}}$ with the generalized momentum method might not be accurate enough for our control law. Additionally, as the contact point $\boldsymbol{p_c}$ is also only estimated, the contact force $\boldsymbol{f_c}$ is subject to further inaccuracies.

For comparison, we implement another adaption of the Cartesian Impedance Controller described in [ASOFH03] that does not decouple the impedance dynamics. With

$$\begin{aligned} \boldsymbol{\tau_d} &= \boldsymbol{J_c^T f_\tau} + \boldsymbol{C\dot{q}} + \boldsymbol{g} + \boldsymbol{f_\varnothing}, \\ \boldsymbol{f_\tau} &= -\boldsymbol{R^T K R e} - \boldsymbol{R^T D R \dot{x}} \end{aligned} \tag{3.10}$$

the following closed loop dynamics are obtained in quasi stead-state conditions[9]:

$$\boldsymbol{R\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{DR\dot{x}} + \boldsymbol{KRe} = \boldsymbol{Rf_c} \tag{3.11}$$

$$\Leftrightarrow \boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{R^T DR\dot{x}} + \boldsymbol{R^T KRe} = \boldsymbol{f_c}. \tag{3.12}$$

$$\Leftrightarrow \boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{D_R\dot{x}} + \boldsymbol{K_R e} = \boldsymbol{f_c}. \tag{3.13}$$

Note that, compared to (2.32), the rotation matrix $\boldsymbol{R}$ is not applied directly to the acceleration $\ddot{\boldsymbol{x}}$, but to the product $\boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}}$. It is consequently not possible to regard (3.11) as a standard impedance dynamics, expressed in a different coordinate system. However, referring to (3.12) and (3.13), it is still possible to shape the damping and stiffness matrices $\boldsymbol{D_R} = \boldsymbol{R^T DR}$ and $\boldsymbol{K_R} = \boldsymbol{R^T KR}$ such that they appear diagonal in a desired coordinate system[10]. However, with this control law, it is not possible to shape the desired inertia of the closed loop dynamics. Hence, to distinguish the control law (3.10) from (2.74)-(2.76), we refer to the former as *comparative* control law and to the latter as *inertia shaping* control law.
As for the inertia shaping control law, the contact forces $\boldsymbol{f_c}$ must be limited in static clamping conditions for the comparative control law, too. Therefore, we have to ensure that

$$\|\boldsymbol{K_R e}\| \leq f_{\max}. \tag{3.14}$$

This can be done by using (2.29) and (2.30) in the 2-dimensional case. For the contact cases considered in this thesis ($\boldsymbol{f_c} = [f_{c,[1]}, f_{c,[2]}, f_{c,[3]}, 0, 0, 0]^T$), this can be achieved with

$$|f_i| \leq \frac{|f_{c,[i]}|}{\|\boldsymbol{f_c}\|} f_{\max}\ , \text{ for } 1 \leq i \leq 3 \tag{3.15}$$

where

$$f_i = (\boldsymbol{K_R e})_i\ , \text{ for } 1 \leq i \leq 3. \tag{3.16}$$

Here, $\frac{|f_{c,[i]}|}{\|\boldsymbol{f_c}\|}$ represents the fraction of $\boldsymbol{f_c}$ that f_i is responsible for.

[9] $\dot{\boldsymbol{\Lambda}}(\boldsymbol{q})$ and $\boldsymbol{\dot{J}_c\dot{q}}$ are assumed to be zero.

[10] The error $\boldsymbol{e}$ is rotated with $\boldsymbol{R}$ into the desired coordinate system. Then, the diagonal matrix $\boldsymbol{K}$ is applied and the result is transformed back to the original coordinate system by $\boldsymbol{R^T}$. This is analogously valid for $\boldsymbol{R^T DR\dot{x}}$.

Table 3.8: Parameters of the control law (3.10).

$\boldsymbol{K}$ $\left[\frac{\mathrm{N}}{\mathrm{m}}\right]$	$\boldsymbol{D}$ $\left[\frac{\mathrm{Ns}}{\mathrm{m}}\right]$	$K_\varnothing$ $\left[\frac{\mathrm{N}}{\mathrm{rad}}\right]$	$D_\varnothing$ $\left[\frac{\mathrm{Ns}}{\mathrm{m}}\right]$
$\mathrm{diag}\begin{pmatrix}400, 1000, 1000\\ 75, 75, 75\end{pmatrix}$	$\mathrm{diag}\begin{pmatrix}40, 63, 63\\ 12, 12, 12\end{pmatrix}$	5	4.47

We incorporate the requirement (3.14) by setting up a velocity saturation law comparable to (2.72) for every coordinate direction:

$$\begin{aligned} f_{\tau,[i]} &= D_{R,[i,i]}(\dot{x}_i - \nu_i \dot{x}_{d,[i]}) \ , \text{ for } 1 \leq i \leq 3 \\ \dot{\boldsymbol{x}}_{\boldsymbol{d}} &= \boldsymbol{D}_{\boldsymbol{R},\boldsymbol{p}}^{-1} \boldsymbol{K}_{\boldsymbol{R},\boldsymbol{p}}(-\boldsymbol{e}_{\boldsymbol{p}}) \\ \nu_i &= min\left(1, \frac{v_{\max,[i]}}{\|\dot{\boldsymbol{x}}_{\boldsymbol{d}}\|}\right) \ , \text{ for } 1 \leq i \leq 3 \\ \boldsymbol{v}_{\max} &= \boldsymbol{D}_{\boldsymbol{R}}^{-\mathbf{1}} \boldsymbol{f}_{\lim} \\ \boldsymbol{f}_{\lim} &= \frac{f_{\max}}{\|\boldsymbol{f}_{\boldsymbol{c}}\|} \left[|f_{c,[1]}|, |f_{c,[2]}|, |f_{c,[3]}|\right]^T \end{aligned} \tag{3.17}$$

The control forces $f_{\tau,[4]} - f_{\tau,[6]}$ for the orientation are not affected by this velocity saturation.

Employing the comparative control law (3.10) with the velocity saturation (3.17), we repeat the above experiment with the parameters listed in Tab. 3.8. In contrast to the previous experiment, the rotation matrix $\boldsymbol{R}$ stays constant for each target pose $\boldsymbol{x}_{\boldsymbol{d}}$ and is chosen such that $\boldsymbol{K}_{\boldsymbol{R}}$ and $\boldsymbol{D}_{\boldsymbol{R}}$ appear diagonal in the coordinate system aligned to the initial position error direction. As can be seen in Fig. 3.35, the desired force of 50 N is better tracked for both the chest and the thumb contact. Between $t = 15$ s and $t = 18$ s for example, the mean estimated contact force is $\|\boldsymbol{f}_{\boldsymbol{c}}\| = 46.8$ N. Although this force error of approximately 6% is much in terms of force tracking control, it is acceptable in our case where we are mostly interested in limiting the external force. Especially since the force limit of 50 N is only exceeded when pushing the robot back, which violates the steady-state assumptions.

3.2.3 Hand and Chest Clamping Experiment

The next two experiments compare the two control laws (2.74)-(2.76) and (3.10) in real clamping scenarios. In the first experiment, the hand of a human subject is clamped by the robot (Fig. 3.36a). To this end, the robot is commanded a trajectory moving directly towards the wall environment. Depending on where on this trajectory the contact happens, it is either a clamping situation (Fig. 3.36a), or a non clamping contact (Fig. 3.36b). To differentiate between these scenarios,

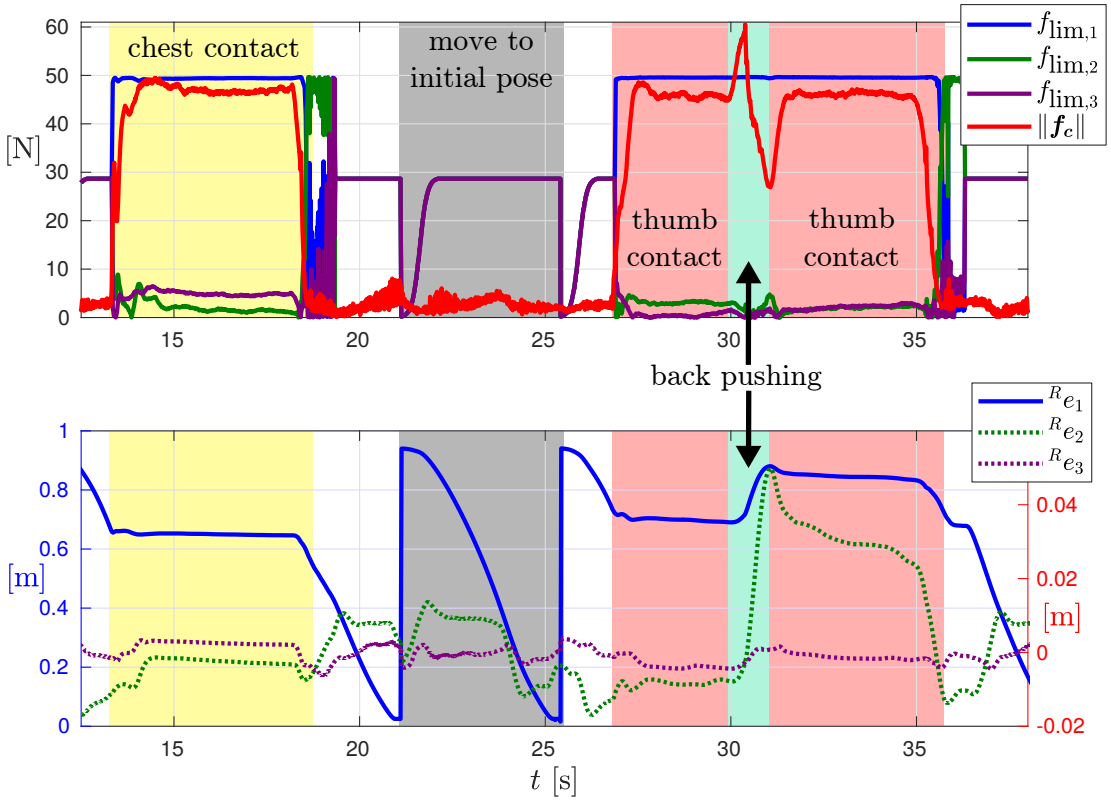
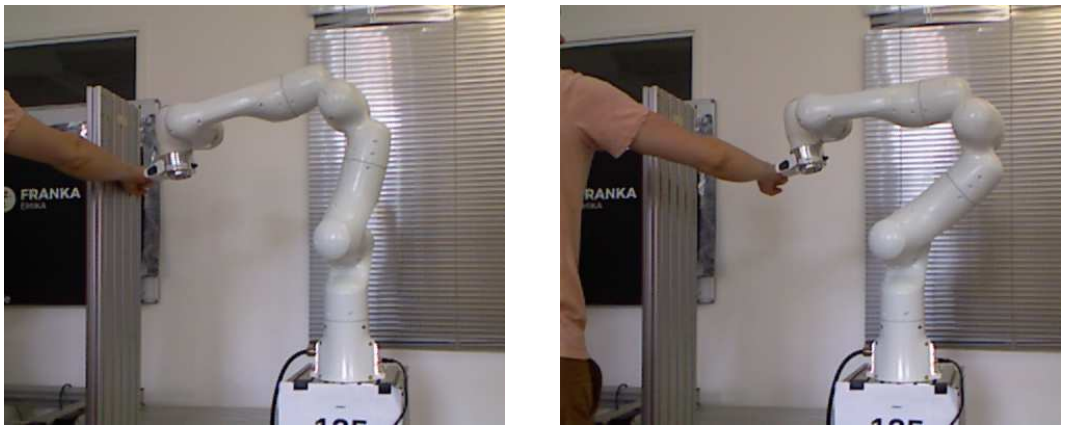


Figure 3.35: Results of the contact force experiment using the control law (3.10). In non-contact phases, $\boldsymbol{f}_{\text{lim}}$ of the velocity saturation law (3.17) is set to $\frac{f_{\max}}{\sqrt{3}}$ in all directions as there is no contact force direction. Since the direction of the estimated contact force fluctuates much when contact is released, $\boldsymbol{f}_{\text{lim}}$ also fluctuates shortly after the contact loss. When the desired force limit $f_{\max}$ is set to a high value, the maximal velocity $\boldsymbol{v}_{\max}$ of the velocity saturation law (3.17) can be too fast for safe human-robot collaboration. Hence, $\boldsymbol{v}_{\max}$ is upper bounded in dynamic situations. Only in quasi steady-state conditions, this upper bound is preempted to achieve the desired steady-state contact force. The effect of this upper bound on $\boldsymbol{v}_{\max}$ can be seen on the contact force decrease while pushing the robot back. As soon as the quasi steady-state is reached again, the contact force rises also. The first force peak while back-pushing the robot is due to additional damping forces of the control law. In the bottom graph the scale for ${}^R e_1$ is on the left, for ${}^R e_2$ and ${}^R e_3$ on the right.



(a) Hand clamping situation (b) Non-clamping hand contact

Figure 3.36: Hand clamping experiment. The robot is commanded with the same trajectory as in the second foam block experiment (Fig. 3.31). The contact point is obtained as described in Fig. 3.30.

we use our clamping identification algorithm ($d_{\clubsuit} = 9$, GJK) with a collaborative zone composed of the *m_hand* entry (see Tab. 3.3). As the force limits for the clamped body parts in Tab. 3.3 are based on preventing injuries above or equal to AIS 1, we use a lower force threshold for our experiments as we don't want to inflict pain on the human subject. Hence, whenever a possible clamping situation is signaled, we set the maximal force $f_{\max}$ to 50 N, otherwise it is set to 100 N. The parameters of the control laws are listed in Tab. 3.7 for the inertia shaping CCIC, and in Tab. 3.8 for the comparative control law (3.10)[11].

As illustrated in Fig. 3.37 the clamping identification algorithm signals a possible clamping situation before the actual clamping contact occurs. For both control laws, there is a contact force peak at the beginning of the collision. The force peak for the inertia shaping CCIC (top graph) is smaller since the impact velocity is also smaller due to a higher damping coefficient. As explained in Sec. 2.1.6, this transient force peak can be attenuated by damping the closed loop dynamics critically. However, critical damping is dependent on the stiffness k_e of the environment which is in general unknown or highly varying for human-robot contacts. Even if k_e is known, a critical damped system would lead to low velocities due to the velocity saturation law for high stiffness environments, making such a damping design unproductive.

As the end-effector has almost reached its target pose $\boldsymbol{x_d}$ in the clamping situation, the position error in initial error direction ${}^{R}e_1$ is so small that the velocity

[11]For both controllers, we set the first entry of $\boldsymbol{K}$ and $\boldsymbol{D}$ equal to the second and third entry in non-clamping cases. This makes it harder to push back the robot.

saturation law does not affect the steady-state force. This is the reason why the contact force for the comparative control law stays at roughly 23.4 N for the initial contact phase. When increasing ${}^{R}e_1$ by pushing the robot back, the steady-state contact force range from 44.4 N to 52.7 N during the second steady-state phase. Although clamping the hand is not possible anymore during this second steady-state, we treat this situation as a clamping scenario as the contact has not been interrupted since the initial clamped contact. Otherwise, increasing the force limit $f_{\max}$ in this case would make it difficult to free oneself from being clamped.
When the *initial* contact occurs further away from the wall environment (bigger ${}^{R}e_1$), the contact is labeled as non-clamping[12]. Here, $f_{\max}$ is set to $f_{\max} = 100$ N. However, the velocity saturation of the comparative control law is implemented such that it treats non-clamping contacts like no contact at all, i.e. $\boldsymbol{f}_{\lim} = \frac{f_{\max}}{\sqrt{3}}[1,1,1]^T$. As the contact force $\boldsymbol{f_c}$ acts mainly in $\boldsymbol{e_p}$ direction, it is consequently limited by $\sim \frac{100\text{ N}}{\sqrt{3}} = 57.7$ N. In the steady-state of the non-clamping contact, the contact force adheres to this limit when the robot is controlled with the comparative control law. Due to the inaccuracies of the modeled dynamics, as described in the previous experiment (Sec. 3.2.2), the inertia shaping CCIC stays far away from the force limit.

In the second experiment, the chest of the human subject is clamped by the robot, shown in Fig. 3.39a. The parameters for the controllers and for the velocity saturation are the same as in the previous experiment. However, the robot's workspace is now partitioned into multiple collaborative zones. The respective cuboid-shaped zones are stacked on top of each other, parallel to the floor. They are illustrated in Fig. 3.38a. Since this collaborative zone setup conveys a high amount of uncertainty (especially because the minimal and maximal dimensions of the *m_above_legs* entry are far apart from each other), possible clamping situations are identified much more conservatively. As reported in the top graph of Fig. 3.40, this even leads to wrong clamping contact identifications at $t = 4.7$ s, triggered by noisy contact force estimations. When using the inertia shaping CCIC law, the contact point $\boldsymbol{p_c}$ slides along the contact surface to reach its target pose $\boldsymbol{x_d}$, as explained in the *Obstacle circumvention* part of Sec. 2.1.6. This process is indicated in Fig. 3.40 (top) by a continuous decrease in the position error e_1 in x-direction and an adaption of the error e_2 in y-direction (the respective world coordinate frame is displayed in Fig. 3.38a.). As the target pose $\boldsymbol{x_d}$ is placed further in negative x-direction than the human subject is standing, the end-effector resolves the clamping situation autonomously by loosing contact with the human's chest after having slided all across it. In contrast, the comparative control law (3.10) does not change the principal impedance direction $\mathfrak{p}$. Therefore, the contact point does not slide across the chest, when being controlled with this control law. Consequently,

[12] As long as the clamping identification algorithm does not signal a potential clamping situation.

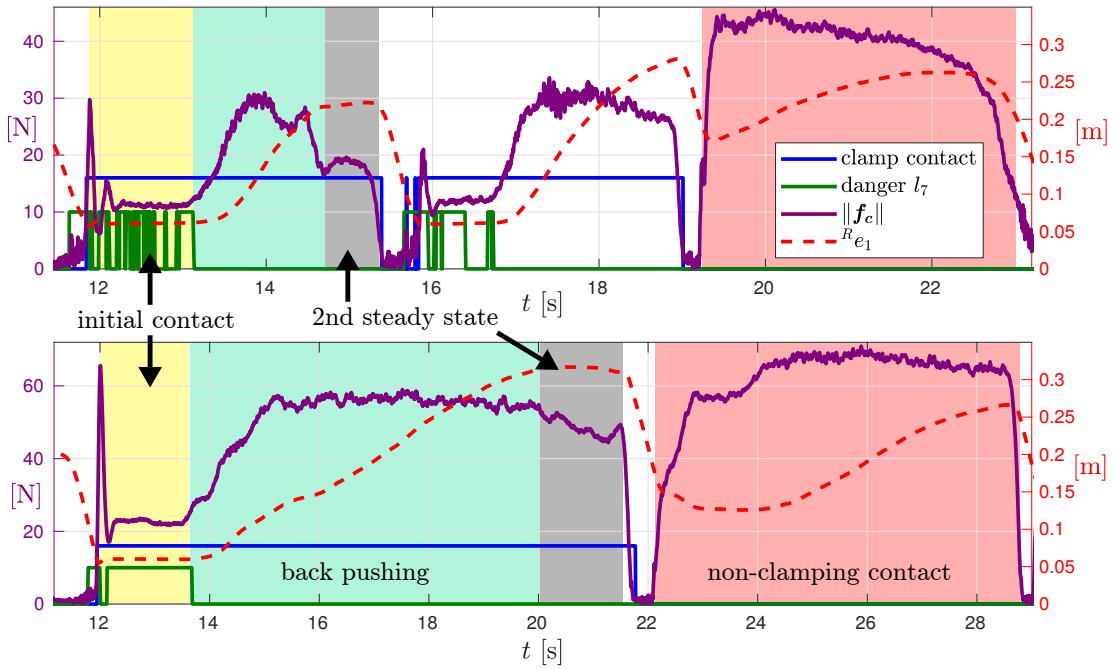
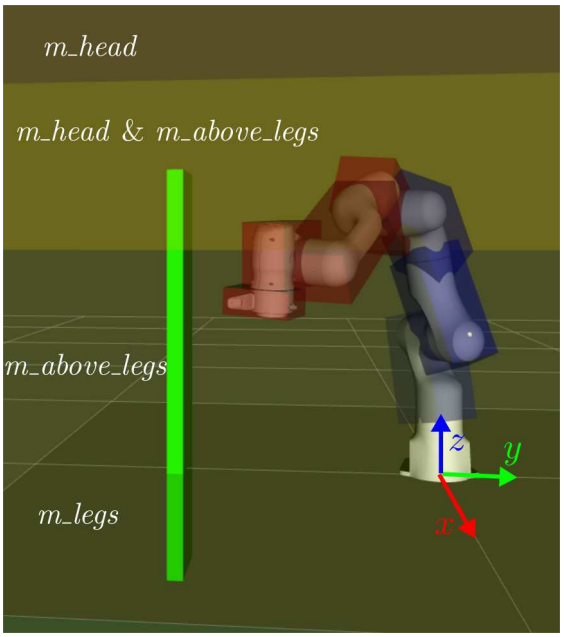
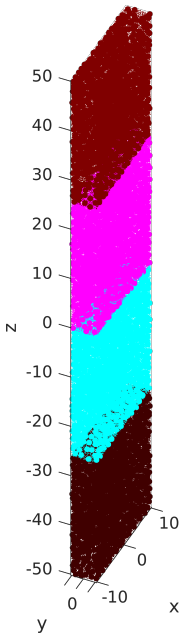


Figure 3.37: Results of the hand clamping experiment. For the top graph, the inertia shaping CCIC is used, for the bottom graph the comparative control law. Same phases are highlighted with identical colors, the legend applies to both graphs. In the top graph, there is a second clamping contact. After pushing the robot back, the hand was put back to the wall. As the robot follows immediately this motion to reach its target pose, there is an additional short contact before the actual second clamping contact. Since this contact already extracts some kinetic energy from the end-effector, the contact force peak for the second clamping contact is reduced. High phases of *danger* l_7 indicates that the clamping identification algorithm signals a potential clamping situation. When additionally the contact force pass a threshold, the *clamp contact* signal is triggered.

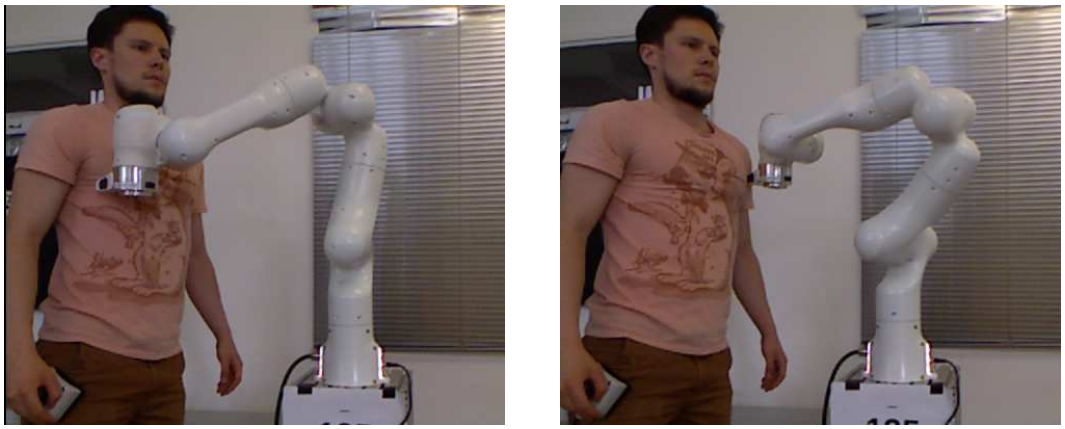


(a) Collaborative zones



(b) Sample points of wall environment

Figure 3.38: The collaborative zone setup for the chest clamping experiment is shown in *(a)*. The cuboid shaped zones are stacked on top of each other. The border between the *m_above_legs* and *m_legs* zone is the height of the mounting point of the robot. The zones' borders are better understandable in combination with Fig. 3.39. The minimal and maximal dimensions, as well as the force limit for these zones are listed in Tab. 3.3. For visualization, the wall is modeled as a simple cuboid. For the clamping identification however, the wall is represented as 4 OBBTrees, as illustrated in *(b)*. In total, there are 60893 sample points and 3736 leaf nodes. With the collaborative zone setup as in *(a)*, the displayed robot configuration leads to possible clamping situations for links $7, 6, 5$ and 4, as indicated by the red color of their OBB roots.



(a) Initial chest contact (b) Contact point slides along chest

Figure 3.39: Setup for the chest clamping experiment. When the robot is controlled with the inertia shaping CCIC *(b)*, the contact point slides along the chest until the contact is lost.

the position errors e_1 and e_2 stay constant after reaching a steady-state in the bottom graph of Fig 3.40. Although sliding along a human body part can resolve the clamping situation, it was perceived as rather inconvenient and painful for higher contact forces.

During the sliding action of the inertia shaping controlled robot, the effects of the stability controller (see Sec. 2.1.6 or Sec. 3.1.4) are clearly visible. Due to the sliding motion, the contact situation is constantly changing, affecting the contact force direction. This in turn leads to oscillations in the principal impedance direction $\mathfrak{p}$, and finally to oscillations in the position error ${}^R\boldsymbol{e_p}$ expressed in the coordinate frame aligned with $\mathfrak{p}$. When $\|\boldsymbol{e}_\perp\| = \left\| \left[{}^R e_2, {}^R e_3 \right]^T \right\|$ reaches a threshold, the stability controller is triggered. This correlation is illustrated in Fig. 3.41. As can be seen in both Fig. 3.40 and Fig. 3.41, the oscillations in the contact force $\boldsymbol{f_c}$ diminish while the stability controller is active.

3.3 Discussion

The results of the simulations and experiments demonstrate the capabilities of the proposed clamping conscious control pipeline as well as the effects of several design parameters. Concerning the clamping identification algorithm, using the SAT based distance approximations does not result in the expected boost in computation time. Its fast execution time is thwarted by the overhead of the OBBTree traversal. On the contrary, the small approximation errors lead in general to an inferior performance, compared to the GJK algorithm. A substantial decrease in

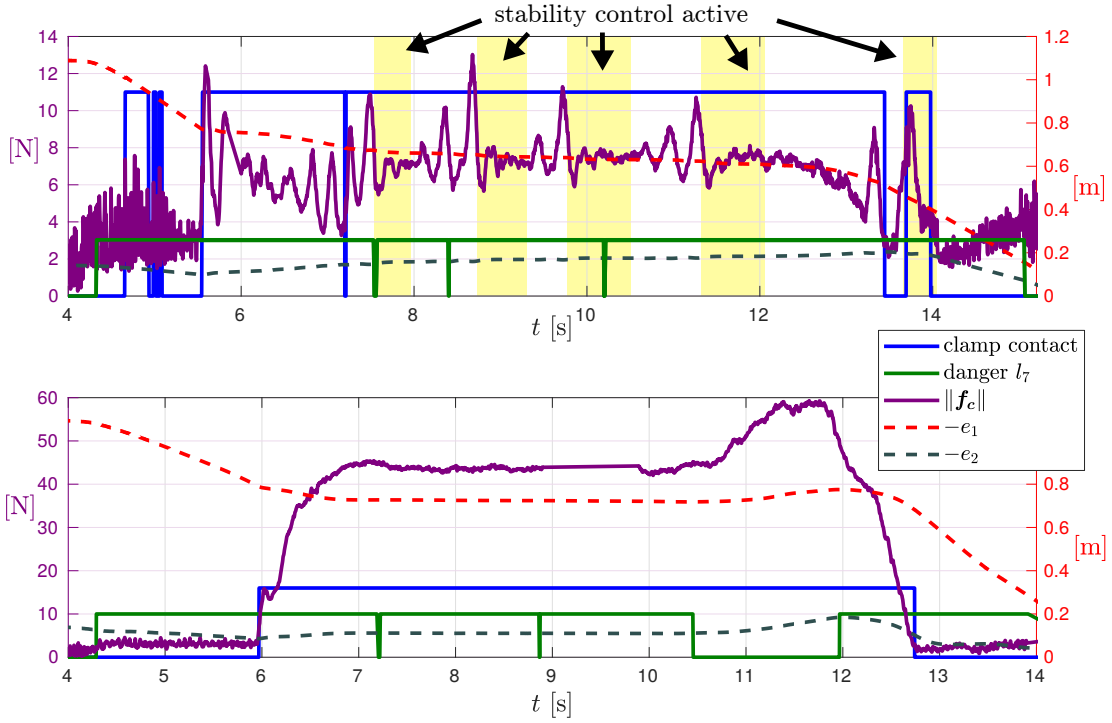


Figure 3.40: Results for the chest clamping experiment using the inertia shaping CCIC (top) and the comparative control law (bottom). The errors e_1 and e_2, expressed in the world coordinate frame, refer to the scale on the right. The remaining signals are as in Fig. 3.37. In the top graph, the phases are highlighted where the stability controller is active. As the comparative control law does not resolve the clamping situation autonomously, the end-effector has to be pushed away to free oneself from being clamped. This phase is indicated by the increase in the contact force in the bottom graph.

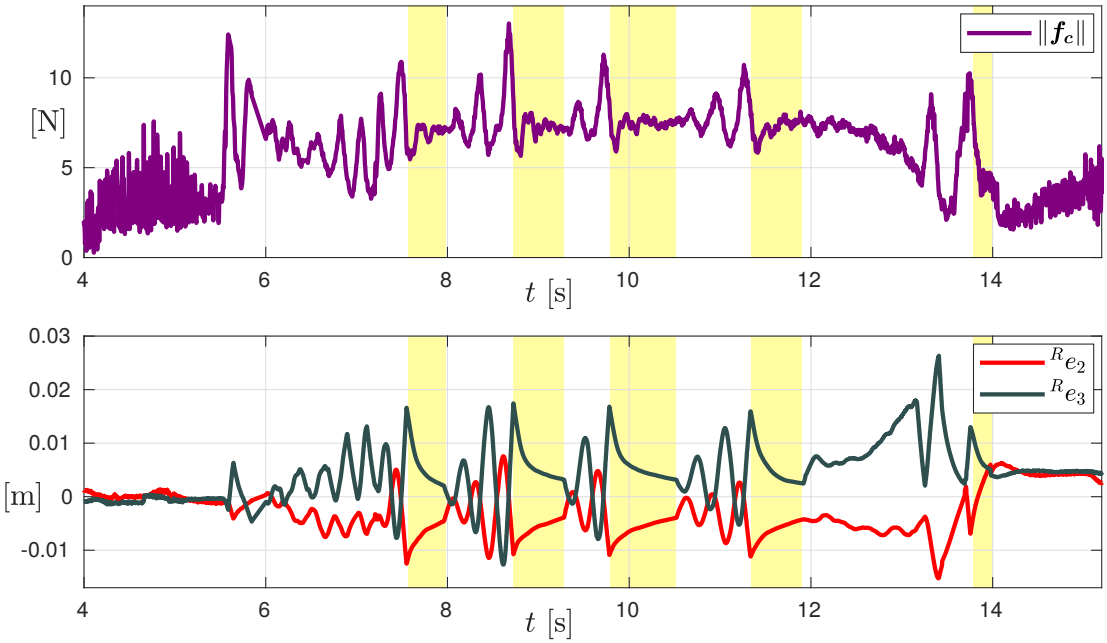


Figure 3.41: Correlation between the errors ${}^R e_1$ and ${}^R e_2$ perpendicular to $\mathfrak{p}$, the contact force $\|\boldsymbol{f_c}\|$ and the stability controller. The phases where the stability controller is active are highlighted in yellow.

computation time is achieved by incorporating the caching and sorting scheme as explained in Sec. 2.2.4. For the considered scenarios, this leads on average to a better performance than using a state of the art minimum distance implementation from fcl. However, in some cases, the maximum computation time and the maximum approximation error are higher for our algorithm. An additional feature of the proposed clamping identification algorithm that is not contained in the comparative fcl implementation, is the inclusion of velocity based EOC. On the one hand, these EOC slow down the clamping identification algorithm, but on the other hand, they lead to less false positive evaluations.

As confirmed by the experiments, clamping situations are reliably detected for all tested scenarios. Depending on the uncertainties of the body parts' extents that are present in the scene, the clamping situation is signaled within a long or short time period before the actual contact happens. Multiple reaction to clamping situations are possible. The robot can e.g. be put into zero gravity mode, exerting no steady state contact forces at all, or these contact forces can be limited to safe values. The inertia shaping control law that should do the latter, shows good performance in simulations, but fails to do so in the experiments. The accuracy of the modeled robot dynamics, as well as the accuracy of the contact force estimation are not sufficient to obtain the desired closed loop behavior. The comparative control law tracks the force limit better. However, as it does not

change the principal impedance direction, the contact point does not slide along the body part to resolve the clamping situation.

3.4 Future Work

For this thesis, the clamping identification algorithm is used in rather simple environments (a collection of geometric primitives for the simulations and a wall obstacle for the experiments). To better reason about its validity in real scenarios, more complex environments have to be tested. For example, complex environments might be scanned and transformed into point clouds. Furthermore, in its current state, the algorithm regards the entire environment as static. However, in most scenarios, the robot manipulates movable objects that should be considered in the clamping evaluation as well. This makes it necessary to monitor the scene, as otherwise their poses are unknown.

Monitoring the scene benefits the entire clamping conscious pipeline in multiple ways. Firstly, as described in Sec. 2.2.2, collaborative zones can be created dynamically, depending on the location of the respective body parts. To this end, skeleton tracking can be used as e.g. provided by the Microsoft Kinect [LSAD12, BFC14]. Bounding boxes around the tracked body parts can than serve as collaborative zones. Furthermore, even without skeleton tracking, contact points in clamping situations can be better estimated using real-time minimum distance information between the robot and the human as e.g. in [Hir15, SHL17]. Points of minimum distance between the robot and the human prior to the contact are likely to be close to the actual contact point. As the contact manifold is often occluded and cannot be captured directly with RGB-D cameras, the contact point (or even manifold) detection can be improved by building geometric models of the human body parts. This is e.g. done in real time (~ 2 s for all body parts) with errors close to the RGB-D sensor resolution in [Bar13]. After having build the model, the body part poses can be updated with even faster rates e.g. using skeleton tracking. Having models of the environment, the robot and the human body parts enables calculating the contact manifolds that are of interest in clamping situations: the contact manifolds between the body parts and the environment as well as the contact manifolds between the body parts and the robot. Those manifolds can be used to not only estimate the contact force on the manifolds, but to also estimate the contact pressure. As tolerable pressure limits (see [ISO16]) are exceeded faster than force limits for small contact manifolds, estimating the contact pressure is imperative when clamping occurs with pointy geometries.

Another limitation of the clamping identification algorithm is the fact that during the OBBTree traversal, the velocities of the OBBs are approximated with the linear velocity of the respective robot link (see Sec. 2.2.5). Consequently, the

clamping identification is only accurate for OBBs whose velocity does not differ much from the link's linear velocity. To remedy this limitation, the velocity of each OBB can be calculated and used, leading to more computational effort, though. A method that finds a suitable compromise between accuracy and computational effort regarding the velocity based EOC still needs to be found.
Further investigations should also be done regarding the obstacle circumvention feature of the proposed control scheme. Although clamping situations can be resolved this way, the sliding motion can be too painful due to shearing forces.

Chapter 4

Conclusion

When humans and collaborative robots work together, situations may arise where the robot clamps a human coworker. If these situations are not noticed or dealt with in a safe manner, the robot can inflict serious injuries on the coworker. This thesis presents methods to both identify robot-human clamping situations and to render these situations safe. Starting from distance and velocity based conditions that apply to clamping situations, we create an identification algorithm that checks in real-time whether these conditions are fulfilled. To this end, we approximate the robot and the environment with hierarchies of Bounding Volumes, representing a coarse to fine approximation scheme. As we do not track the coworkers in the workspace, the system possesses only vague information about their poses and extents through the configuration of so called collaborative zones. These zones are placed throughout the workspace and each one contains information which body parts could be present in the respective zone. The extents of the body parts are then queried from a database that contains the minimal and maximal dimension of the respective body parts[1]. As those extents go along with certain uncertainties, the clamping identification algorithm uses worst-case assumptions to deliver safe results. The accuracy of the algorithm is thus bounded by these uncertainties.

Checking the clamping conditions is done by iteratively refining the approximations of the robot and the environment. If the clamping conditions do not apply for a coarse approximation, a clamping situation can be ruled out early. Several parameters of this hierarchy of approximations are evaluated in terms of computation speed and accuracy. As the distance calculation between two Oriented Bounding Boxes plays a major role therein, we introduce an efficient way to approximate both the minimum and maximum distance between two Oriented Bounding Boxes based on the Separation of Axes Theorem. However, simulations have shown that the identification algorithm possesses too much computational overhead in order for this approximative method to be useful. Nonetheless, the

[1] Obtained from anthropometric data representing 98 % of the US population.

proposed clamping identification algorithm outperforms state of the art distance calculation implementations that we adapted to our use-case. For the simulated scenarios, an average rate of 500 Hz is obtained.

To limit the contact forces in clamping situations, a variable impedance control scheme is presented that modifies the imposed dynamics according to the contact force. In theory, the imposed dynamics are decoupled in the direction of the contact force, making it only necessary to limit the contact force in one coordinate direction. Being potentially unstable, we pair this control scheme with a stability controller that brings back the system to a stable state. Stability constraints for the stability controller are obtained with Lyapunov theory. In practice, the control scheme performs poorly with regard to reaching a desired force in steady-state conditions. Therefore, another Cartesian impedance controller is proposed that performs better under real conditions. Both controllers are evaluated and compared in experiments where body parts of a human subject are clamped. In all experiments, the clamping situations are identified successfully, and the steady state contact force is limited to a safe value. Future work will mainly focus on tracking human coworkers in the workspace as this reduces dramatically the uncertainties present in the clamping identification algorithm. Additionally, when fitting a geometric model to the tracked coworkers, contacts can be rendered safe not only regarding contact forces, but also regarding contact pressures.

Appendix A

Distance Calculation

A.1 Maximum Distance and Minkowski Sum

The Minkowski sum of two sets of points Ω_A and Ω_B is the addition of every point in Ω_A to every point in Ω_B:

$$\Omega_A + \Omega_B = \{a + b | a \in \Omega_A, b \in \Omega_B\}.$$

Calculating all point to point distances between Ω_A and Ω_B can be expressed by $\|\Omega_A - \Omega_B\|$. The maximum distance between Ω_A and Ω_B is therefore the farthest point of $\Omega_A - \Omega_B$ from the origin. When Ω_A and Ω_B represents polygons (in 2-D) or polyhedra (in 3-D), their Minkowski sum is again a polygon or polyhedron. In this case, the farthest point from the origin is coincident with a vertex of the polygon (polyhedron). This is visualized for the 2-D case in Fig. A.1. If a point x is the farthest point from the origin, all points of the Minkowski sum are contained in the circle around the origin with radius r, where r is the distance of x from the origin. The direct neighboring points of x must consequently also be within this circle. This can only be the case when x lays on a vertex (as point q). Otherwise, the neighboring points to one side (as for point g), or the neighboring points on both sides (as for point p) are outside the circle. The same concept applies for the 3-D case.

A.2 Distance Projection on Additional Separation Axis

Given the projection axis

$$\boldsymbol{l} = (\boldsymbol{a_i} \times \boldsymbol{b_j}) \times \boldsymbol{a_i}, \tag{A.1}$$

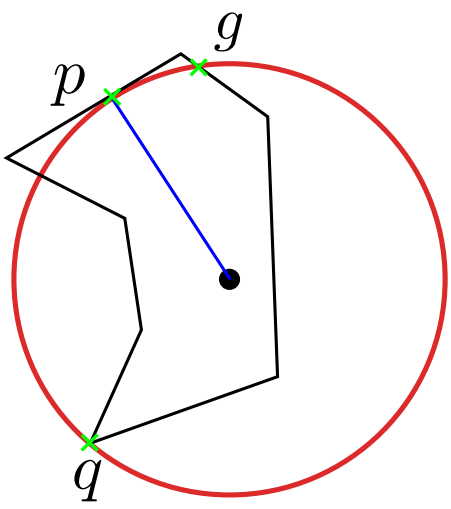


Figure A.1: The black polygon is the Minkowski sum of two polygons. Every point within this polygon (including the polygon) is element of the Minkowski sum. A point x belonging to the Minkowski sum that is farthest away from the origin (black circle) must be coincident with a vertex of the polygon. Even if the polygon's edge touches the circle of equidistance tangentially in p, the neighboring points of p are outside this circle.

we derive here the simplifications of the projected distance approximations (2.105)-(2.107). Projecting OBB A as per (2.4) on $\boldsymbol{l}$ yields

$$r^A = \sum_k |\alpha_k \boldsymbol{a_k} \cdot ((\boldsymbol{a_i} \times \boldsymbol{b_j}) \times \boldsymbol{a_i})| \tag{A.2}$$

$$= \sum_k | - \alpha_k \boldsymbol{a_k} \cdot (\boldsymbol{a_i} \times (\boldsymbol{a_i} \times \boldsymbol{b_j}))| \tag{A.3}$$

$$= \sum_k | - \alpha_k (\boldsymbol{a_i} \times \boldsymbol{b_j}) \cdot (\boldsymbol{a_k} \times \boldsymbol{a_i})|. \tag{A.4}$$

The transition from (A.2) to (A.3) follows from the cross product's anticommutativity, (A.4) from the circular shift property of the *scalar triple product* [Wei02]:

$$\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c}) = \boldsymbol{c} \cdot (\boldsymbol{a} \times \boldsymbol{b}).$$

For $k = i$, the cross product $\boldsymbol{a_k} \times \boldsymbol{a_i}$ evaluates to zero. Hence,

$$r^A = \sum_{k,k \neq i} | - \alpha_k (\boldsymbol{a_i} \times \boldsymbol{b_j}) \cdot (\boldsymbol{a_k} \times \boldsymbol{a_i})| \tag{A.5}$$

$$= \sum_{k,k \neq i} |\alpha_k (\boldsymbol{a_i} \times \boldsymbol{b_j}) \cdot \boldsymbol{a_m}|, \tag{A.6}$$

with

$$\boldsymbol{a_m} \perp \boldsymbol{a_k} \wedge \boldsymbol{a_m} \perp \boldsymbol{a_i}.$$

Applying the circular shift property of the scalar triple product again on (A.6) gives

$$r^A = \sum_{k,k\neq i} |\alpha_k \boldsymbol{b_j} \cdot (\boldsymbol{a_m} \times \boldsymbol{a_i})| \tag{A.7}$$

$$= \sum_{k,k\neq i} |\alpha_k \boldsymbol{b_j} \cdot \boldsymbol{a_k}| \tag{A.8}$$

$$= \sum_{k,k\neq i} \alpha_k \bar{R}_{[k,j]} \tag{A.9}$$

$$= \alpha_{[i\gg 1]} \bar{R}_{[i\gg 1,j]} + \alpha_{[i\gg 2]} \bar{R}_{[i\gg 2,j]}. \tag{A.10}$$

Equation (A.8) follows from the perpendicular properties of $\boldsymbol{a_m}$, equation (A.9) from (2.90), and equation (A.10) is just a different notation of (A.10).

The projection of OBB B on $\boldsymbol{l}$, as per (2.4), is

$$r^B = \sum_k |\beta_k \boldsymbol{b_k} \cdot ((\boldsymbol{a_i} \times \boldsymbol{b_j}) \times \boldsymbol{a_i})| \tag{A.11}$$

$$= \sum_k | - \beta_k \boldsymbol{b_k} \cdot (\boldsymbol{a_i} \times (\boldsymbol{a_i} \times \boldsymbol{b_j}))| \tag{A.12}$$

$$= \sum_k | - \beta_k (\boldsymbol{a_i} \times \boldsymbol{b_j}) \cdot (\boldsymbol{b_k} \times \boldsymbol{a_i})| \tag{A.13}$$

$$= \sum_k |\beta_k (\boldsymbol{b_j} \times \boldsymbol{a_i}) \cdot (\boldsymbol{b_k} \times \boldsymbol{a_i})|, \tag{A.14}$$

$$\tag{A.15}$$

where the steps (A.11)-(A.13) are analogously to (A.2)-(A.4). Next, we split the sum into the case $k = j$ and $k \neq j$:

$$r^B = \sum_{k,k=j} |\beta_k (\boldsymbol{b_j} \times \boldsymbol{a_i}) \cdot (\boldsymbol{b_k} \times \boldsymbol{a_i})| + \sum_{k,k\neq j} |\beta_k (\boldsymbol{b_j} \times \boldsymbol{a_i}) \cdot (\boldsymbol{b_k} \times \boldsymbol{a_i})| \tag{A.16}$$

$$= \beta_j \|\boldsymbol{b_j} \times \boldsymbol{a_i}\|^2 + \sum_{k,k\neq j} |\beta_k (\boldsymbol{b_j} \times \boldsymbol{a_i}) \cdot (\boldsymbol{b_k} \times \boldsymbol{a_i})| \tag{A.17}$$

$$= \beta_j \frac{1}{\lambda^2} + \sum_{k,k\neq j} |\beta_k (\boldsymbol{b_j} \times \boldsymbol{a_i}) \cdot (\boldsymbol{b_k} \times \boldsymbol{a_i})|, \tag{A.18}$$

with

$$\lambda = \frac{1}{\|\boldsymbol{a_i} \times \boldsymbol{b_j}\|}$$

as in (2.97). The second summand of (A.18) can be simplified by using the three-dimensional *Binet-Cauchy Identity* [Wei02], which states that

$$(\boldsymbol{a} \times \boldsymbol{b}) \cdot (\boldsymbol{c} \times \boldsymbol{d}) = (\boldsymbol{a} \cdot \boldsymbol{c})(\boldsymbol{b} \cdot \boldsymbol{d}) - (\boldsymbol{a} \cdot \boldsymbol{d})(\boldsymbol{b} \cdot \boldsymbol{c}).$$

Consequently, (A.18) becomes

$$r^B = \beta_j \frac{1}{\lambda^2} + \sum_{k,k\neq j} \left|\beta_k \left((\boldsymbol{b_j}\cdot\boldsymbol{b_k})(\boldsymbol{a_i}\cdot\boldsymbol{a_i}) - (\boldsymbol{b_j}\cdot\boldsymbol{a_i})(\boldsymbol{a_i}\cdot\boldsymbol{b_k})\right)\right| \tag{A.19}$$

$$= \beta_j \frac{1}{\lambda^2} + \sum_{k,k\neq j} \left|\beta_k (\boldsymbol{b_j}\cdot\boldsymbol{a_i})(\boldsymbol{a_i}\cdot\boldsymbol{b_k})\right| \tag{A.20}$$

$$= \beta_j \frac{1}{\lambda^2} + \sum_{k,k\neq j} \beta_k \bar{R}_{[i,j]} \bar{R}_{[i,k]} \tag{A.21}$$

$$= \beta_j \frac{1}{\lambda^2} + \bar{R}_{[i,j]} \sum_{k,k\neq j} \beta_k \bar{R}_{[i,k]} \tag{A.22}$$

$$= \beta_j \frac{1}{\lambda^2} + \bar{R}_{[i,j]} \left(\beta_{j\gg 1} \bar{R}_{[i,j\gg 1]} + \beta_{j\gg 2} \bar{R}_{[i,j\gg 2]}\right). \tag{A.23}$$

To simplify the projection of the translation $\boldsymbol{t}\cdot\boldsymbol{l}$, we utilize the *vector triple product* identity [Wei02]:

$$\boldsymbol{a} \times (\boldsymbol{b} \times \boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{a}\cdot\boldsymbol{c}) - \boldsymbol{c}(\boldsymbol{a}\cdot\boldsymbol{b}).$$

Thus, the projection axis $\boldsymbol{l}$ in (A.1) can be reformulated as

$$\boldsymbol{l} = \boldsymbol{b_j}(\boldsymbol{a_i}\cdot\boldsymbol{a_i}) - \boldsymbol{a_i}(\boldsymbol{a_i}\cdot\boldsymbol{b_j}) \tag{A.24}$$

$$= \boldsymbol{b_j} - \boldsymbol{a_i}\, {}^A_B R_{[i,j]} \tag{A.25}$$

$$= \begin{bmatrix} {}^A_B R_{[1,j]} \\ {}^A_B R_{[2,j]} \\ {}^A_B R_{[3,j]} \end{bmatrix} - \boldsymbol{a_i}\, {}^A_B R_{[i,j]}, \tag{A.26}$$

where the representations of axes $\boldsymbol{b_j}$ as in (2.90) are used. With

$$\hat{\sigma}(i,j) = \begin{cases} 1, & \text{for } i = j \\ 0, & \text{otherwise} \end{cases} \tag{A.27}$$

we can write $\boldsymbol{a_i}$ as

$$\boldsymbol{a_i} = \begin{bmatrix} \hat{\sigma}(i,1) \\ \hat{\sigma}(i,2) \\ \hat{\sigma}(i,3) \end{bmatrix}. \tag{A.28}$$

When substituting (A.28) into (A.26), we can write (A.26) row-wise:

$$\begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = \begin{bmatrix} {}^A_B R_{[1,j]} - \hat{\sigma}(i,1)\, {}^A_B R_{[i,j]} \\ {}^A_B R_{[2,j]} - \hat{\sigma}(i,2)\, {}^A_B R_{[i,j]} \\ {}^A_B R_{[3,j]} - \hat{\sigma}(i,3)\, {}^A_B R_{[i,j]} \end{bmatrix} \tag{A.29}$$

$$= \begin{bmatrix} {}^A_B R_{[1,j]} - \hat{\sigma}(i,1)\, {}^A_B R_{[1,j]} \\ {}^A_B R_{[2,j]} - \hat{\sigma}(i,2)\, {}^A_B R_{[2,j]} \\ {}^A_B R_{[3,j]} - \hat{\sigma}(i,3)\, {}^A_B R_{[3,j]} \end{bmatrix} \tag{A.30}$$

$$= \begin{bmatrix} {}^A_B R_{[1,j]}(1 - \hat{\sigma}(i,1)) \\ {}^A_B R_{[2,j]}(1 - \hat{\sigma}(i,2)) \\ {}^A_B R_{[3,j]}(1 - \hat{\sigma}(i,3)) \end{bmatrix} \tag{A.31}$$

The step from (A.29) to (A.30) is possible since $\hat{\sigma}(i,j)$ functions as a selective variable that allows only equal values for i and j. From (A.31) we see that $l_i = 0$. Projecting the translation $\boldsymbol{t}$ onto $\boldsymbol{l}$, we obtain

$$|\boldsymbol{t} \cdot \boldsymbol{l}| = |\sum_{k,k\neq i} t_k \, {}^A_B R_{[k,j]}| \tag{A.32}$$

$$= |t_{[i\gg 1]} \, {}^A_B R_{[i\gg 1,j]} + t_{[i\gg 2]} \, {}^A_B R_{[i\gg 2,j]}| \tag{A.33}$$

The simplifications (A.10), (A.23) and (A.33) are the ones that are used in (2.105)-(2.107).

Appendix B

Clamping Conscious Control

B.1 Solution of the One-Dimensional System Dynamics

To solve the non-homogeneous second order linear differential equation

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = \frac{f_0}{m}, \tag{B.1}$$

we first have to solve the homogeneous equation

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = 0. \tag{B.2}$$

Assuming $x(t) = e^{zt}$ as a candidate solution to (B.2), the homogeneous equation becomes

$$e^{zt}\left(z^2 + 2\zeta\omega_n z + \omega_n^2\right) = 0, \tag{B.3}$$

with the roots

$$z_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}. \tag{B.4}$$

Depending on the value of ζ, the roots are either imaginary (underdamped), real and identical (critically damped), or real and not identical (overdamped).

Underdamped

With $\zeta < 1$, the roots (B.4) are

$$\begin{aligned} z_1 &= -\zeta\omega_n + i\omega_n\sqrt{1-\zeta^2} \\ z_2 &= -\zeta\omega_n - i\omega_n\sqrt{1-\zeta^2}, \end{aligned} \tag{B.5}$$

with i being the imaginary unit. The general solution of (B.2) is a linear combination of both modal solutions

$$x_h(t) = A_1 e^{z_1 t} + A_2 e^{z_2 t}. \tag{B.6}$$

Substituting (B.5) into (B.6), we arrive at

$$x_h(t) = e^{-\zeta\omega_n t}\left(C_1\cos(\omega_d t) + C_2\sin(\omega_d t)\right), \tag{B.7}$$

with

$$\begin{aligned} C_1 &= A_1 + A_2 \\ C_2 &= A_1 - A_2 \\ \omega_d &= \omega_n\sqrt{1-\zeta^2}. \end{aligned}$$

To solve the non-homogeneous equation (B.1), we take the solution $x_h(t)$ and add a particular solution $x_p(t)$ to it that solves (B.1):

$$x(t) = x_h(t) + x_p(t). \tag{B.8}$$

The right-hand side of (B.1) can be reformulated as

$$\frac{f_0}{m} = \frac{f_0}{k_e}\omega_n^2$$

by using $\omega_n = \sqrt{\frac{k_e}{m}}$. Hence, a particular solution of (B.1) is

$$x_p(t) = \frac{f_0}{k_e}. \tag{B.9}$$

Using (B.8), the general solution to (B.1) becomes

$$x(t) = e^{-\zeta\omega_n t}\left(C_1\cos(\omega_d t) + C_2\sin(\omega_d t)\right) + \frac{f_0}{k_e}. \tag{B.10}$$

The constants C_1 and C_2 can be determined with the initial conditions

$$\begin{aligned} x(0) &= x_e \\ \dot{x}(0) &= v_0. \end{aligned} \tag{B.11}$$

It follows that

$$\begin{aligned} x(0) &= C_1 + \frac{f_0}{k_e} \stackrel{!}{=} x_e \Rightarrow C_1 = x_e - \frac{f_0}{k_e} = x_e - \frac{k_e x_e - f_{\max}}{k_e} = \frac{f_{\max}}{k_e} \\ \dot{x}(0) &= -\zeta\omega_n C_1 + C_2\omega_d \stackrel{!}{=} v_0 \Rightarrow C_2 = \frac{v_0 + \zeta\omega_n C_1}{\omega_d} = \frac{v_0 + \zeta\omega_n \frac{f_{\max}}{k_e}}{\omega_d}. \end{aligned} \tag{B.12}$$

When writing the linear combination of sine and cosine in (B.10) as a single sine [Caz07] and substituting (B.12), we obtain the solution (2.26).

Critically damped

With $\zeta = 1$, the roots in (B.4) become identical

$$z_{1,2} = -\zeta\omega_n. \tag{B.13}$$

The general solution of (B.2) in this case is [MMOL11]

$$x_h(t) = e^{-\zeta\omega_n t}\left(C_1 + C_2 t\right). \tag{B.14}$$

Adding the same particular solution $x_p(t)$ (B.9) to (B.14), we arrive at

$$x(t) = e^{-\zeta\omega_n t}\left(C_1 + C_2 t\right) + \frac{f_0}{k_e}. \tag{B.15}$$

The constants C_1 and C_2 are derived from the initial conditions (B.11):

$$\begin{aligned} x(0) &= C_1 + \frac{f_0}{k_e} \stackrel{!}{=} x_e \Rightarrow C_1 = x_e - \frac{f_0}{k_e} = x_e - \frac{k_e x_e - f_{\max}}{k_e} = \frac{f_{\max}}{k_e} \\ \dot{x}(0) &= -\omega_n C_1 + C_2 \stackrel{!}{=} v_0 \Rightarrow C_2 = v_0 + \omega_n C_1 = v_0 + \omega_n \frac{f_{\max}}{k_e}. \end{aligned} \tag{B.16}$$

Overdamped

The roots (B.4) for $\zeta > 1$ take the form

$$\begin{aligned} z_1 &= -\zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1} \\ z_2 &= -\zeta\omega_n - \omega_n\sqrt{\zeta^2 - 1}, \end{aligned} \tag{B.17}$$

leading to the homogeneous solution

$$x_h(t) = C_1 e^{z_1 t} + C_2 e^{z_2 t}, \tag{B.18}$$

and to the general solution of (B.1)

$$x(t) = C_1 e^{z_1 t} + C_2 e^{z_2 t} + \frac{f_0}{k_e}. \tag{B.19}$$

Plugging in the initial conditions (B.11) leads to a system of linear equations for the constants C_1 and C_2

$$\begin{array}{ccccc} C_1 & + & C_2 & = & \frac{f_{\max}}{k_e} \\ z_1 C_1 & + & z_2 C_2 & = & v_0 \end{array}$$

Solving this system of linear equations produces (2.28).

B.2 Vector Alignment

The method for finding the rotation matrix that aligns an unit vector onto another is taken from [vdB16] and is added here for completeness. In our case, we have the positional error $\boldsymbol{e_p}$ at two consecutive time instances $\boldsymbol{e_p}(t_0) = \boldsymbol{e_{p0}}$ and $\boldsymbol{e_p}(t_1) = \boldsymbol{e_{p1}}$. At time t_0, frame ${}^{e}\chi$ was aligned with $\boldsymbol{e_{p0}}$ (frame ${}^{e0}\chi$), at time t_1 with $\boldsymbol{e_{p1}}$ (frame ${}^{e1}\chi$). The rotation matrix that rotates frame ${}^{e0}\chi$ to ${}^{e1}\chi$ is ${}^{\boldsymbol{e_{p1}}}_{\boldsymbol{e_{p0}}}\boldsymbol{R}$. The rotation of the robot's base frame ${}^{C}\chi$ to ${}^{e1}\chi$ is expressed by the rotation matrix $\boldsymbol{R_p}(\boldsymbol{e_{p1}})$:

$$\boldsymbol{R_p}(\boldsymbol{e_{p1}}) = {}^{\boldsymbol{e_{p1}}}_{\boldsymbol{e_{p0}}}\boldsymbol{R}\boldsymbol{R_p}(\boldsymbol{e_{p0}}).$$

With $\omega = \frac{\boldsymbol{e_{p0}}}{\|\boldsymbol{e_{p0}}\|} \times \frac{\boldsymbol{e_{p1}}}{\|\boldsymbol{e_{p1}}\|}$ and $cos(\alpha) = \frac{\boldsymbol{e_{p0}}}{\|\boldsymbol{e_{p0}}\|} \cdot \frac{\boldsymbol{e_{p1}}}{\|\boldsymbol{e_{p1}}\|}$, ${}^{\boldsymbol{e_{p1}}}_{\boldsymbol{e_{p0}}}\boldsymbol{R}$ can be calculated with

$${}^{\boldsymbol{e_{p1}}}_{\boldsymbol{e_{p0}}}\boldsymbol{R} = \boldsymbol{I_3} + [\boldsymbol{\omega}]_\times + [\boldsymbol{\omega}]_\times^2 \frac{1}{1 + cos(\alpha)}.$$

Here, $\boldsymbol{I_3}$ is the identity matrix and the operator $[\cdot]_\times$ transforms its operand into a skew symmetric matrix:

$$[\boldsymbol{\omega}]_\times \triangleq \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}.$$

B.3 Positive Definiteness of Lyapunov Function

We will show that

$$V(\dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{e_p}, t) = \frac{1}{2}\left(\boldsymbol{R_p}\dot{\boldsymbol{e}}_{\boldsymbol{p}}\right)^T \boldsymbol{\Lambda_{d,p}}\boldsymbol{R_p}\dot{\boldsymbol{e}}_{\boldsymbol{p}} + \frac{1}{2}\left(\boldsymbol{R_p}\boldsymbol{e_p}\right)^T \boldsymbol{K_p}\boldsymbol{R_p}\boldsymbol{e_p} \tag{B.20}$$

is positive definite when $\boldsymbol{\Lambda_{d,p}}$ and $\boldsymbol{K_p}$ are positive definite and diagonal. Consider a diagonal, positive definite matrix $\boldsymbol{A}$

$$\boldsymbol{A} = \begin{bmatrix} a_1 & 0 & 0 & \dots \\ 0 & a_2 & 0 & \dots \\ 0 & 0 & a_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} \sqrt{(a_1)}\sqrt{(a_1)} & 0 & 0 & \dots \\ 0 & \sqrt{(a_2)}\sqrt{(a_2)} & 0 & \dots \\ 0 & 0 & \sqrt{(a_3)}\sqrt{(a_3)} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{(a_1)} & 0 & 0 & \dots \\ 0 & \sqrt{(a_2)} & 0 & \dots \\ 0 & 0 & \sqrt{(a_3)} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \sqrt{(a_1)} & 0 & 0 & \dots \\ 0 & \sqrt{(a_2)} & 0 & \dots \\ 0 & 0 & \sqrt{(a_3)} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \breve{\boldsymbol{A}}\breve{\boldsymbol{A}},$$

with a_i positive, scalar values. When $\boldsymbol{A}$ is multiplied from the left and from the right with a rotation matrix - vector product as in (B.20):

$$(\boldsymbol{Re})^T \boldsymbol{ARe} = (\boldsymbol{Re})^T \breve{\boldsymbol{A}}\breve{\boldsymbol{A}}\boldsymbol{Re} = \boldsymbol{e^T R^T}\breve{\boldsymbol{A}}^T\breve{\boldsymbol{A}}\boldsymbol{Re} = \left(\breve{\boldsymbol{A}}\boldsymbol{Re}\right)^T \breve{\boldsymbol{A}}\boldsymbol{Re}. \tag{B.21}$$

the entire product will be positive since $\left(\breve{\boldsymbol{A}}\boldsymbol{R}\boldsymbol{e}\right)^T \breve{\boldsymbol{A}}\boldsymbol{R}\boldsymbol{e}$ is a dot product with identical vectors. Our Lyapunov function is a sum of two such products, which has to be positive as a consequence. Additionally, it follows that

$$\left(\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}\right)^T = \boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R} = \boldsymbol{R}^T\breve{\boldsymbol{A}}\breve{\boldsymbol{A}}\boldsymbol{R} = \left(\breve{\boldsymbol{A}}\boldsymbol{R}\right)^T \breve{\boldsymbol{A}}\boldsymbol{R}$$

is symmetric and positive definite.

B.4 Stable Trajectory Adaption

Here we will give an example on how to adapt a desired trajectory $\boldsymbol{x_d}$ such that its dynamics will be stable when controlled with the stability controller of Sec. 2.1.6. The necessary conditions to fulfill are described in (2.60)-(2.62). For the sake of simplicity, we consider the special case

$$\begin{aligned}\Delta_{\Lambda_1\Lambda_2} &= \Delta_{\Lambda_1\Lambda_3}\\ D_{p,[2,2]} &= D_{p,[3,3]}.\end{aligned}$$

Thus, condition (2.62) simplifies to

$$4D_{p,[1,1]}D_{p,[3,3]} > \frac{\left(\Delta_{\Lambda_1\Lambda_3}\right)^2}{\left\|\boldsymbol{e_p}\right\|^2}\left(\left({}^e\dot{e}_{p,[3]}\right)^2 + \left({}^e\dot{e}_{p,[2]}\right)^2\right). \tag{B.22}$$

Dropping the ${}^e\cdot$ superscript and introducing the short notations $D_{p,[i,i]} = D_i$ and ${}^e\dot{e}_{p,[i]} = \dot{e}_i$, (B.22) can be rearranged to

$$\begin{aligned}\left(\dot{e}_3\right)^2 + \left(\dot{e}_2\right)^2 &< \frac{\left\|\boldsymbol{e_p}\right\|^2 4D_1D_3}{\left(\Delta_{\Lambda_1\Lambda_3}\right)^2}\\ \left(\dot{x}_3 - \dot{x}_{d,[3]}\right)^2 + \left(\dot{x}_2 - \dot{x}_{d,[2]}\right)^2 &< \frac{\left\|\boldsymbol{e_p}\right\|^2 4D_1D_3}{\left(\Delta_{\Lambda_1\Lambda_3}\right)^2}.\end{aligned} \tag{B.23}$$

The change in the desired position $x_{d,[i]}$ can then be chosen such to satisfy (B.23).

List of Figures

List of Tables

List of Algorithms

Abbreviations

AABB Axis Aligned Bounding Box

AIS Abbreviated Injury Scale

BV Bounding Volume

BVH Bounding Volume Hierarchy

CCIC Clamping Conscious Impedance Control

cobot collaborative robot

DM² Distributed Macro-Mini

EOC Early Out Criterion

FCI Franka Control Interface

fcl flexible collision library

GJK Gilbert-Johnson-Keerthi (algorithm)

HIC_{36} Head Injury Criterion

i.i.d. independent and identically distributed

LWRIII Light Weight Robot III

MR Magneto-Rheological

OBB Oriented Bounding Box

pdf positive definite

pHRI physical Human Robot Interaction

ROS Robotic Operation System

SAT Separation Axis Theorem

SEA Series Elastic Actuator

VIA Variable Impedance Approach

VST Variable Stiffness Transmission

Nomenclature

$\mathbf{0_n}$ Matrix of zeros $\in \mathbb{R}^{n \times n}$.

$\boldsymbol{C}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ Coriolis and centrifugal matrix containing the Christoffel symbols.

$d_{\clubsuit}$ Depth of OBBTree.

$d_{\mathbf{max}}$ Maximum distance.

$\tilde{d}_{\mathbf{max}}$ Approximated maximum distance.

$d_{\mathbf{min}}$ Minimum distance.

$\tilde{d}_{\mathbf{min}}$ Approximated minimum distance.

$\boldsymbol{e}$ Cartesian pose error defined as $\boldsymbol{x} - \boldsymbol{x_d}$.

$\boldsymbol{f_c}$ Estimated contact forces.

$\boldsymbol{f}_{\mathbf{ext}}$ Vector of external forces acting on the robot's end-effector.

$f_{\mathbf{max}}$ Maximum allowed force value for a body part or collaborative zone.

$\boldsymbol{g}(\boldsymbol{q})$ Torque vector that is due to gravity acting on the robot.

$\boldsymbol{I_n}$ Identity matrix $\in \mathbb{R}^{n \times n}$.

$\boldsymbol{J}$ Jacobian matrix.

$\boldsymbol{l}$ Separation vector.

$\boldsymbol{\Lambda}(\boldsymbol{x})$ Inertia matrix in the operational space framework.

$\boldsymbol{M}(\boldsymbol{q})$ Inertia matrix, symmetric and positive definite.

$\bar{\boldsymbol{M}}(\boldsymbol{q})$ Modified inertia matrix of elastic joint robot to be used in rigid joint model.

$\mathfrak{p}$ Principal Impedance Direction.

$\boldsymbol{q}$ Vector of joint angles of the manipulator.

$\dot{\boldsymbol{q}}$ Vector of joint velocities.

$\ddot{\boldsymbol{q}}$ Vector of joint accelerations.

$\mathbf{size_{max}}$ Biggest size of the respective body part.

$\mathbf{size_{min}}$ Smallest size of the respective body part.

$\boldsymbol{\tau}$ Vector of commanded torques.

$\boldsymbol{\tau_{ext}}$ Vector of external torques acting on the robot due to $\boldsymbol{f}_{\text{ext}}$.

$\boldsymbol{x}$ Cartesian pose.

$\boldsymbol{x_d}$ Desired Cartesian pose.

$\dot{\boldsymbol{x}}$ Cartesian velocity.

$\ddot{\boldsymbol{x}}$ Cartesian acceleration.

Bibliography

[AASB+06] Rachid Alami, Alin Albu-Schäffer, Antonio Bicchi, Rainer Bischoff, Raja Chatila, Alessandro De Luca, Agostino De Santis, Georges Giralt, Jérémie Guiochet, Gerd Hirzinger, et al. Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1–16. IEEE, 2006.

[ASEG+08] Alin Albu-Schaffer, Oliver Eiberger, Markus Grebenstein, Sami Haddadin, Christian Ott, Thomas Wimbock, Sebastian Wolf, and Gerd Hirzinger. Soft robotics. *IEEE Robotics & Automation Magazine*, 15(3), 2008.

[ASOFH03] Alin Albu-Schaffer, Christian Ott, Udo Frese, and Gerd Hirzinger. Cartesian impedance control of redundant robots: Recent results with the dlr-light-weight-arms. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 3704–3709. IEEE, 2003.

[ASOH07] Alin Albu-Schäffer, Christian Ott, and Gerd Hirzinger. A unified passivity-based control framework for position, torque and impedance control of flexible joint robots. *The International Journal of Robotics Research*, 26(1):23–39, 2007.

[Bar13] Angelos Barmpoutis. Tensor body: Real-time reconstruction of the human body and avatar synthesis from rgb-d. *IEEE Transactions on Cybernetics*, 43(5):1347–1356, 2013.

[BE14] Roland Behrens and Norbert Elkmann. Study on meaningful and verified thresholds for minimizing the consequences of human-robot collisions. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3378–3383. IEEE, 2014.

[BFC14] André Brandao, Leandro AF Fernandes, and Esteban Clua. M5aie a method for body part detection and tracking using rgb-d images. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, volume 1, pages 367–377. IEEE, 2014.

[BK02] Oliver Brock and Oussama Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052, 2002.

[BKS+10] Rainer Bischoff, Johannes Kurth, Günter Schreiber, Ralf Koeppe, Alin Albu-Schäffer, Alexander Beyer, Oliver Eiberger, Sami Haddadin, Andreas Stemmer, Gerhard Grunwald, et al. The kuka-dlr lightweight robot arm-a new reference platform for robotics research and manufacturing. In *Robotics (ISR), 2010 41st international symposium on and 2010 6th German conference on robotics (ROBOTIK)*, pages 1–8. VDE, 2010.

[BT04] Antonio Bicchi and Giovanni Tonietti. Fast and" soft-arm" tactics [robot arm design]. *IEEE Robotics & Automation Magazine*, 11(2):22–33, 2004.

[Caz07] Gilles Cazelais. Linear combination of sine and cosine. Website, 2007. URL: `http://pages.pacificcoast.net/~cazelais/252/lc-trig.pdf`.

[CCS12] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transaction on Visualization and Computer Graphics*, 18(6):914–924, 2012.

[CLMP95] Jonathan D Cohen, Ming C Lin, Dinesh Manocha, and Madhav Ponamgi. I-collide: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 189–ff. ACM, 1995.

[CMF16] Andrea Calanca, Riccardo Muradore, and Paolo Fiorini. A review of algorithms for compliant control of stiff and fixed-compliance robots. *IEEE/ASME Transactions on Mechatronics*, 21(2):613–624, 2016.

[Cra05] John J Craig. *Introduction to robotics: mechanics and control*, volume 3. Pearson Prentice Hall Upper Saddle River, 2005.

[DL00] Alessandro De Luca. Feedforward/feedback laws for the control of flexible robots. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 233–240. IEEE, 2000.

[DLASHH06] Alessandro De Luca, Alin Albu-Schaffer, Sami Haddadin, and Gerd Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1623–1630. IEEE, 2006.

[DLF12] Alessandro De Luca and Fabrizio Flacco. Integrated control for phri: Collision avoidance, detection, reaction and collaboration. In *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*, pages 288–295. IEEE, 2012.

[DLM05] Alessandro De Luca and Raffaella Mattone. Sensorless robot collision detection and hybrid force/motion control. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 999–1004. IEEE, 2005.

[dWSB12] Carlos Canudas de Wit, Bruno Siciliano, and Georges Bastin. *Theory of robot control*. Springer Science & Business Media, 2012.

[EMI17] FRANKA EMIKA. Panda, the sensitive, interconnected lightweight robot for everybody. Website, 2017. URL: `https://www.franka.de/`.

[EMI18a] FRANKA EMIKA. Franka control interface. Website, 2018. URL: `http://support.franka.de/`.

[EMI18b] FRANKA EMIKA. Franka ros. Website, 2018. URL: `https://github.com/frankaemika/franka_ros`.

[FDL17] Flacco Fabrizio and Alessandro De Luca. Real-time computation of distance to dynamic obstacles with multiple depth sensors. *IEEE Robotics and Automation Letters*, 2(1):56–63, 2017.

[FKDLK12] Fabrizio Flacco, Torsten Kröger, Alessandro De Luca, and Oussama Khatib. A depth space approach to human-robot collision avoidance. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 338–345. IEEE, 2012.

[FKDLK15] Fabrizio Flacco, Torsten Kroeger, Alessandro De Luca, and Oussama Khatib. A depth space approach for evaluating distance to objects. *Journal of Intelligent & Robotic Systems*, 80:7, 2015.

[FMR97] Gianni Ferretti, G Magnani, and Paolo Rocco. Toward the implementation of hybrid position/force control in industrial robots. *IEEE Transactions on robotics and automation*, 13(6):838–845, 1997.

[FSF13] Federica Ferraguti, Cristian Secchi, and Cesare Fantuzzi. A tank-based approach to impedance control with variable stiffness. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4948–4953. IEEE, 2013.

[Gar17] Willow Garage. Pr2. Website, 2017. URL: `http://www.willowgarage.com/pages/pr2/design`.

[GFDL13] Milad Geravand, Fabrizio Flacco, and Alessandro De Luca. Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4000–4007. IEEE, 2013.

[Gil91] George T Gilbert. Positive definite matrices and sylvester's criterion. *The American Mathematical Monthly*, 98(1):44–46, 1991.

[GJK88] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.

[GLM96] Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM, 1996.

[GML00] Stefan Gottschalk, Dinesh Manocha, and Ming C Lin. *Collision queries using oriented bounding boxes*. PhD thesis, University of North Carolina at Chapel Hill, 2000.

[GOH15] Saskia Golz, Christian Osendorfer, and Sami Haddadin. Using tactile sensation for learning contact knowledge: Discriminate collision from physical interaction. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3788–3794. IEEE, 2015.

[Had13] Sami Haddadin. *Towards safe robots: approaching Asimov's 1st law*, volume 90. Springer, 2013.

[HASDLH08] Sami Haddadin, Alin Albu-Schaffer, Alessandro De Luca, and Gerd Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3356–3363. IEEE, 2008.

[HASEH10] Sami Haddadin, Alin Albu-Schäffer, Oliver Eiberger, and Gerd Hirzinger. New insights concerning intrinsic joint elasticity for safety. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2181–2187. IEEE, 2010.

[HASF+09a] S. Haddadin, A. Albu-Schaffer, M. Frommberger, J. Rossmann, and G. Hirzinger. The'dlr crash report': Towards a standard crash-testing protocol for robot safety - part ii: Discussions. In *2009 IEEE International Conference on Robotics and Automation*, pages 280–287, May 2009.

[HASF+09b] Sami Haddadin, Alin Albu-Schaffer, Mirko Frommberger, Jurgen Rossmann, and Gerd Hirzinger. The 'dlr crash report': Towards a standard crash-testing protocol for robot safety - part i: Results. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 272–279. IEEE, 2009.

[HASFH08] S. Haddadin, A. Albu-Schaffer, M. Frommberger, and G. Hirzinger. The role of the robot mass and velocity in physical human-robot interaction - part ii: Constrained blunt impacts. In *2008 IEEE International Conference on Robotics and Automation*, pages 1339–1345, May 2008.

[HASH07] Sami Haddadin, Alin Albu-Schäffer, and Gerd Hirzinger. Safety evaluation of physical human-robot interaction via crash-testing. In *Robotics: Science and Systems*, volume 3, pages 217–224, 2007.

[HASH08] Sami Haddadin, Alin Albu-Schaffer, and Gerd Hirzinger. The role of the robot mass and velocity in physical human-robot interaction-part i: Non-constrained blunt impacts. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1331–1338. IEEE, 2008.

[HASH09] Sami Haddadin, Alin Albu-Schäffer, and Gerd Hirzinger. Requirements for safe robots: Measurements, analysis and new insights. *The International Journal of Robotics Research*, 28(11-12):1507–1527, 2009.

[HASH+11] S. Haddadin, A. Albu-Schaffer, F. Haddadin, J. Rosmann, and G. Hirzinger. Study on soft-tissue injury in robotics. *IEEE Robotics Automation Magazine*, 18(4):20–34, Dec 2011.

[HDLAS17] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 2017.

[He99] Taosong He. Fast collision detection using quospo trees. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 55–62. ACM, 1999.

[HHK+12] Sami Haddadin, Simon Haddadin, Augusto Khoury, Tim Rokahr, Sven Parusel, Rainer Burgkart, Antonio Bicchi, and Alin Albu-Schäffer. On making robots understand safety: Embedding injury knowledge into control. *The International Journal of Robotics Research*, 31(13):1578–1602, 2012.

[Hir15] Fabian Hirt. Fast robot-obstacles distance evaluation in the depth space. Bachelor thesis, Technische Universität München, 2015.

[Hog84] Neville Hogan. Impedance control: An approach to manipulation. In *American Control Conference, 1984*, pages 304–313. IEEE, 1984.

[HSAS+02] Gerd Hirzinger, Norbert Sporer, A Albu-Schaffer, M Hahnle, Rainer Krenn, A Pascucci, and Markus Schedl. Dlr's torque-controlled light weight robot iii-are we reaching the technological limits now? In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1710–1716. IEEE, 2002.

[Huy08] Johnny Huynh. Separating axis theorem for oriented bounding boxes. `jkh.me/files/tutorials/Separating%20Axis%20Theorem%20for%20Oriented%20Bounding%20Boxes.pdf`, 2008.

[HZ03] Jochen Heinzmann and Alexander Zelinsky. Quantitative safety guarantees for physical human-robot interaction. *The International Journal of Robotics Research*, 22(7-8):479–504, 2003.

[ISO16] Robots and robotic devices – Collaborative robots. Standard, International Organization for Standardization, Geneva, CH, February 2016.

[Joh70] C. R. Johnson. Positive definite matrices. *The American Mathematical Monthly*, 77(3):259–264, 1970.

[KB16] Klas Kronander and Aude Billard. Stability considerations for variable impedance control. *IEEE Transactions on Robotics*, 32(5):1298–1305, 2016.

[Kha86] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[Kha87] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[Kor15] Johannes Korsawe. Minimal bounding box. Website, 2015. URL: https://de.mathworks.com/matlabcentral/fileexchange/18264-minimal-bounding-box.

[KZB12] Seyed Mohammad Khansari-Zadeh and Aude Billard. A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, 32(4):433–454, 2012.

[Lav05] Eugene Lavretsky. Adaptive control: Cds 270, 2005. Lecture Notes. URL: http://www.cds.caltech.edu/archive/help/uploads/wiki/files/140/CDS270_Lectures_1-3.pdf.

[LB08] K Lee and M Buss. Force tracking impedance control with variable target stiffness. *IFAC Proceedings Volumes*, 41(2):6751–6756, 2008.

[LGLM00] Eric Larsen, Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. Fast distance queries with rectangular swept sphere volumes. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 4, pages 3719–3726. IEEE, 2000.

[LKVS+10] Thomas Lens, Jürgen Kunz, Oskar Von Stryk, Christian Trommer, and Andreas Karguth. Biorob-arm: A quickly deployable and intrinsically safe, light-weight robot arm for service robotics applications. In *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1–6. VDE, 2010.

[LSAD12] Mark A Livingston, Jay Sebastian, Zhuming Ai, and Jonathan W Decker. Performance measurements for the microsoft kinect skeleton. In *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, pages 119–120. IEEE, 2012.

[Luc16] Alessandro De Luca. Lecture slides in robotics 2, 2016.

[MB12] Haifa Mehdi and Olfa Boubaker. Stiffness and impedance control using lyapunov theory for robot-aided rehabilitation. *International Journal of Social Robotics*, 4:107–119, 2012.

[MFDL14] Emanuele Magrini, Fabrizio Flacco, and Alessandro De Luca. Estimation of contact forces using a virtual force sensor. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2126–2133. IEEE, 2014.

[MFDL15] Emanuele Magrini, Fabrizio Flacco, and Alessandro De Luca. Control of generalized contact motion and force in physical human-robot interaction. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2298–2304. IEEE, 2015.

[MMG14] Axel Muttray, Dipl-Psych Michael Melia, and Britta Geißler. Wissenschaftlicher Schlussbericht zum Vorhaben FP-0317: "Kollaborierende Roboter–Ermittlung der Schmerzempfindlichkeit an der Mensch-Maschine-Schnittstelle". 2014.

[MMOL11] Arthur Mattuck, Haynes Miller, Jeremey Orloff, and John Lewis. Under, over and critical damping. MIT OpenCourseWare, 2011. URL: `https://ocw.mit.edu/courses/mathematics/18-03sc-differential-equations-fall-2011/unit-ii-second-order-constant-coefficient-linear-equations/damped-harmonic-oscillators/MIT18_03SCF11_s13_2text.pdf`.

[MPT05] William A McNeely, Kevin D Puterbaugh, and James J Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *ACM SIGGRAPH 2005 Courses*, page 42. ACM, 2005.

[MT15] Alexander Martin Turrillas. Improvement of a multi-body collision computation framework and its application to robot (self-) collision avoidance. Master thesis, Universidad de Navarra, 2015.

[ODAS15] Christian Ott, Alexander Dietrich, and Alin Albu-Schäffer. Prioritized multi-task compliance control of redundant manipulators. *Automatica*, 53:416–423, 2015.

[PAW11] Robert Platt, Muhammad Abdallah, and Charles Wampler. Multiple-priority impedance control. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 6033–6038. IEEE, 2011.

[PCM12] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3859–3866. IEEE, 2012.

[PCW12] Yong-Lae Park, Bor-Rong Chen, and Robert J Wood. Design and fabrication of soft artificial skin using embedded microchannels and liquid conductors. *IEEE Sensors Journal*, 12(8):2711–2718, 2012.

[PW95] Gill A Pratt and Matthew M Williamson. Series elastic actuators. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 399–406. IEEE, 1995.

[QAN11] Morgan Quigley, Alan Asbeck, and Andrew Ng. A low-cost compliant 7-dof robotic manipulator. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 6051–6058. IEEE, 2011.

[Sen07] Luis Sentis. *Synthesis and control of whole-body behaviors in humanoid systems*. Stanford university USA, 2007.

[SH15] Christopher Schindlbeck and Sami Haddadin. Unified passivity-based cartesian force/impedance control for rigid and flexible joint robots via task-energy tanks. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 440–447. IEEE, 2015.

[SHL17] Matteo Saveriano, Fabian Hirt, and Dongheui Lee. Human-aware motion reshaping using dynamical systems. *Pattern Recognition Letters*, 2017.

[SK14] Alex S. Shafer and Mehrdad R. Kermani. Development of high performance intrinsically safe 3-dof robot. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 619–624. IEEE, 2014.

[SK16] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[SS12] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[SSK08] Dongjun Shin, Irene Sardellitti, and Oussama Khatib. A hybrid actuation approach for human-friendly robot design. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1747–1752. IEEE, 2008.

[SSLeS14] Mikel Sagardia, Theodoros Stouraitis, and Joao Lopes e Silva. A new fast and robust collision detection and force computation algorithm applied to the physics engine bullet: Method, integration, and evaluation. In *Conference and Exhibition of the European Association of Virtual and Augmented Reality*, 2014.

[SSP+10] Dongjun Shin, Irene Sardellitti, Yong-Lae Park, Oussama Khatib, and Mark Cutkosky. Design and control of a bio-inspired human-friendly robot. *The International Journal of Robotics Research*, 29(5):571–584, 2010.

[TdVS14] Tadele Shiferaw Tadele, Theo de Vries, and Stefano Stramigioli. The safety of domestic robotics: A survey of various safety-related publications. *IEEE robotics & automation magazine*, 21(3):134–142, 2014.

[Til02] Alvin R Tilley. *The measure of man and woman: human factors in design*, volume 1. John Wiley & Sons, 2002.

[VASB+13] Bram Vanderborght, Alin Albu-Schäffer, Antonio Bicchi, Etienne Burdet, Darwin G Caldwell, Raffaella Carloni, M Catalano, Oliver Eiberger, Werner Friedl, Ganesh Ganesh, et al. Variable impedance actuators: A review. *Robotics and autonomous systems*, 61(12):1601–1614, 2013.

[VDB01] Gino Van Den Bergen. Proximity queries and penetration depth computation on 3d game objects. In *Game developers conference*, volume 170, 2001.

[vdB16] Jur van den Berg. Calculate rotation matrix to align vector a to vector b in 3d? Mathematics Stack Exchange, 2016. URL: `https://math.stackexchange.com/q/476311`.

[VSTH16] Jonathan Vorndamme, Moritz Schappler, Alexander Tödtheide, and Sami Haddadin. Soft robotics for the hydraulic atlas arms: Joint impedance control with collision detection and disturbance compensation. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3360–3367. IEEE, 2016.

[WBVdLS08] Keenan A Wyrobek, Eric H Berger, HF Machiel Van der Loos, and J Kenneth Salisbury. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2165–2170. IEEE, 2008.

[Wei02] Eric W Weisstein. *CRC concise encyclopedia of mathematics*. CRC press, 2002.

[Wel13] René Weller. A brief overview of collision detection. In *New Geometric Data Structures for Collision Detection and Haptics*, pages 9–46. Springer, 2013.

[WZ09] Rene Weller and Gabriel Zachmann. Inner sphere trees for proximity and penetration queries. In *Robotics: science and systems*, volume 2, 2009.

[YHH+97] Yoji Yamada, Yasuhiro Hirasawa, Shengyang Huang, Yoji Umetani, and Kazutsugu Suita. Human-robot contact in the safeguarding space. *IEEE/ASME transactions on mechatronics*, 2(4):230–236, 1997.

[Zha16] Shiyu Zhao. Time derivative of rotation matrices: A tutorial. *arXiv preprint arXiv:1609.06088*, 2016.

[ZRKS04] Michael Zinn, Bernard Roth, Oussama Khatib, and J Kenneth Salisbury. A new actuation approach for human friendly robot design. *The international journal of robotics research*, 23(4-5):379–398, 2004.