Catching a Flying Dart using a Cable-Driven Parallel Robot

DIPLOMA THESIS

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Univ.-Prof. Dr. techn. Andreas Kugi
Dipl.-Ing. Ulrich Knechtelsdorfer BSc
Dipl.-Ing. Michael Schwegel BSc

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by
Georg Feiler
Matriculation number 01525499

Vienna, August 2023

Complex Dynamical Systems Group
A-1040 Wien, Gußhausstr. 27–29, Internet: https://www.acin.tuwien.ac.at

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct - Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, August 2023

Georg Feiler

Preamble

This thesis is the second of two master's theses dealing with the design and development of a cable-driven robot technology demonstrator, which is able to catch a dart thrown by a human such that the dart hits any desired scoring segment on the dartboard. From the conceptualization phase to finishing both theses, the project required about 2 and a half years to complete and I have truly put my heart and soul into it. While many challenges had to be solved, I am very grateful and also a little proud that the design, assembly, development and testing of the robot went relatively smoothly and that the final result meets all requirements and expectations.

The experiences that I gathered while working on this project, but also while pursuing my studies in electrical engineering and mechanical engineering, have been unique and unforgettable. I want to express my deep and honest gratitude to numerous wonderful individuals, that I had the pleasure of meeting, working and studying with, who have shaped my life in many ways. All the generous, compassionate, creative and inspiring people around me have made my journey enjoyable and have empowered me to grow throughout my studies and have enable the successful completion of this project.

I want to thank Prof. Kugi for enabling this project and providing an environment for research that is exciting, inspiring but at the same time meaningful and productive. My deep gratitude belongs to Ulrich Knechtelsdorfer and Michael Schwegel for their tireless support, their open ears and their dedication. I could not have wished for better supervision and guidance. Thank you for your invaluable insights, your open-mindedness and your patience with me. My sincere gratitude also belongs to Gerald Ebmer, Christian Hartl-Nešić and Thomas Weingartshofer for their invaluable and generous help with hardware and software related problems which saved me countless frustrating hours of debugging.

I also want to express my deep gratitude for the limitless love and care from my family. I want to thank my parents Maria and Gerhard for their unwavering support and for never ceasing to believe in me. I want to thank my brother Martin for inspiring me and my lovely girlfriend Luisa for her love and support along the way.

Last but not least, I would like to express my honest gratitude and sincere appreciation to the fellow students that I had the honor of studying with and who have become good friends in the process. Thank you Felix, Benjamin, Elias, Christopher and Jan for all the inspiring conversations, your ideas and your advice.

While the future that lies ahead for my professional life is uncertain, I am looking forward to the challenges and endeavours that the future might hold. I am grateful for all the wonderful humans that I have the honor of travelling alongside on this journey.

Vienna, August 2023

Abstract

In the present work, a cable-driven parallel robot is employed to catch a flying dart thrown by a human. The flight of the dart is captured in real-time by an infrared camera system and a tournament dartboard is moved accordingly, such that the dart hits the desired scoring segment on the dartboard.

An algorithm is developed to estimate the flight trajectory of the dart and predict the impact location. For this purpose, the flight behavior of a dart thrown by an amateur player is studied, and an aerodynamic model for the dart is developed. To move the dartboard accordingly, a real-time trajectory generator is designed. The trajectories generated by the trajectory generator are adapted by an input shaping algorithm, to reduce structural vibrations induced by the dartboard motion due to reaction forces. Finally, a trajectory following controller is designed and studied, which uses a cable force distribution algorithm to resolve kinematic redundancies present in the cable-driven robot.

The individual components and algorithms employed in the control system are validated separately via simulations. Furthermore, experiments are conducted to demonstrate the effectiveness and reliability of the overall dart-catching robot system.

Kurzfassung

In dieser Arbeit wird ein paralleler Seilroboter angesteuert, um einen von einem Menschen geworfenen fliegenden Dartpfeil zu fangen. Der Flug des Pfeils wird in Echtzeit von einem infrarot Kamerasystem erfasst und eine Turnierdartscheibe wird entsprechend bewegt, sodass der Dartpfeil das gewünschte Feld der Dartscheibe trifft.

Aus den während des Flugs gemessenen Positionen des Dartpfeils wird die Flugbahn sowie der Auftreffort des Pfeils geschätzt. Dazu wird das Flugverhalten eines von einem Amateurspieler geworfenen Dartpfeils untersucht und modelliert. Um eine schnelle Bewegung der Dartscheibe zur Zielpose zu ermöglichen, wird ein echtzeitfähiger Trajektoriengenerator entworfen. Die somit berechnete Trajektorie wird in weiterer Folge angepasst, um eine Anregung von Vibrationen des Roboters zu reduzieren. Ein Trajektorienfolgeregler für den Seilroboter wird entworfen und näher untersucht, bei dem ein nichtlinearer Algorithmus zur Berechnung der Seilkraftverteilung zum Einsatz kommt.

Die einzelnen Komponenten des Systems werden mithilfe von Simulationen validiert. Anhand von Experimenten wird die Funktionsfähigkeit und Zuverlässigkeit des Gesamtsystems demonstriert.

Contents

List of Figures

List of Symbols

$\mathbf{x}$	$\in \mathbb{R}^{n_x}$ pose of the end effector.
$\hat{\mathbf{x}}$	$\in \mathbb{R}^{n_x}$ approximate solution for the end effector pose.
$\mathbf{x}_\mathrm{d}$	$\in \mathbb{R}^{n_x}$ trajectory for the EE pose.
$\mathbf{x}_\mathrm{set}$	$\in \mathbb{R}^{n_x}$ target set-point for the EE pose.
x_C	x-coordinate of point C.
y_C	y-coordinate of point C.
φ	rotation angle of the end effector.
n_q	number of cables.
n_x	degrees of freedom (DOF).
n_d	input shaping FIR filter length.
n_mk	number of marker signals in the raw infrared camera data.
r	radius of the end effector.
γ	angle between the cable contact point and the vertical axis.
s	cable winding direction.
l	total cable length.
l_r	rolled cable length.
l_s	spare cable length.
l_S	dart shaft length between markers M and B.
l_B	dart barrel length between markers F and M.
l_T	dart tip length between markers F and the tip T.
l_D	distance from dart center of gravity D to marker F.
l_{ac}	distance from dart center of gravity D to the aerodynamic center ac.
i	$\in \mathbb{N}$ running index, cable index.
i_F	$\in \mathbb{N}$ index for the signal of marker F in the raw marker data.
j	$\in \mathbb{N}$ measurement timestep index.
k	$\in \mathbb{N}$ controller timestep index.
p	value defining a p-norm.
p_i	filter pole.
t	time.
t_j	time corresponding to the measurement step j.
$\mathbf{q}$	$\in \mathbb{R}^{n_q}$ joint vector (cable lengths).
$\boldsymbol{\phi}_\mathrm{ik}$	$\mathbb{R}^{n_x} \to \mathbb{R}^{n_q}$ mapping defining the inverse kinematic transformation.
$\boldsymbol{\phi}_\mathrm{fk}$	$\mathbb{R}^{n_q} \to \mathbb{R}^{n_q}$ mapping defining the forward kinematic transformation.
$\hat{\boldsymbol{\phi}}_\mathrm{fk}$	$\mathbb{R}^{n}_{q} \to \mathbb{R}^{n}_{x}$ approximation of $\boldsymbol{\phi}_\mathrm{fk}$.
$\mathbf{J}_\mathrm{ik}$	$\in \mathbb{R}^{n_q \times n_x}$ Jacobian matrix of the inverse kinematics.
$\mathbf{J}_\mathrm{ik}^\mathrm{T+}$	$\in \mathbb{R}^{n_q \times n_x}$ pseudoinverse of $\mathbf{J}_\mathrm{ik}^\mathrm{T}$.

$\mathbf{r}_{AB}$	$\in \mathbb{R}^3$ vector pointing from point B to point A.
$\mathbf{r}_C$	$\in \mathbb{R}^3$ position vector pointing to point C.
$\mathbf{e}_{AB}$	$\in \mathbb{R}^3$ unit vector in direction from point B to point A.
$\mathbf{e}_r$	$\in \mathbb{R}^3$ radial basis vector for polar coordinates.
$\mathbf{e}_\varphi$	$\in \mathbb{R}^3$ azimuthal basis vector for polar coordinates.
$\mathbf{e}_\perp$	$\in \mathbb{R}^2$ normal basis vector in the xy-plane.
$\mathbf{e}_\mathrm{x}$	$\in \mathbb{R}^{n_x}$ trajectory following error.
$\mathbf{d}$	$\in \mathbb{R}^{n_x}$ wrench direction.
h_A	offset between winch and groove-plane in z direction.
h_C	offset between point C and groove-plane in z direction.
F_x	force on the end effector in x-direction.
F_{fi}	aerodynamic lift force on the dart.
F_{di}	aerodynamic damping force on the dart.
$\mathbf{F}$	$\mathbb{R}^{10} \times \mathbb{R} \times \mathbb{R}^{10} \times \mathbb{R} \to \mathbb{R}^{10}$ discrete-time transition function.
M_x	torque on the end effector around x-axis.
τ	cable force.
$\boldsymbol{\tau}$	$\in \mathbb{R}^{n_q}$ vector of cable forces.
$\boldsymbol{\tau}_\mathrm{min}$	$\in \mathbb{R}^{n_q}$ vector of minimum pretension cable forces.
$\boldsymbol{\tau}_\mathrm{max}$	$\in \mathbb{R}^{n_q}$ vector of maximum cable forces.
$\boldsymbol{\tau}_\mathrm{d}$	$\in \mathbb{R}^{n_q}$ vector of desired cable forces.
$\boldsymbol{\tau}_\Delta$	$\in \mathbb{R}^{n_q}$ deviation from the desired cable forces.
$\boldsymbol{\tau}^*$	$\mathbb{R}^{n_x} \to \mathbb{R}^n_q$ cable force distribution function.
$\mathbf{f}$	$\in \mathbb{R}^{n_x}$ vector of generalized forces (wrench) acting on the end effector.
$\mathbf{f}_\mathrm{d}$	$\in \mathbb{R}^{n_x}$ desired wrench vector on the end effector.
$\mathbf{f}(\mathbf{z}, u)$	$\mathbb{R}^{10} \times \mathbb{R} \to \mathbb{R}^{10}$ dynamical model for the dart.
f_s	sampling frequency of the robot controller.
f_0	undamped natural frequency in Hz.
f_n	notch filter frequency in Hz.
Δf	notch filter bandwidth in Hz.
T_s	sample time of the robot controller.
T_i	motor torque.
$\mathbf{T}$	$\in \mathbb{R}^{n_q}$ vector of motor torques.
$\mathbf{T}_\mathrm{d}$	$\in \mathbb{R}^{n_q}$ vector of desired motor torques.
$\mathbf{T}_\mathrm{w}$	$\in \mathbb{R}^{n_q}$ vector of winch inertia compensation torques.
ξ	motor angle.
$\boldsymbol{\xi}$	vector of motor angles.
$\mathbf{u}$	motor voltage space vector.
u	switching variable for the dart model.
$\mathbf{i}$	motor current space vector.
ν_w	transmission ratio of the winch mechanism.
$\mathbf{M}$	$\in \mathbb{R}^{n_x \times n_x}$ mass matrix.
$\mathbf{M}_\mathrm{EE}$	$\in \mathbb{R}^{n_x \times n_x}$ end effector mass matrix.
$\mathbf{C}$	$\in \mathbb{R}^{n_x \times n_x}$ Coriolis matrix.

$\mathbf{\Lambda}$	$\in \mathbb{R}^{n_x \times n_q}$ torque transmission matrix.
g	gravitational acceleration.
$\mathbf{g}$	$\in \mathbb{R}^{n_x}$ generalized gravity vector.
m_{EE}	mass of the end effector.
m_D	mass of the dart.
I_{w}	mass moment of inertia of the winch system..
$I_{z,\mathrm{EE}}$	mass moment of inertia of the EE around the z-axis.
$I_{D,\perp}$	mass moment of inertia of the dart perpendicular to the z_D-axis.
$I_{D,\parallel}$	mass moment of inertia of the dart around to the z_D-axis.
$\mathbf{K}_{\mathrm{P}}$	$\in \mathbb{R}^{n_x \times n_x}$ proportional controller gain matrix.
$\mathbf{K}_{\mathrm{D}}$	$\in \mathbb{R}^{n_x \times n_x}$ derivative controller gain matrix.
$\mathbf{K}$	$\in \mathbb{R}^{10 \times 9}$ Kalman gain matrix.
K	constant in the input shaping filter.
$\mathbf{r}_{\mathrm{d}}$	$\in \mathbb{R}^2$ trajectory for the EE position.
$\hat{\mathbf{r}}_{\mathrm{d}}$	$\in \mathbb{R}^2$ unfiltered position computed by the trajectory generator.
$\mathbf{r}_{\mathrm{set}}$	$\in \mathbb{R}^2$ target set-point for the EE position.
$\mathbf{r}^*$	$\in \mathbb{R}^2$ time optimal trajectory for the EE position.
$\mathbf{r}_{\mathrm{p}}$	$\in \mathbb{R}^2$ dart impact position in the xy-plane.
$\mathbf{v}_{\mathrm{d}}$	$\in \mathbb{R}^2$ trajectory for the EE velocity.
$\hat{\mathbf{v}}_{\mathrm{d}}$	$\in \mathbb{R}^2$ unfiltered acceleration computed by the trajectory generator.
$\mathbf{v}_0$	$\in \mathbb{R}^2$ initial velocity of the trajectory.
$\mathbf{v}^*$	$\in \mathbb{R}^2$ time optimal trajectory for the EE velocity.
$\mathbf{a}_{\mathrm{d}}$	$\in \mathbb{R}^2$ trajectory for the EE acceleration.
$\hat{\mathbf{a}}_{\mathrm{d}}$	$\in \mathbb{R}^2$ unfiltered acceleration computed by the trajectory generator.
$\mathbf{a}^*$	$\in \mathbb{R}^2$ time optimal trajectory for the EE acceleration.
$\mathbf{j}_{\mathrm{d}}$	$\in \mathbb{R}^2$ trajectory for the EE jerk.
j_{step}	resulting jerk for a step input.
φ_{d}	trajectory for the EE rotation angle.
φ_{set}	target set-point for the EE rotation angle.
ω_{d}	trajectory for the angular velocity of the EE.
$\omega_{\max}$	angular velocity limit for the EE.
$\dot{\omega}_{\max}$	angular acceleration limit for the EE.
$\dot{\omega}_n$	notch filter angular frequency in rad/s.
ω_0	undamped natural angular frequency in rad/s.
ω_1	angular velocity in the dart reference frame around x_D-axis.
ω_2	angular velocity in the dart reference frame around y_D-axis.
$v_{\max}$	velocity limit for the EE.
$a_{\max}$	acceleration limit for the EE.
$\mathbf{G}$	$\mathbb{R}^2 \times \mathbb{R}^2 \times \cdots \to \mathbb{R}^2$ mapping defining a filtering function.
v_r	radial component of the vector $\mathbf{v}$.
$v_\perp$	normal component of the vector $\mathbf{v}$.
$\hat{l}_r$	distance from the trajectory to the set-point.
Ω	angular velocity in the local reference frame.

Ω_n	normalized notch frequency in rad/sample.
α_1, α_2	damping factors for the trajectory generator.
$t_\perp$	time to complete the velocity profile in normal direction.
$t_{r,1}$	time to complete the velocity profile in case Ⓘ.
$\tilde{t}_{r,1}$	lower bound for the time $t_{r,1}$.
N_{FIR}	length of the moving average FIR filter.
c_i	constant in the IIR filter transfer function.
C_i	constant for the dynamic dart model.
n	IIR filter order.
β	coefficient for the notch filter.
$\mathbf{H}$	$\in \mathbb{R}^{2\times2}$ impulse response matrix of the FIR filter.
H_n	notch filter discrete-time transfer function.
$\mathbf{I}$	identity matrix.
$\mathbf{I}_D$	inertia tensor in the dart reference frame.
$G(s)$	IIR filter transfer function.
α_0	initial angle of the trajectory.
t_{opt}	optimal (minimal) duration of the trajectory.
t_{OTG}	duration of the trajectory generated by the proposed OTG algorithm.
Δt_{rel}	relative excess time required by the OTG algorithm.
A	amplitude of the structural vibration.
ζ	damping ratio of the structural vibration.
$\mathbf{Y}$	raw measurement data from the infrared (IR) camera system.
$\mathbf{y}$	$\in \mathbb{R}^9$ marker measurement data.
$\hat{\mathbf{y}}$	$\in \mathbb{R}^9$ a-priori estimate of the marker measurement data.
$\mathbf{z}$	$\in \mathbb{R}^{10}$ kinematic state of the dart.
$\hat{\mathbf{z}}$	$\in \mathbb{R}^{10}$ a-priori estimate of the kinematic state of the dart.
$\hat{\mathbf{z}}^+$	$\in \mathbb{R}^{10}$ a-posteriori estimate of the kinematic state of the dart.
$\mathbf{h}$	$\mathbb{R}^{10} \to \mathbb{R}^9$ output function mapping a dart state to marker positions.
$\mathbf{w}$	$\mathbb{R}^{10}$ process noise/disturbance.
$\mathbf{v}$	$\mathbb{R}^9$ measurement noise/disturbance.
ϑ	angle parameterizing the orientation of the dart.
ψ	angle parameterizing the orientation of the dart.
$\mathbf{R}^D$	$\in SO(3)$ rotation matrix from the dart ref. frame to the inertial ref. frame.
$\mathbf{R}$	$\in \mathbb{R}^{9\times9}$ measurement noise covariance matrix.
$\mathbf{Q}$	$\in \mathbb{Q}^{10\times10}$ process noise covariance matrix.
$\mathbf{P}$	$\in \mathbb{R}^{10\times10}$ a-priori estimate of the estimation error covariance matrix.
$\mathbf{P}^+$	$\in \mathbb{R}^{10\times10}$ a-posteriori estimate of the estimation error covariance matrix.
$\mathbf{\Phi}$	$\in \mathbb{R}^{10\times10}$ transition matrix of the linearized system.
$\mathbf{C}$	$\in \mathbb{R}^{9\times10}$ output matrix of the linearized system.
ac	aerodynamic center of the dart.
$\mathrm{E}(\cdot)$	expected value operator.
$\delta_{j,k}$	Kronecker symbol.

1 Introduction

Sports and competitions have always been great activities for testing skills, training abilities and pushing the boundaries of what is physically possible while at the same time bringing enjoyment and entertainment. While humans participate in sports since ancient times [1], sports and competitions have been widely used across many different disciplines of technology to demonstrate capabilities, train and improve the state-of-the-art and develop new solutions for all kinds of problems. Sports and competitions are a useful tool in the world of robotics to improve robot designs, algorithms and materials.

Many different sports have been used as benchmarks for robotic applications because they usually require fast, precise and skillful motion. Among them are badminton [2], tennis [3], archery [4], soccer [5] and ping-pong [6–9]. The sport of darts is particularly interesting for robotic applications because it can be played indoors with a very small amount of space and requires great levels of speed as well as precision. Furthermore, it is very popular and commonly played all around the world. Consequently, the skills and processes involved are easy to understand and communicate. This makes darts very suitable for demonstrating the performance and capabilities of a robotic system.

In this work, a robot capable of catching a flying dart thrown by a human is developed. The thesis builds upon the preceding work [10], where a cable-driven parallel robot (CDPR) is developed and implemented which is capable of highly dynamic motion. The end effector (EE) of the CDPR is equipped with a tournament dartboard. Figure 1.1 shows a photograph of the CDPR prototype.

An infrared (IR) camera system is used to track the position of a steel dart. In the present work, the algorithms and components for tracking the flight-trajectory of the dart, predicting the dart's impact location and quickly moving the dartboard accordingly are developed. The resulting control system is capable of automatically catching a flying dart which is thrown at the dartboard, such that any desired scoring segment can be hit. Thus, the aim of this thesis is to create a technology demonstrator for showing the performance and precision of the CDPR as well as the employed algorithms and control strategy.

1.1 Literature review

In the literature, different aspects of the sport of darts have been used in the context of robotics. In [11], the dart throwing motion is used as a benchmark application for a hybrid variable stiffness actuator. The throwing motion itself is modelled and generalized using reinforcement learning techniques in [12]. An assistive robotic trainer is presented in [13], which gently guides a human thrower using a robotic actuator attached to the human upper limb when throwing a dart.

In his master's thesis [14], Linderoth presents a vision-based algorithm for tracking the motion of a flying dart and predicting the dart's impact location. The computer vision

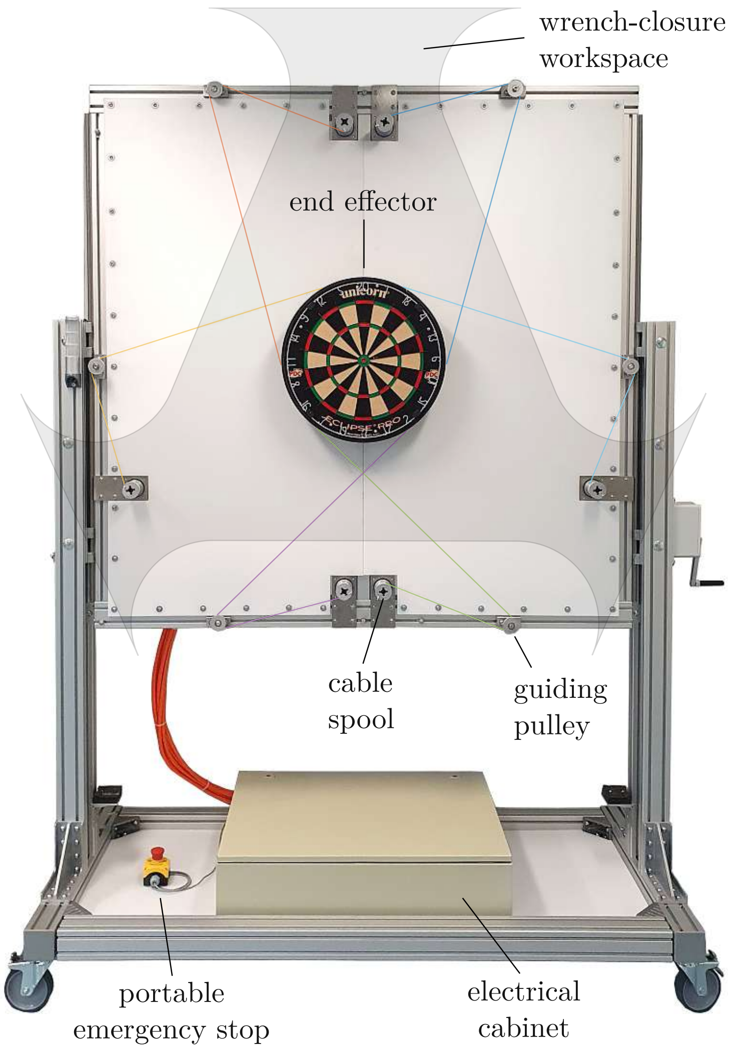


Figure 1.1: Front view of the robot prototype developed in [10]. The white cables are colored for better visibility.

system is based on RGB cameras and the flight trajectory of the dart is tracked using a Kalman filter.

The dart catching robot *Mi5-Dartboard* [15], which not only tracks the flight of the dart but also moves the dartboard accordingly, is implemented by a joint team from TU-Munich and ITQ GmbH as a technology demonstrator. The system uses RGB cameras and a specialized pulsed lighting system to measure and track the position of the dart and a gantry composed of industrial linear axes to move the dartboard.

Wolfslehner presents a high-speed hydraulic positioning system for a dartboard in his master's thesis [16]. Based on this system, the dart catching robot *Magic Darts* [17] is implemented by a joint team from JKU-Linz, INRAS GmbH and the Linz Center of Mechatronics (LCM). The system uses a radar-based measurement system for tracking the position of the steel dart.

Rober designs and implements a dart catching robot [18], based on a CDPR for moving the dartboard and a commercial IR camera system for tracking the dart. The CDPR for moving the dartboard is made from low-cost components and is used for entertainment and educational purposes. The system implemented by Rober is similar to the system presented in the present thesis and was used as inspiration.

1.2 Overview

The contents of this thesis encompass the control system required to catch a flying dart, such that the dart hits the desired scoring segment on the dartboard. Figure 1.2 provides an overview of the developed control system structure. The robot controller lies at the heart of the resulting control system and is implemented on a real-time computer system. All algorithms developed in this work are encompassed by the robot controller which essentially coordinates all the necessary steps for catching the dart. In the following, the interaction of the different elements and subsystems depicted in Figure 1.2 is explained, to provide an overview of the developed control system architecture.

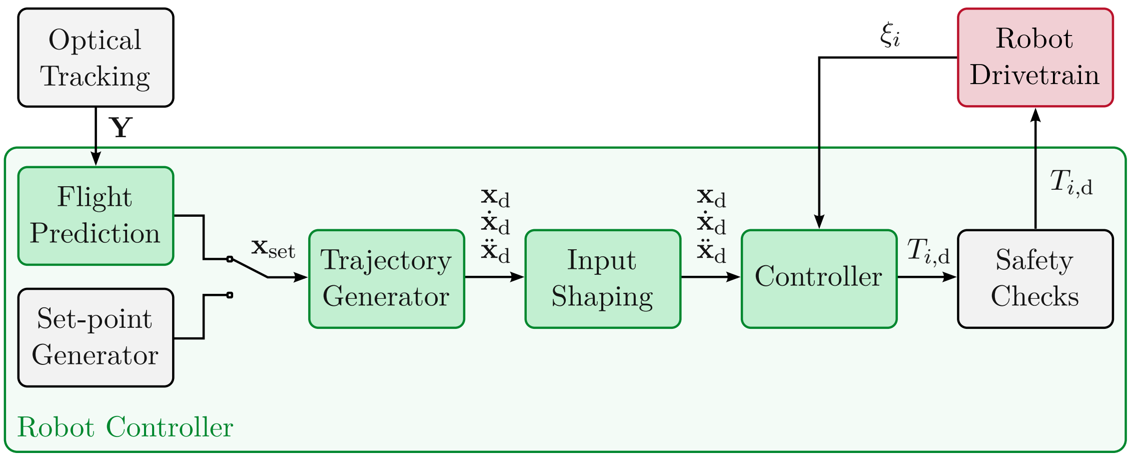

Optical Tracking
$\mathbf{Y}$
Flight Prediction
Set-point Generator
$\mathbf{x}_{\mathrm{set}}$
Trajectory Generator
$\mathbf{x}_{\mathrm{d}}$ $\dot{\mathbf{x}}_{\mathrm{d}}$ $\ddot{\mathbf{x}}_{\mathrm{d}}$
Input Shaping
$\mathbf{x}_{\mathrm{d}}$ $\dot{\mathbf{x}}_{\mathrm{d}}$ $\ddot{\mathbf{x}}_{\mathrm{d}}$
Controller
$T_{i,\mathrm{d}}$
Safety Checks
$T_{i,\mathrm{d}}$
Robot Drivetrain
ξ_i
Robot Controller

Figure 1.2: Overview over the control system structure.

The Optical Tracking system measures the spatial coordinates of three markers attached to the dart and sends these measurements $\mathbf{Y}$ to the robot controller. In the first step, the raw marker measurements are processed by the Flight Prediction algorithm, which tracks the flight trajectory of the dart and computes the dart's impact location on the board. A suitable pose $\mathbf{x}_{\mathrm{set}}$ for catching the dart is calculated, which is used as a set-point for the target pose of the EE. Alternatively, to execute predefined motions for testing and demonstration purposes instead of catching a dart, a sequence of poses can be generated by the Set-point Generator.

The Trajectory Generator algorithm uses the current set-point value to generate a trajectory for the EE pose, velocity and acceleration $\mathbf{x}_{\mathrm{d}}, \dot{\mathbf{x}}_{\mathrm{d}}, \ddot{\mathbf{x}}_{\mathrm{d}}$. The created trajectory adheres to predefined dynamical limits and hence this stage acts as a filter, which ensures that the motion is sufficiently smooth and feasible within the dynamical capabilities of the robot.

To minimize mechanical vibrations in the structural components of the robot, the Input Shaping stage can optionally apply additional corrections to the trajectory and shape it in a way, such that the dominant natural frequency of the robot frame is not excited by the EE motion. Thus, unwanted oscillations are reduced.

The trajectory tracking Controller uses the desired trajectory $\mathbf{x}_\mathrm{d}, \dot{\mathbf{x}}_\mathrm{d}, \ddot{\mathbf{x}}_\mathrm{d}$ and the feedback provided by the motor angles ξ_i to calculate suitable motor torque values $T_{i,\mathrm{d}}$ for each servomotor. The control algorithm not only has to ensure that the EE tracks the trajectory as closely as possible but also that the cable forces remain within their limits. Because cables can only transmit tensile forces, special care must be taken that the cables always remain under tension and never become slack.

Finally, an additional stage of Safety Checks is applied to the motor torques $T_{i,\mathrm{d}}$ to ensure that the values are admissible before forwarding them to the Robot Drivetrain.

The servomotor controllers inside the drivetrain operate in a cascaded torque control and regulate the motor voltages and motor currents, such that the desired torques are applied to the winches of the CDPR.

The whole process is repeated iteratively as long as the dart is in flight. Thus, the system corrects uncertainties and disturbances and continuously adjusts the motion of the EE as the dart approaches the dartboard.

While the optical tracking system, the robot controller and the servomotor controllers operate as digital discrete time systems, they use different sample times and are not synchronized. The IR camera system supports a maximum frame rate of 360 Hz and transmits the most recent measurement data $\mathbf{Y}$ to the robot controller as soon as they are available. The robot controller operates at a frequency of 8 kHz and the cascaded servomotor controllers use a sampling frequency of 32 kHz.

1.3 Structure of the thesis

The structure of this thesis is based on the overview provided in Section 1.2 and visualized in Figure 1.2. The individual components and elements are discussed in reverse order starting with the robot hardware and concluding with the optical tracking of the dart.

First, in Chapter 2, a control-oriented model of the robot is presented which is the foundation of the control strategy. The kinematics of the CDPR are briefly outlined in Section 2.1 and in the following the dynamics are modelled in Section 2.2.

Based on the mathematical model, a control strategy is developed in Chapter 3. The problem of distributing the cable forces, such that sufficient tension is ensured in all cables is discussed in Section 3.1. Based on the solution of the force distribution problem, a trajectory tracking controller is designed in Section 3.2.

Next, a trajectory generation strategy is developed in Chapter 4. Existing approaches from the literature are examined in Section 4.2 and a trajectory generation algorithm is designed in Section 4.3. Simulations demonstrating the effectiveness of the algorithm are provided in Section 4.4.

Chapter 5 revolves around input shaping methods for vibration reduction. Here, different input shaping techniques are discussed and compared.

A flight prediction algorithm for tracking the dart and predicting the dart's impact location is developed in Chapter 6. To introduce and outline the developed approach, an overview of the steps involved in the algorithm is provided in Section 6.1. The mathematical model used for describing the dart flight is explained in Section 6.2. Based on this model, an observer for estimating the state of the dart is designed in Section 6.4. The procedure for predicting the impact location from the dart state is outlined in Section 6.5.

In Chapter 7, the developed prototype is used to validate the individual components and to demonstrate the effectiveness of the entire system. The experimental setup is presented in Section 7.1 and the hardware components are briefly introduced. The trajectory tracking error achieved by the controller is examined in Section 7.2. In Section 7.3 the vibration reduction achieved by applying different input shaping methods is assessed experimentally. The dart tracking and impact prediction performance is examined and validated in Section 7.4. Finally, the reliability of the entire system is tested by catching flying darts thrown by different amateur players such that they hit different segments on the dartboard as desired in Section 7.5.

The work is concluded in Chapter 8 with a summary of the developed components and the findings obtained during the designing the system. Furthermore, an outlook on possible future improvements and extensions is provided.

2 Mathematical robot model

The mathematical robot model is the foundation for the subsequent development of a controller. The robot used for positioning the dartboard was developed in the preceding master's thesis [10], where the CDPR concept is presented and studied. More information and a detailed derivation of the kinematics and the dynamic model can be found therein. A more general and in-depth discussion on kinematics and dynamic descriptions of cable driven robots can be found in the book [19].

In Section 2.1, the kinematics of the robot used for positioning the dartboard are briefly reviewed and the essential relations are established. Subsequently, in Section 2.2, the equations of motion are presented.

2.1 Kinematics

The kinematics of a robotic system studies the motion of different parts of the robot using the geometry and couplings of the individual components of the robot. To describe a pose $\mathbf{x} \in \mathbb{R}^{n_x}$ of the planar manipulator, the Cartesian coordinates of the EE center-point x_C and y_C and the EE rotation angle φ are used in the form

$$\mathbf{x} = \begin{bmatrix} x_C & y_C & \varphi \end{bmatrix}^{\mathrm{T}} . \tag{2.1}$$

Thus, the robot has $n_x = 3$ degrees of freedom, two of which are translational and one is rotational.

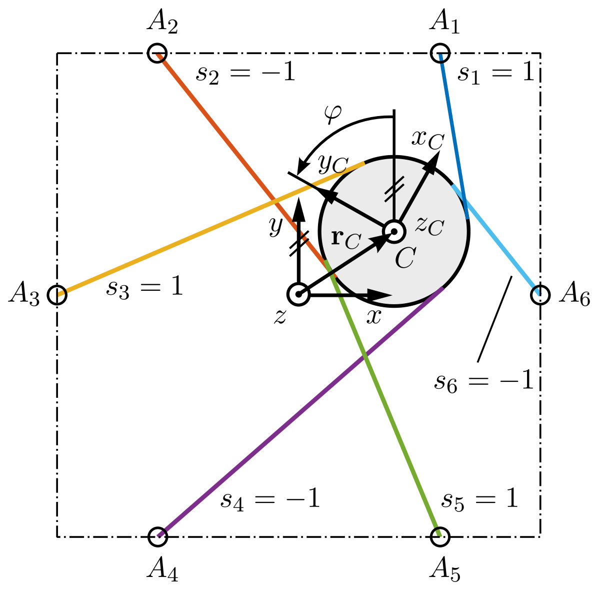
To actuate the CDPR, $n_q = 6$ cables are utilized. The individual cable lengths l_i with $i = 1, \dots, 6$ are actuated by electric motors and represent the joint coordinates. The vector of joint coordinates $\mathbf{q} \in \mathbb{R}^{n_q}$ is obtained by collecting the cable lengths in the form

$$\mathbf{q} = \begin{bmatrix} l_1 & l_2 & \dots & l_{n_q} \end{bmatrix}^{\mathrm{T}} . \tag{2.2}$$

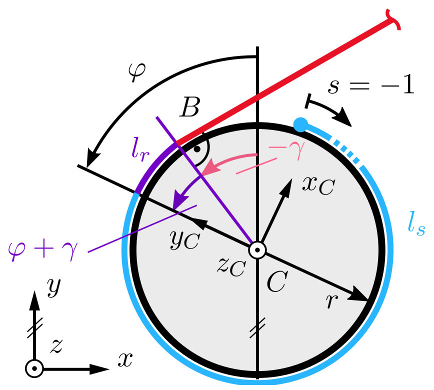
For a CDPR, a number of $n_q = n_x + 1$ cables is required to completely restrain the robot and ensure tension in all cables without relying on external forces such as gravity. For the present robot, there are additional redundancies since $n_q > n_x + 1$ and hence the CDPR is called *redundantly restrained*, see, e. g., [19].

Figure 2.1 provides an overview of the CDPR structure and geometry. The cable geometry in the xy-plane is visualized in Figure 2.1(a). Each cable is fixed to the EE and then wrapped (possibly more than once) around a circular fixture attached to the EE in a groove. The cable detaches the circular EE tangentially at the contact point B and spans to the winch anchor-point A. The cable wrapped around the EE is depicted in Figure 2.1(b). Cables can be either wound in clockwise (CW) or counterclockwise (CCW) direction which is indicated by the wrapping direction $s = -1$ or $s = 1$, respectively, as labelled in Figure 2.1(a).

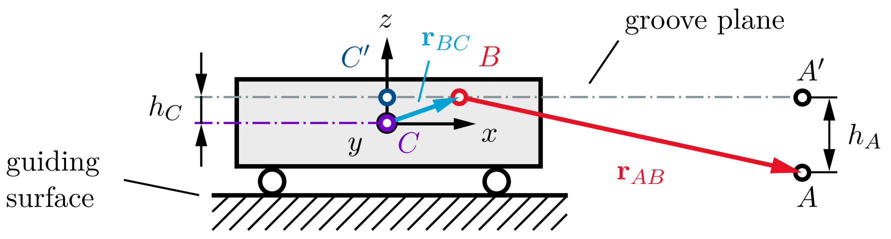
To prevent collisions between the cables, the grooves are offset in z-direction by the height h_C as shown in Figure 2.1(c). To block any motion of the EE in z-direction and prevent tilting around the x- and y-axis, the EE slides along a guiding surface. Stable contact with the guiding surface is critical to achieve reliable guidance. For this purpose, each cable anchor-point A is offset in negative z-direction by the height h_A with respect to the corresponding groove plane as illustrated in Figure 2.1(c).



(a) Cable geometry in the xy-plane.



(b) Cable wrapped around the EE.



(c) Cable geometry in the xz-plane.

Figure 2.1: Robot geometry.

2.1.1 Inverse kinematics

In the so-called inverse kinematics problem, the EE pose $\mathbf{x}$ is known and the cable lengths i. e. joint coordinates $\mathbf{q}$ are sought. Hence, the inverse kinematics problem can be formulated in the form of the so-called inverse kinematic transformation function

$\boldsymbol{\phi}_{\mathrm{ik}} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_q}$ which can be written in the form

$$\mathbf{q} = \boldsymbol{\phi}_{\mathrm{ik}}(\mathbf{x}) \ . \tag{2.3}$$

To solve the inverse kinematics problem, the vectors $\mathbf{r}_{AB}$ and $\mathbf{r}_{BC}$, depicted in Figure 2.1(c), can be calculated from the robot geometry and the EE position x_C and y_C as shown in [10]. The cable length in contact with the EE can be partitioned into the spare length l_s and the rolled cable length l_r as depicted in Figure 2.1(b). The spare cable length l_s is a design parameter and can be chosen freely. The pose-dependent length l_r can be calculated using the relation

$$l_{\mathrm{r}} = -sr(\varphi + \gamma) \ , \tag{2.4}$$

where the angle γ is obtained from the relation

$$\gamma = \mathrm{atan2}\Big(\mathbf{r}_{BC}^{\mathrm{T}}\mathbf{e}_x, \mathbf{r}_{BC}^{\mathrm{T}}\mathbf{e}_y\Big) \ , \tag{2.5}$$

utilizing the four-quadrant arc-tangent function [20]. Finally, the total cable length for each cable is obtained in the form

$$l = \|\mathbf{r}_{AB}\|_2 + l_r + l_s \ . \tag{2.6}$$

Repeating this computation for all n_q cables yields the inverse kinematics transformation function $\boldsymbol{\phi}_{\mathrm{ik}}(\mathbf{x})$ from (2.3). The inverse kinematic transformation is straightforward, computationally inexpensive to evaluate and always has a unique solution for physically feasible poses. Infeasible poses, where one of the winches lies inside the EE disk yield conjugated complex solutions which are not physically relevant.

2.1.2 Forward kinematics

In the so-called forward- or direct kinematic problem, the joint coordinates $\mathbf{q}$ are available and the EE pose $\mathbf{x}$ is to be determined. Similarly to the inverse kinematics, the forward kinematic problem can be formulated using the forward kinematic transformation function $\boldsymbol{\phi}_{\mathrm{fk}} : \mathbb{R}^{n_q} \to \mathbb{R}^{n_x}$ which can be written in the form

$$\mathbf{x} = \boldsymbol{\phi}_{\mathrm{fk}}(\mathbf{q}) \ . \tag{2.7}$$

In contrast to the inverse kinematics problem, the forward kinematic problem is over-determined because the robot is kinematically over-constrained since $n_q > n_x$. For parallel manipulators such as the present CDPR, the forward kinematics problem is in general much more complicated than the inverse kinematics problem while the opposite holds true for serial kinematic robots [21].

In the case of CDPRs, it can be shown that the inverse kinematic transformation function always has a unique solution while the forward kinematic transformation function might have one, multiple or infinitely many solutions for given joint coordinates $\mathbf{q}$ as a result of the over constrained nature of this robot type [19]. In a real robot implementation, an exact solution for the forward kinematics transformation usually does not exist due to the presence of measurement errors which causes the measured joint coordinates $\mathbf{q}$ to be infeasible.

In the preceding master's thesis [10], an approximate solution algorithm in the form

$$\hat{\mathbf{x}} = \hat{\boldsymbol{\phi}}_{\mathrm{fk}}(\mathbf{q}) \tag{2.8}$$

is developed, which is used in the present work. The pose $\hat{\mathbf{x}}$ denotes the approximate solution for the forward kinematic transformation function given any joint vector $\mathbf{q}$. The algorithm formulates the forward kinematic problem as the intersection of modified involute curves which is computed by numerically solving a nonlinear least-squares problem. The algorithm is robust with respect to noise and converges reliably within the whole reachable workspace of the CDPR prototype as demonstrated in [10].

2.1.3 Jacobian matrix

To find a relation between the joint space velocity $\dot{\mathbf{q}}$ and the task space velocity $\mathbf{x}$, Equation (2.3) can be differentiated with respect to time. This yields

$$\begin{aligned} \frac{\mathrm{d}\mathbf{q}(t)}{\mathrm{d}t} &= \frac{\mathrm{d}\boldsymbol{\phi}_{\mathrm{ik}}(\mathbf{x}(t))}{\mathrm{d}t} = \underbrace{\frac{\partial \boldsymbol{\phi}_{\mathrm{ik}}}{\partial \mathbf{x}}}_{\mathbf{J}_{\mathrm{ik}}(\mathbf{x})} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \\ \dot{\mathbf{q}} &= \mathbf{J}_{\mathrm{ik}}\dot{\mathbf{x}} \,, \end{aligned} \tag{2.9}$$

with the Jacobian matrix of the inverse kinematics $\mathbf{J}_{\mathrm{ik}}(\mathbf{x}) \in \mathbb{R}^{n_q \times n_x}$. Note, that $\mathbf{J}_{\mathrm{ik}}$ is rectangular and thus cannot be inverted. Due to the kinematically over-constrained design of the present CDPR, the joint coordinates $\mathbf{q}$ and the joint velocities $\dot{\mathbf{q}}$ must be compliant with the kinematic constraints and cannot be chosen arbitrarily.

The collection of forces and torques acting on the EE in the task space is commonly referred to as *wrench* in the literature [19, 22]. For the employed CDPR, the task space wrench $\mathbf{f}$ is defined as

$$\mathbf{f} = \begin{bmatrix} F_x & F_y & M_z \end{bmatrix}^{\mathrm{T}} \,, \tag{2.10}$$

where F_x and F_y denote the forces in x- and y-direction, respectively, and M_z denotes the torque around the z-axis.

The cable force vector $\boldsymbol{\tau} \in (\mathbb{R}^+)^{n_q}$ collects the individual cable forces τ_i and reads as $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_{n_q}]^{\mathrm{T}}$. Note, that the cable forces are required to be positive due to the constraint that cables can only transmit pulling forces and the convention that pulling forces are considered positive.

The relation between cable forces and task space wrench is given by the transpose of the Jacobian of the inverse kinematics in the form

$$\mathbf{f} = -\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}\boldsymbol{\tau} \,, \tag{2.11}$$

as a result of the duality of velocities and forces, see [10, 19]. The minus sign in Equation (2.11) is a result of the convention that pulling forces are considered positive as elaborated in more detail in [10].

2.2 Dynamics

The CDPR is mathematically modelled and a control oriented dynamical model of the robot is derived in the preceding thesis [10]. In this section, the governing equations of motion are briefly presented and the essential relations which are important for developing a control concept are discussed.

To analyze the dynamics of the CDPR, all moving masses contained in the system must be considered. The robot consists of the EE and the winch mechanisms which are connected via cables. The EE has the mass m_{EE} and the moment of inertia with respect to the z-axis $I_{\mathrm{EE},z}$. Each winch mechanism can be described using the moment of inertia I_{w} with respect to the motor axis, which accounts for the inertia of all rotating parts inside the winch mechanism. For the sake of simplicity, it is assumed that all winches are identical and consequently possess the same inertia. The motor angle of each winch is denoted ξ_i and all motor angles are collected in the vector of motor angles $\boldsymbol{\xi}$. Similarly, the motor torques are denoted T_i and collected in the vector of motor torques $\mathbf{T}$. Due to their low mass in comparison to the other components of the robot, the cables are assumed to be massless.

The CDPR is a multibody system which can be converted to single body subsystems by separating the winches and the EE at the cables. Figure 2.2 depicts the resulting free-body diagram for the winches and the EE.

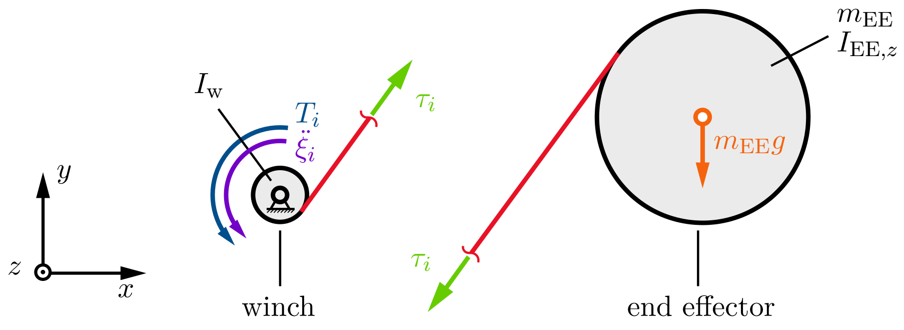


Figure 2.2: Free body diagram, see [10].

The winches convert the rotary motion of the servomotors to a linear motion i. e. a change in cable lengths. Thus, the angular motor velocities $\dot{\boldsymbol{\xi}}$ are related to the cable velocities $\dot{\mathbf{q}}$ in the form

$$\dot{\mathbf{q}} = \nu_{\mathrm{w}} \dot{\boldsymbol{\xi}} \;, \tag{2.12}$$

where ν_{w} denotes the so-called winch transmission ratio.

The balance of angular momentum for the winch subsystem reads as

$$I_{\mathrm{w}} \ddot{\boldsymbol{\xi}} = \mathbf{T} + \nu_{\mathrm{w}} \boldsymbol{\tau} \;, \tag{2.13}$$

Similarly, the balance of momentum for the EE can be written in the form

$$\mathbf{M}_{\mathrm{EE}} \ddot{\mathbf{x}} = \mathbf{f} - \mathbf{g} = -\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}} \boldsymbol{\tau} - \mathbf{g} \;, \tag{2.14}$$

with the mass matrix

$$\mathbf{M}_{\mathrm{EE}} = \begin{bmatrix} m_{\mathrm{EE}} & 0 & 0 \\ 0 & m_{\mathrm{EE}} & 0 \\ 0 & 0 & I_{\mathrm{EE},z} \end{bmatrix} . \tag{2.15}$$

The vector $\mathbf{g}$ contains the gravitational force acting on the EE, i. e.

$$\mathbf{g} = \begin{bmatrix} 0 & m_{\mathrm{EE}} g & 0 \end{bmatrix}^{\mathrm{T}} , \tag{2.16}$$

where g denotes the gravitational acceleration.

By differentiating relation (2.12) with respect to time and substituting into Equation (2.13), then solving for $\boldsymbol{\tau}$ and inserting into (2.14), the equations of motion

$$\underbrace{\left(\mathbf{M}_{\mathrm{EE}} + \frac{I_{\mathrm{w}}}{\nu_{\mathrm{w}}^2}\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}\mathbf{J}_{\mathrm{ik}}\right)}_{\mathbf{M}(\mathbf{x})} \ddot{\mathbf{x}} + \underbrace{\left(\frac{I_{\mathrm{w}}}{\nu_{\mathrm{w}}^2}\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}\dot{\mathbf{J}}_{\mathrm{ik}}\right)}_{\mathbf{C}(\mathbf{x},\dot{\mathbf{x}})} \dot{\mathbf{x}} + \mathbf{g} = \underbrace{\frac{1}{\nu_{\mathrm{w}}}\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}}_{\boldsymbol{\Lambda}(\mathbf{x})} \mathbf{T} \tag{2.17}$$

can be obtained. Here, $\mathbf{M}(\mathbf{x})$ denotes the positive definite mass matrix, $\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})$ denotes the Coriolis matrix and $\boldsymbol{\Lambda}(\mathbf{x})$ the so-called torque transformation matrix.

3 Controller Design

In this chapter, a trajectory following control strategy is designed based on the mathematical model presented in Chapter 2. The work extends the preceding master's thesis [10] and elaborates on the control structure and provides additional details with respect to its limits, capabilities and implementation.

CDPRs exhibit the inherent constraint that sufficient tension must be guaranteed in all cables at all times, which must be ensured by the control strategy. For this purpose, a cable force distribution algorithm is discussed in Section 3.1. This algorithm is then used in Section 3.2 to develop a controller for the robot. The error dynamics of the closed loop are derived in Section 3.3 and the stability is assessed. To demonstrate and validate the controller performance, experiments are conducted on the CDPR prototype in Section 7.2.

3.1 Cable force distribution

3.1.1 Problem formulation

To ensure tension in all cables of a CDPR, the kinematic redundancies can be exploited. The wrench $\mathbf{f}$ created as a result of the cable forces is given by the relation (2.11). Because the cable forces required to achieve a given wrench are not unique, the null space of $\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}$ can be utilized to achieve positive cable forces. In the completely restrained case, where $n_q = n_x + 1$, the null space is 1-dimensional and the problem can be solved using comparatively simple algorithms as discussed in [23].

For a redundantly restrained CDPR, such as the present robot prototype where $n_q > n_x + 1$, additional redundancies must be resolved. To achieve a smooth motion of the EE along a smooth desired trajectory $\mathbf{x}_{\mathrm{d}}(t) \in \mathbb{R}^{n_x}$, the force distribution must be chosen such that the forces are continuous along the trajectory. This can be achieved by formulating the cable force distribution problem as an optimization problem using a p-norm in the form

$$\begin{aligned} \min_{\boldsymbol{\tau}} \quad & \|\boldsymbol{\tau} - \boldsymbol{\tau}_{\mathrm{d}}\|_p \\ \text{s.t.} \quad & \boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max}, \\ & \mathbf{f}_{\mathrm{d}} = -\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}} \boldsymbol{\tau} \ . \end{aligned} \tag{3.1}$$

Here, the desired cable pre-tension forces $\boldsymbol{\tau}_{\mathrm{d}}$ can be chosen as a design parameter. The limits for the cable forces $\boldsymbol{\tau}_{\min}$ and $\boldsymbol{\tau}_{\max}$ as well as the resulting task space wrench $\mathbf{f}_{\mathrm{d}}$ are enforced as constraints.

It can be shown, that the solution $\boldsymbol{\tau}^*$ of the optimization problem from Eq. (3.1) is unique and continuous for a continuous trajectory for $2 \leq p < \infty$ and thus, this formulation is suitable for the use in a CDPR [24, 25].

Choosing larger values of p causes the most stressed cable to have a larger contribution to the objective function value of the optimization problem from Eq. (3.1). This leads to smaller values of the force in the most stressed cable and is usually considered a more efficient utilization of the force transmission capabilities of the cables. However, larger values of p can introduce numerical issues. For handling high numbers of p, a specialized gradient method is proposed by Verhoeven in [24], which is computationally expensive and uses a sophisticated scaling strategy to minimize numerical errors. Therein, numerical results up to $p = 9$ are examined. A closed-form solution for $p = 2$ is presented by Pott et al. in [26], which is very computationally efficient and numerically robust.

An in-depth comparison of different numerical algorithms for solving (3.1) using different p-norms can be found in [19].

3.1.2 Solution method

In this work, the optimization problem from (3.1) using $p = 2$ is employed to solve the underconstrained cable force distribution problem with high numerical robustness and stability. Hence, the cable force distribution function $\boldsymbol{\tau}^* : \mathbb{R}^{n_x} \to \mathbb{R}^{n_q}$ can be defined in the form

$$\begin{aligned} \boldsymbol{\tau}^*(\mathbf{f}_\mathrm{d}) = \arg\min_{\boldsymbol{\tau}} \quad & \|\boldsymbol{\tau} - \boldsymbol{\tau}_\mathrm{d}\|_2 \\ \text{s.t.} \quad & \boldsymbol{\tau}_\mathrm{min} \le \boldsymbol{\tau} \le \boldsymbol{\tau}_\mathrm{max}, \\ & \mathbf{f}_\mathrm{d} = -\mathbf{J}_\mathrm{ik}^\mathrm{T}\boldsymbol{\tau} \ . \end{aligned} \tag{3.2}$$

To solve the optimization problem from Eq. (3.2), the so-called *improved closed-form method* by Pott [27] is employed. This method is an extension of the closed-form approach presented in [26] and it is very computationally efficient. Thus, the algorithm is particularly well suited for use in a real-time control system with short cycle times. Furthermore, the algorithm uses an iterative scheme to resolve all constraints which has a strictly bounded limit for the number of iterations. Thus, the execution time on a real-time computer system is strictly bounded.

In the following, the individual steps of the algorithm are briefly outlined and the behavior for the present CDPR is investigated.

The basic idea behind the method is to use the Moore-Penrose pseudoinverse to obtain a solution which satisfies the constraint $\mathbf{f}_\mathrm{d} = -\mathbf{J}_\mathrm{ik}^\mathrm{T}\boldsymbol{\tau}$ while minimizing the objective function $\|\boldsymbol{\tau} - \boldsymbol{\tau}_\mathrm{d}\|_2$. Such a solution can be found by separating the cable forces into the desired pre-tension forces $\boldsymbol{\tau}_\mathrm{d}$ and a deviation $\boldsymbol{\tau}_\Delta$ from these forces in the form

$$\boldsymbol{\tau} = \boldsymbol{\tau}_\mathrm{d} + \boldsymbol{\tau}_\Delta \ . \tag{3.3}$$

The wrench exerted on the EE by $\boldsymbol{\tau}_\mathrm{d}$ can be subtracted from the desired total wrench $\mathbf{f}_\mathrm{d}$ and a solution for $\boldsymbol{\tau}_\Delta$ can be computed which exerts the remaining force on the EE such that the total wrench is equal to $\mathbf{f}_\mathrm{d}$. Hence, a solution for the cable forces is obtained in the form

$$\boldsymbol{\tau}_\Delta = -\mathbf{J}_\mathrm{ik}^{\mathrm{T}+}(\mathbf{f}_\mathrm{d} + \mathbf{J}_\mathrm{ik}^\mathrm{T}\boldsymbol{\tau}_\mathrm{d}) \ , \tag{3.4}$$

using the pseudoinverse of the Jacobian transpose $\mathbf{J}_\mathrm{ik}^{\mathrm{T}+}$ which reads as

$$\mathbf{J}_\mathrm{ik}^{\mathrm{T}+} = (\mathbf{J}_\mathrm{ik}\mathbf{J}_\mathrm{ik}^\mathrm{T})^{-1}\mathbf{J}_\mathrm{ik} \ . \tag{3.5}$$

Combining Eqs. (3.3) and (3.4) yields an expression for the cable forces

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\mathrm{d}} - \mathbf{J}_{\mathrm{ik}}^{\mathrm{T}+}(\mathbf{f}_{\mathrm{d}} + \mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}\boldsymbol{\tau}_{\mathrm{d}}) \, , \tag{3.6}$$

which does not necessarily satisfy the constraints $\boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max}$. Thus, any violation of these constraints must be subsequently resolved. For this purpose, the cable force τ_i which violates the respective limit by the largest magnitude is fixed to the corresponding limit. As a consequence, the number of redundancies decreases and the problem can be reduced to a force distribution problem with $n_q - 1$ cables. This procedure is repeated iteratively until either a valid solution is found which satisfies $\boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max}$ or the problem can no longer be reduced because there are no more redundancies but Eq. (3.6) still does not satisfy the limits. In the latter case, the algorithm fails to find a valid solution.

The individual steps of the cable force distribution method are summarized in Algorithm 3.1.

Algorithm 3.1: Improved closed-form method [27]

```
Input:   $\mathbf{f}_{\mathrm{d}}$                                   (desired wrench)
         $\boldsymbol{\tau}_{\mathrm{d}}$                                   (desired cable tension)
         $\boldsymbol{\tau}_{\min},\ \boldsymbol{\tau}_{\max}$              (cable force limits)
         $\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}$                             (Jacobian transpose)
Output:  $\boldsymbol{\tau}^*$                                  (optimal cable forces)

$n_x \leftarrow$ Number of rows of $\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}$;
$n_q \leftarrow$ Number of columns of $\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}$;

repeat
    Step 1: Calculate $\boldsymbol{\tau}$ using Eq. (3.6);
    if $\boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max}$ then
        $\boldsymbol{\tau}^* \leftarrow \boldsymbol{\tau}$ ;
        return $\boldsymbol{\tau}^*$ $\Rightarrow$ (Feasible solution found!)
    end
    Step 2: Find $i$ for which $\tau_i$ violates the limits most;
    $\tau_{\mathrm{lim}} \leftarrow$ most violated limit $\tau_{\min}$ or $\tau_{\max}$;
    $\tau_i^* \leftarrow \tau_{\mathrm{lim}}$ ;
    Step 3: Remove $i$-th cable to reduce problem:
    $\left[\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}\right]_i \leftarrow$ $i$-th column of $\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}$ ;
    $\mathbf{f}_{\mathrm{d}} \leftarrow \mathbf{f}_{\mathrm{d}} + \left[\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}\right]_i \tau_{\mathrm{lim}}$ ;
    $\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}} \leftarrow$ remove $i$-th column from $\mathbf{J}_{\mathrm{ik}}^{\mathrm{T}}$ ;
    $n_q \leftarrow n_q - 1$;
until $n_q < n_x$;

No feasible solution found!
```

The strategy employed in the *improved closed-form method* for resolving violations of the cable force limits is based on the assumption, that the cable which violates the constraints most must be fixed at the respective limit in the optimal solution of Eq. (3.2). Hence, the most violated inequality constraint is assumed to be active and acts as an equality constraint. To justify this assumption, the author of [27] suggests using

$$\boldsymbol{\tau}_{\mathrm{d}} = \frac{1}{2}(\boldsymbol{\tau}_{\mathrm{min}} + \boldsymbol{\tau}_{\mathrm{max}}) = \boldsymbol{\tau}_{\mathrm{m}} \; . \tag{3.7}$$

This choice guarantees symmetry of the forces $\boldsymbol{\tau}_{\Delta}$ with respect to the cable force limits and ensures the best possible behavior with respect to finding a feasible solution. For any other choice of $\boldsymbol{\tau}_{\mathrm{d}}$, there is a bias towards either selecting the lower or upper inequality constraints as active constraints prematurely. Thus, feasible solutions where different inequality constraints are active can be excluded although they might exist. Because the limits which are closer to $\boldsymbol{\tau}_{\mathrm{d}}$ are preferred by the algorithm, the method always assumes that those inequality constraints are active, which correspond to solutions which have lower objective function values. Alternative solutions where other combinations of inequality constraints are active, i. e. other cables are at the limit, are discarded. These prematurely discarded combinations of active constraints might be feasible solutions but always yield larger cost function values than the preferred combinations of active constraints.

For this reason, the algorithm might fail to find a feasible force distribution but it will never return a sub-optimal or non-feasible solution. In other words, if the algorithm finds a solution then this solution is guaranteed to be optimal. However, it is possible that the algorithm fails to find a feasible solution even though it exists.

3.1.3 Solution behavior

In this section, the behavior of the *improved closed-form method* is investigated via numerical simulations. As previously discussed in Section 3.1.2, the behavior of the algorithm with respect to finding a feasible solution depends on the choice of $\boldsymbol{\tau}_{\mathrm{d}}$. The method was originally conceived for $\boldsymbol{\tau}_{\mathrm{d}} = \boldsymbol{\tau}_{\mathrm{m}}$, see Eq. (3.7), and any other choice of $\boldsymbol{\tau}_{\mathrm{d}}$ negatively impacts the ability of the method to find feasible solutions. Using $\boldsymbol{\tau}_{\mathrm{d}} = \boldsymbol{\tau}_{\mathrm{m}}$ is a reasonable compromise for choosing the desired cable pre-tension force since an equal distance from the lower and upper limits results in a balanced distribution of forces. However, from a practical perspective $\boldsymbol{\tau}_{\mathrm{m}}$ might be unnecessarily large and result in unnecessarily large standstill torques in the driving motors when the robot is at rest and the EE is not moving. These large standstill torques cause high standstill currents and result in unnecessarily large losses in the driving motors and power electronics. In particular, for the task at hand when catching a flying dart, the peak acceleration of the CDPR is of primary importance. This peak acceleration must be provided by the robot only for a short amount of time while the dart is in the air and the dartboard is moved to the impact location of the dart and not continuously during operation. Thus, the upper force limits $\boldsymbol{\tau}_{\mathrm{max}}$ originate from the peak torque limits of the drivetrain whereas the desired torque $\boldsymbol{\tau}_{\mathrm{d}}$ must be chosen such that it can be continuously provided by the electric drive system. Commonly, the continuous torque ratings of electric motors are limited by thermal effects and considerably lower than the peak torques, which can be provided for a short duration of time due to the thermal time constants present in the system. Hence, choosing $\boldsymbol{\tau}_{\mathrm{d}} = \boldsymbol{\tau}_{\mathrm{m}}$ might either violate the rated continuous power of the electric drive system or restrict the choice of $\boldsymbol{\tau}_{\mathrm{max}}$.

For this reason, the impact of decreasing the target cable force $\boldsymbol{\tau}_\mathrm{d}$ is studied in the following. In addition, the so-called wrench-feasible workspace is investigated for wrenches with different magnitude and direction to gain more insight into the capabilities of the CDPR under the given force limits. The wrench-feasible workspace for a given wrench $\mathbf{f}_\mathrm{d}$ is defined as the set of all EE poses $\mathbf{x}$, for which the optimization problem from Eq. (3.2) is feasible.

Different desired wrenches $\mathbf{f}_\mathrm{d}$ are applied to the EE for this purpose. These wrenches are chosen in the form

$$\mathbf{f}_\mathrm{d} = f_d\,\mathbf{d}\,, \tag{3.8}$$

where the magnitude of the resulting wrench is given by the scalar f_d and the direction is specified by $\mathbf{d}$. For the directions, a cylindrical coordinate system centered at the origin is used as depicted in Figure 3.1.

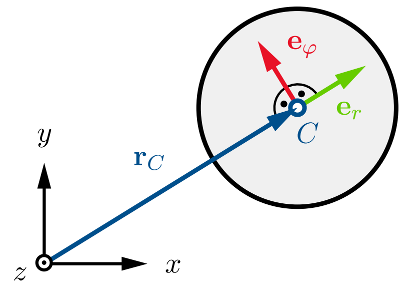


Figure 3.1: Basis vectors $\mathbf{e}_r$ and $\mathbf{e}_\varphi$ for cylindrical coordinates.

The radial basis vector $\mathbf{e}_r$ and the azimutal basis vector $\mathbf{e}_\varphi$ are given by

$$\mathbf{e}_r = \frac{1}{\sqrt{x_C^2 + y_C^2}}\begin{bmatrix} x_C \\ y_C \\ 0 \end{bmatrix}, \qquad \mathbf{e}_\varphi = \frac{1}{\sqrt{x_C^2 + y_C^2}}\begin{bmatrix} -y_C \\ x_C \\ 0 \end{bmatrix}. \tag{3.9}$$

The *improved closed-form method* as described in Algorithm 3.1 is applied to find the region, where a feasible force distribution $\boldsymbol{\tau}^*$ can be successfully found by the algorithm for each desired wrench $\mathbf{f}_\mathrm{d}$. To find the regions in which the algorithm fails to find a feasible solution even though one exists, the MATLAB function `lsqlin` is used to solve the optimization problem from Eq. (3.2). This command implements the trust-region method described in [28].

The dimensions and the cable force limits of the prototype CDPR developed in [10] are used. The cable force limits are identical for all cables and can be written in the form

$$\boldsymbol{\tau}_\mathrm{min} = \mathbf{1}\tau_\mathrm{min} \qquad \boldsymbol{\tau}_\mathrm{max} = \mathbf{1}\tau_\mathrm{max} \tag{3.10}$$

$$\tau_\mathrm{min} = 10\,\mathrm{N} \qquad \tau_\mathrm{max} = 280\,\mathrm{N}\,. \tag{3.11}$$

For analyzing the behavior of the algorithm, different values for the desired cable tension $\boldsymbol{\tau}_\mathrm{d}$ are chosen. Here, equal desired tension forces are selected for all cables such that $\boldsymbol{\tau}_\mathrm{d} = \mathbf{1}\tau_\mathrm{d}$. Figure 3.2 presents the results for $\tau_\mathrm{d} = \tau_\mathrm{m} = 145\,\mathrm{N}$. For comparison, Figure 3.3 shows the behavior of the algorithm, when the desired cable tension is reduced to $\tau_\mathrm{d} = 50\,\mathrm{N}$. The wrench-feasible workspace i. e. the set of EE poses for which the desired wrench $\mathbf{f}_\mathrm{d}$ is feasible is visualized using different shades of green color in Figures 3.2 and 3.3. The regions in which the force distribution algorithm fails to find a feasible solution are highlighted using shades of red in Figures 3.2 and 3.3.

The wrench-feasible workspace in radial outward direction $\mathbf{e}_r$ contracts quickly for larger magnitudes f_d of the desired wrench. This is visualized by the green regions in Figures 3.2(a) and 3.3(a). In comparison, the three side-lobes of the wrench-feasible workspace in tangential direction $\mathbf{e}_\varphi$ become more and more curved when the magnitude f_d of the desired wrench is increased as shown in Figures 3.2(b) and 3.3(b).

When the desired cable tension is chosen as $\tau_\mathrm{d} = \tau_\mathrm{m} = 145\,\mathrm{N}$, the algorithm performs well and only fails in very few cases close to the border of the wrench-feasible workspace as indicated by the red regions in Figure 3.2. In the case of wrenches in radially outward direction $\mathbf{e}_r$, the algorithm only fails to find a feasible cable force distribution in a single point located in the bottom right region of Figure 3.2(a) for $f_\mathrm{d} = 200\,\mathrm{N}$. This could potentially be caused by numerical issues. In comparison, the algorithm fails to find a feasible cable force distribution for wrenches in tangential direction $\mathbf{e}_\varphi$ close to the border of the wrench-feasible workspace as depicted in Figure 3.2(b). This confirms that the assumption employed in choosing the active constraints does not hold in general, even for the special choice $\tau_\mathrm{d} = \tau_\mathrm{m}$.

When the desired tension force τ_d is reduced, the algorithm still performs well for wrenches in radial outwards direction $\mathbf{e}_r$. Figure 3.3(a) shows that the algorithm only fails to find a feasible solution close to the borders of the wrench-feasible workspace for larger amplitudes of the desired wrench f_d. For smaller amplitudes of f_d, the *improved closed-form method* succeeds in finding the optimal solution in almost the entire wrench-feasible workspace.

In comparison, for wrenches in tangential direction $\mathbf{e}_\varphi$ there exist larger regions where the algorithm fails to find a feasible force distribution. These regions are visualized in Figure 3.3(b) and extend closer to the center of the workspace.

In conclusion, the studies performed in this section show that the effective wrench-feasible workspace shrinks when decreasing the desired cable tension τ_d due to the inability of the cable force distribution algorithm to find all feasible cable force distributions. The quantitative results suggest that for the present CDPR prototype a reduction to $\tau_\mathrm{d} = 50\,\mathrm{N}$ yields a practically useful workspace for wrenches with a magnitude of up to $f_\mathrm{d} \approx 300\,\mathrm{N}$.

3.2 Control structure

In this section, a controller for the CDPR is designed based on the cable force distribution algorithm from Section 3.1 and the control-oriented dynamical robot model outlined in Section 2.2.

The controller follows a desired trajectory in the task space given by $\mathbf{x}_\mathrm{d}(t) \in \mathbb{R}^{n_x}$ and stabilizes the trajectory tracking error defined as

$$\mathbf{e}_\mathrm{x} = \begin{bmatrix} e_x & e_y & e_\varphi \end{bmatrix}^\mathrm{T} = \mathbf{x} - \mathbf{x}_\mathrm{d}\ . \tag{3.12}$$

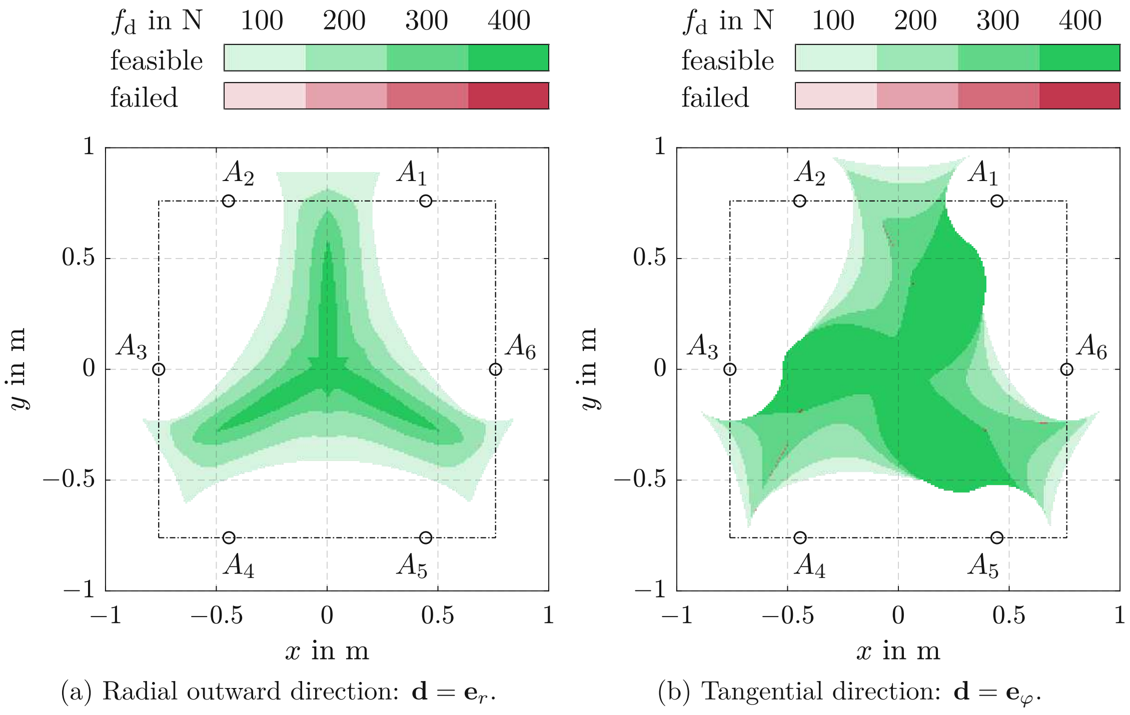


Figure 3.2: Solution behavior for $\tau_d = 145\,\text{N}$.

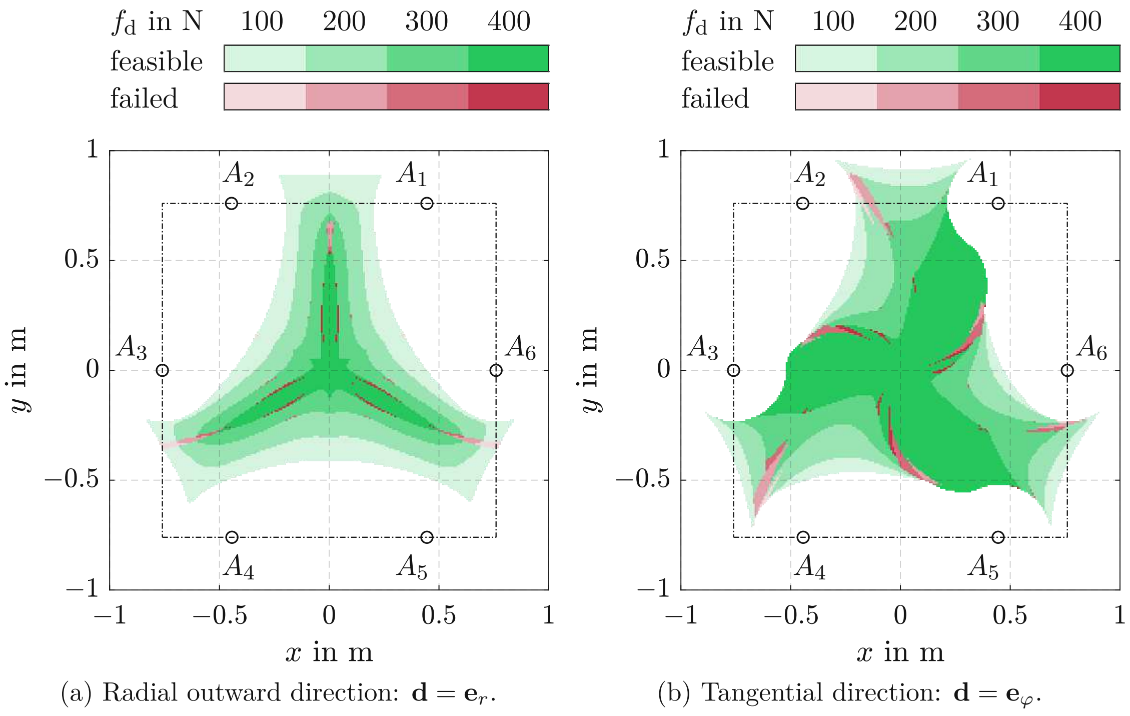


Figure 3.3: Solution behavior for $\tau_d = 50\,\text{N}$.

To design a control law for the CDPR, the robot can be treated as two separate subsystems. The first subsystem is formed by the EE and the second subsystem encompasses the n_q winches. These two subsystems are coupled via cables as visualized in Figure 2.2.

By applying the force distribution function $\boldsymbol{\tau}^*(\mathbf{f}_\mathrm{d})$ from Eq. (3.2), a desired wrench $\mathbf{f}_\mathrm{d}$ can be exerted on the EE while ensuring that the cables remain under sufficient tension. Hence, the force distribution algorithm serves as a tool to handle the unilateral constraint that the cables can only transmit pulling forces from the winches to the EE. With this in mind, a control strategy can be designed for the two subsystems separately. Then the individual strategies can be combined by making use of the cable force distribution algorithm.

The so-called PD+ control concept, introduced in [29], is a variation of the very popular computed torque control scheme. It is composed of a control loop with proportional and derivative feedback (PD) with an additional compensation, which feeds forward the nominal joint forces [30]. Such a controller is developed in two steps. In the first step, a PD+ controller for the EE subsystem is designed. Subsequently in the second step, a compensation for the inertial forces caused by the winch subsystem is added.

For the first step, only the EE subsystem is considered. A PD+ control law in task space for the EE subsystem is obtained by rearranging Eq. (2.14). This yields the expression

$$\mathbf{f}_\mathrm{d} = \mathbf{M}_\mathrm{EE}\ddot{\mathbf{x}}_\mathrm{d} + \mathbf{g} - \mathbf{K}_\mathrm{P}\mathbf{e}_\mathrm{x} - \mathbf{K}_\mathrm{D}\dot{\mathbf{e}}_\mathrm{x} \tag{3.13}$$

for the desired wrench $\mathbf{f}_\mathrm{d}$ acting on the EE. Here, the inertial forces of the EE given by $\mathbf{M}_\mathrm{EE}\ddot{\mathbf{x}}_\mathrm{d}$, as well as the gravitational forces $\mathbf{g}$ acting on the EE are compensated. In addition, the gain matrices $\mathbf{K}_\mathrm{P}$ and $\mathbf{K}_\mathrm{D}$ denote the P (proportional) and D (derivative) components of the controller, respectively.

In a second step, compensation terms for the inertial forces caused by the winches are obtained by differentiating the relation from Eq. (2.12) and inserting into Eq. (2.13). This yields the expression

$$\mathbf{T}_\mathrm{w} = \frac{I_\mathrm{w}}{\nu_\mathrm{w}}\left(\dot{\mathbf{J}}_\mathrm{ik}\dot{\mathbf{x}}_\mathrm{d} + \mathbf{J}_\mathrm{ik}\ddot{\mathbf{x}}_\mathrm{d}\right) , \tag{3.14}$$

where $\mathbf{T}_\mathrm{w}$ denotes the torques necessary to overcome the inertial forces of the winches and achieve the desired acceleration.

Finally, the winch inertia compensation term from Eq. (3.14) can be combined with the PD+ controller for the EE from Eq. (3.13) by applying the cable force distribution function $\boldsymbol{\tau}^*(\mathbf{f}_\mathrm{d})$ and respecting the winch transmission ratio ν_w. This yields the control law

$$\mathbf{T}_\mathrm{d} = -\nu_\mathrm{w} \underbrace{\boldsymbol{\tau}^*}_{\substack{\text{Force}\\ \text{Dist.}}} \Bigg(\underbrace{\mathbf{M}_\mathrm{EE}\ddot{\mathbf{x}}_\mathrm{d} + \mathbf{g} - \mathbf{K}_\mathrm{P}\mathbf{e}_\mathrm{x} - \mathbf{K}_\mathrm{D}\dot{\mathbf{e}}_\mathrm{x}}_{\substack{\text{PD+ for EE}\\ \text{Eq. (3.13)}}} \Bigg) + \underbrace{\frac{I_\mathrm{w}}{\nu_\mathrm{w}}\left(\dot{\mathbf{J}}_\mathrm{ik}\dot{\mathbf{x}}_\mathrm{d} + \mathbf{J}_\mathrm{ik}\ddot{\mathbf{x}}_\mathrm{d}\right)}_{\substack{\text{Winch Inertia}\\ \text{Compensation}}} , \tag{3.15}$$

where the control variable $\mathbf{T}_\mathrm{d}$ denotes the desired motor torque. The desired motor torque acts as an input for the cascaded torque control in which the motor voltage $\mathbf{u}_i$ and current $\mathbf{i}_i$ for each motor are controlled such that the desired torque can be established.

Figure 3.4 provides an overview of the resulting control structure and visualizes the aforementioned individual components of the control scheme.

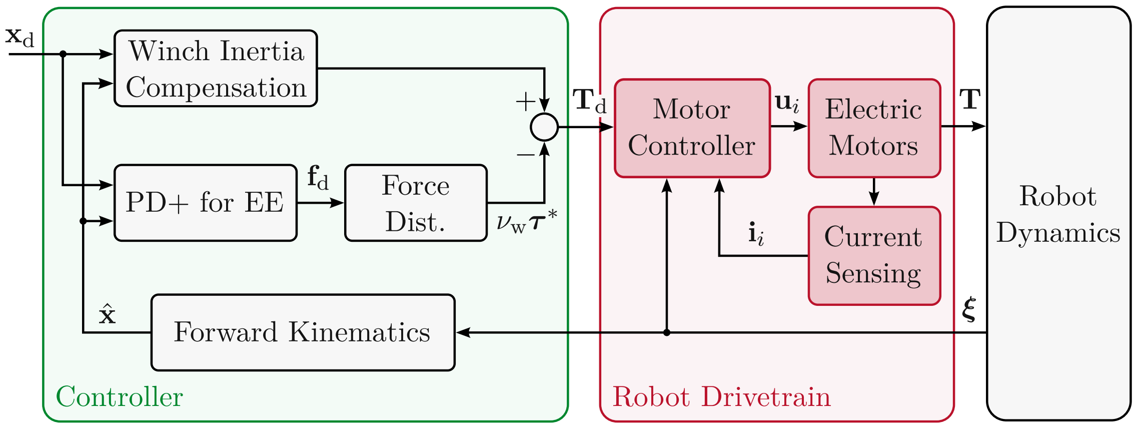


Figure 3.4: Structure of the control concept [10].

It should be noted, that the pose $\mathbf{x}$ of the EE is not directly accessible as a measurement but has to be calculated from the winch angles $\boldsymbol{\xi}$ instead. For this purpose, the forward kinematics is used, which is discussed in detail in [10]. The estimated EE pose using the forward kinematics is denoted $\hat{\mathbf{x}}$.

The controller relies on solving the optimization problem from Eq. (3.2) to guarantee appropriate cable forces. Hence, in case no feasible cable force distribution $\boldsymbol{\tau}^*(\mathbf{f}_\mathrm{d})$ can be found, the robot must be halted to prevent any damage to the cable system due to inadequate cable forces. To prevent this from happening, special care must be taken that the desired trajectory $\mathbf{x}_\mathrm{d}(t)$ is feasible within the dynamical capabilities of the CDPR. This is no trivial task because the set of feasible wrenches is not straightforward to compute and varies significantly within the workspace. To solve this problem for the present application, a real-time capable trajectory generation algorithm which limits the desired velocity $\dot{\mathbf{x}}_\mathrm{d}$ and acceleration $\ddot{\mathbf{x}}_\mathrm{d}$ is presented in Chapter 4.

3.3 Error dynamics and stability

To gain insight into the behavior of the closed-loop system, the error dynamics are derived and analyzed. In the following, it is assumed that the measurement noise is small and as a consequence $\hat{\mathbf{x}} = \mathbf{x}$. Furthermore, it is assumed that the dynamics of the motor torques in the drivetrain are significantly faster than the robot dynamics. Hence, the dynamics of the inner control loop of the cascaded torque control can be neglected and it can be assumed that $\mathbf{T} = \mathbf{T}_\mathrm{d}$.

Substituting the control law from Eq. (3.15) into the equations of motion from Eq. (2.17) yields the error dynamics in the form

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{e}}_\mathrm{x} + (\mathbf{C}(\mathbf{x},\dot{\mathbf{x}}) + \mathbf{K}_\mathrm{D})\dot{\mathbf{e}}_\mathrm{x} + \mathbf{K}_\mathrm{P}\mathbf{e}_\mathrm{x} = \mathbf{0}\ . \tag{3.16}$$

The nonlinear ordinary differential equations from Eq. (3.16) governing the error system are well known for a PD+ controller in the literature [30, 31]. Furthermore, due to the dependence on the trajectory $\mathbf{x}_\mathrm{d}$, the error system is non-autonomous. Via a suitable choice of the controller parameters $\mathbf{K}_\mathrm{P}$ and $\mathbf{K}_\mathrm{D}$, the error dynamics can be adjusted as desired. Thus, the PD+ control scheme in task space can be interpreted as a mechanical compliance controller where the mechanical compliance of the robot along the trajectory can be directly adjusted via the controller parameters. The expression $(\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{K}_\mathrm{D})$ can be interpreted as mechanical damping of the error system and similarly the term $\mathbf{K}_\mathrm{P}$ can be associated with the mechanical stiffness of the error system.

It can be shown that the trajectory tracking error $\mathbf{e}_\mathrm{x}$, described by Eq.(3.16), is globally asymptotically stable for a symmetric and positive definite choice of $\mathbf{K}_\mathrm{P}$ and a positive definite choice of $\mathbf{K}_\mathrm{D}$. A proof can be found in [30] using an appropriate Lyapunov function and the theorem of Matrosov. The controller parameters used in the CDPR prototype are chosen constant and diagonal in the form

$$\mathbf{K}_\mathrm{P} = \begin{bmatrix} K_\mathrm{P} & 0 & 0 \\ 0 & K_\mathrm{P} & 0 \\ 0 & 0 & K_{\mathrm{P},\varphi} \end{bmatrix}, \qquad \mathbf{K}_\mathrm{D} = \begin{bmatrix} K_\mathrm{D} & 0 & 0 \\ 0 & K_\mathrm{D} & 0 \\ 0 & 0 & K_{\mathrm{D},\varphi} \end{bmatrix} \tag{3.17}$$

as described in [10]. Note, that the global asymptotic stability of the PD+ controller is retained, even if the controller matrices $\mathbf{K}_\mathrm{P}$ and $\mathbf{K}_\mathrm{D}$ are chosen time-variant. Thus, the controller can be designed such that the error dynamics are constant and independent of the EE pose $\mathbf{x}$ if this behavior is desired.

4 Trajectory generation

The problem of generating a suitable trajectory for the motion of a robot is a fundamental challenge in robotics. Depending on the application, the constraints and requirements can vary greatly. For example, for some applications the geometric path might be given and a temporal parameterization of that path might be sought. In other cases, the planning of the geometrical path and the time scaling might be necessary and additional constraints such as obstacles might be of interest. An in-depth discussion on motion planning in general and different trajectory generation schemes can be found in [32].

In this chapter, the trajectory generation problem for the present task of catching a flying dart using a CDPR is tackled. First, in Section 4.1 the problem is formulated for the specific task at hand. Second, related work from the literature is reviewed and discussed in Section 4.2. Third, a real-time capable online trajectory generation algorithm (OTG) is proposed based on similar work from the literature in Section 4.3. Finally, in Section 4.4 simulations are conducted to assess and demonstrate the effectiveness and practicality of the presented algorithm.

4.1 Problem formulation

In the chosen system architecture, outlined in Section 1.2 and visualized in Figure 1.2, the trajectory generator receives a set-point $\mathbf{x}_{\text{set}}[k]$ for the EE pose at each timestep k and is required to generate a sufficiently smooth trajectory $(\mathbf{x}_{\text{d}}[k], \dot{\mathbf{x}}_{\text{d}}[k], \ddot{\mathbf{x}}_{\text{d}}[k])$ as an output. The control system is implemented as a discrete time system with the sample time T_s. Hence, all time signals are represented by sequences using the time index $k = 0, 1, 2, \ldots$, such that $\mathbf{x}[k] = \mathbf{x}(kT_s)$.

The set-point $\mathbf{x}_{\text{set}}[k]$ corresponds to the current prediction of the dart impact location and is frequently updated as the dart approaches the dartboard and the prediction is refined. The accuracy of the estimate increases throughout the flight of the dart. In particular during the early stage of the flight, large changes of $\mathbf{x}_{\text{set}}[k]$ are possible.

For this reason, the trajectory generator must be able to adapt the trajectory in real-time and must be robust with respect to large changes in the set-point $\mathbf{x}_{\text{set}}[k]$. Furthermore, the adapted and re-planned trajectory must be smooth up to the second derivative and thus smoothly continue the previously generated values for $(\mathbf{x}_{\text{d}}[k-1], \dot{\mathbf{x}}_{\text{d}}[k-1], \ddot{\mathbf{x}}_{\text{d}}[k-1])$ and eventually end with the set-point pose at rest, i. e. $(\mathbf{x}_{\text{set}}[j], \mathbf{0}, \mathbf{0})$.

As explained in Chapter 3, the required forces and torques acting on the EE must be feasible within the cable force limits. Thus, the trajectory generation algorithm must incorporate limits for the EE motion such that the trajectory can be successfully tracked by the controller.

These considerations can be summarized as stated in Problem 4.1, to formulate the trajectory generation problem for the present application:

Problem 4.1. *Given the previous trajectory history* $(\mathbf{x}_{\mathrm{d}}[i], \dot{\mathbf{x}}_{\mathrm{d}}[i], \ddot{\mathbf{x}}_{\mathrm{d}}[i])$ *where* $i = 0, 1, \ldots, k-1$*, calculate the current trajectory values* $(\mathbf{x}_{\mathrm{d}}[k], \dot{\mathbf{x}}_{\mathrm{d}}[k], \ddot{\mathbf{x}}_{\mathrm{d}}[k])$ *such that the trajectory transitions in finite time to* $(\mathbf{x}_{\mathrm{set}}[k], \mathbf{0}, \mathbf{0})$*, subject to constraints on the velocity* $\dot{\mathbf{x}}_{\mathrm{d}}$ *and the acceleration* $\ddot{\mathbf{x}}_{\mathrm{d}}$*.*

In addition, the Objectives 4.1 are chosen for the specific task of catching a dart.

Objectives 4.1. *The online trajectory generator should satisfy the following objectives:*

O1 *The trajectory should not oscillate or spiral around the set-point* $\mathbf{x}_{\mathrm{set}}$*.*

O2 *The algorithm should be computationally efficient and enable high sampling frequencies of* $f_s > 1\,\mathrm{kHz}$*.*

O3 *The trajectory should be time-optimal for straight motions, e. g. when the set-point* $\mathbf{x}_{\mathrm{set}}$ *is constant and the trajectory starts with zero velocity and zero acceleration.*

The problem statement from Problem 4.1 is relatively general and can be interpreted as a filtering task. In essence, the trajectory generator acts as a filter which smooths the non-continuous set-point values $\mathbf{x}_{\mathrm{set}}[j]$ and thus ensures that the trajectory can be tracked by the controller. To ensure that the trajectory generation algorithm is well suited for the present application, the additional objectives from Objectives 4.1 should be fulfilled by the algorithm. In particular, ***O1*** should guarantee that the trajectory does not excessively spiral or oscillate just before catching the dart. Shortly before the dart impact, the impact prediction is relatively accurate and only minor changes are expected. In addition, ***O2*** is chosen to achieve a high controller sampling frequency which is critical for a fast response of the robot. To achieve fast positioning of the dartboard in the early stage of catching the dart, Objective ***O3*** is introduced. Thus, it is guaranteed that the initial coarse positioning of the dartboard is very time-efficient, when the impact location of the dart has been computed for the first time.

Similar problems frequently arise in many different robotic applications and, thus, there exist many approaches in the literature to solve the present trajectory generation problem. In Section 4.2, different strategies from the literature are reviewed and investigated. Thereafter, in Section 4.3, a suitable algorithm for the dart catching application is designed.

4.2 Literature review

The research field of trajectory generation and motion planning is a very broad topic and there exists a vast variety of literature on the subject [33]. This section aims at providing a brief overview of different approaches to tackle the problem and different solutions for very similar problems found in the literature.

Whenever a robot is required to react quickly to an external unforeseen sensor input, the need for online trajectory generation (OTG) arises [34]. This covers a very wide range of applications, from human-robot interaction where a human provides unforeseeable input, to demanding industrial processes such as tracking applications where sensor input can change quickly and spontaneously.

To enable a robot to follow the generated trajectory, different kinds of limits can be implemented. To optimize the trajectory for given hardware components, the joint torques can be used as *dynamic* constraints [35]. On the other hand, it might be desirable to implement purely *kinematic* constraints, e. g. for velocity and acceleration, which can be done either in joint space or in task space. For example, in a human-robot interaction scenario it might be necessary to constrain the maximum EE velocity and acceleration in task space. Thus, the motion of the tool attached to the EE can be constrained to ensure safe operation [36].

The approaches found in the literature can be classified in three main groups as elaborated in [36]:

1) **Direct approaches:** These strategies define and compute a whole trajectory profile until the target state. Often a sequence of piece-wise polynomials is used which allows for a synthesis of trajectories with the desired starting and end states. In [37], quintic polynomials are adapted in rapid succession to quickly position a ping-pong racket. A heuristic hybrid approach presented in [38] uses a fast trajectory for coarse positioning and a fine-tuned polynomial to reach the desired target state.

   Time-optimal trajectories can be generated by constructing so-called S-curves [20], which follow constant jerk, acceleration and velocity limits. Such trajectories are designed in joint space and usually consider kinematic constraints for the joint trajectories. The method shown in [39] uses decision trees to construct synchronized time-optimal S-curve trajectories for multiple joints with the desired start and end states. A joint space trajectory generation method considering the robot dynamics and thus incorporating kinetic limits is presented in [34].

2) **Indirect approaches:** The idea of these approaches is to consider the trajectory generation problem a dynamic control or a filtering problem. Thus, not the entire trajectory is constructed but only the next timestep is considered with the aim of driving the trajectory towards the target set-point. Since only the next timestep is considered, these approaches are inherently well-suited for online implementation due to their comparably low computational effort.

   In [40], a so-called α-β-γ filter is used to generate a smooth trajectory for grasping a moving object tracked via an optical measurement system. A cascade of FIR-filters is used in [41] to generate a trajectory, while [42] presents a method which uses an adaptive FIR filter to generate a jerk-limited trajectory from an acceleration limited trajectory. Using a nonlinear filter structure, the method presented in [43] is able to generate near time-optimal trajectories in a single dimension. The method developed by Lloyd in [44] generates trajectories in 3D Euclidean space which are subject to limits of the ∞-norm of velocity and acceleration by splitting the motion into components and treating each component independently. Huber et al. [36] present a method for the online trajectory generation of both 3D translation and rotation in Euclidean space with a focus on human-robot interaction scenarios.

A nonlinear filter is designed in [45] to generate a trajectory for a unicycle-like mobile robot.

3) **Optimization based approaches:** By formulating the trajectory generation task as a dynamical optimization problem, time-optimal trajectories satisfying the desired constraints can be synthesised. However, these approaches are usually very computationally expensive and difficult to implement online with low cycle times. Depending on the goal at hand, there exist many different approaches and strategies. Works such as [35], aim at reducing tracking error and motion time by considering the robot's dynamics and using a numerical solver. These methods feature trajectory planning times of several seconds and are thus designed for offline trajectory generation.

   On the other hand, there exist real-time capable optimization based methods such as [46]. In this work, a model predictive control structure is used for trajectory planning in a ball-catching scenario.

   There also exist analytical optimization based techniques for trajectory generation such as [47]. The problem studied by Akulenko in this work originates from aviation and deals with the optimal trajectory for moving a particle to the desired state with limited acceleration but no velocity limits.

Upon closer investigation, it was found that only the algorithm presented by Lloyd [44] formally satisfies the Objectives 4.1 when solving Problem 4.1. However, the algorithm proposed by Lloyd only allows limits in the ∞-norm of velocity and acceleration.

Some other *indirect approaches*, such as the work of Huber et al. [36], satisfy the objectives in most scenarios but tend to generate spiral motions in some cases. Thus, the Objective ***O1*** is violated. An example of such a scenario is shown in Figure 4.6(a).

Most of the *direct approaches* found in the literature generate a trajectory in joint space which is not suitable for the chosen control structure. Approaches using quintic polynomials violate ***O3*** because the generated trajectories are not time-optimal for straight motions.

Using an optimization based approach while still satisfying ***O2*** is difficult, due to the large computational effort involved. The analytical optimization based approach from [47] is promising in terms of high computational efficiency but lacks a velocity limit.

For this reason, a novel trajectory generation algorithm is designed in the following.

4.3 Trajectory generation algorithm

In this section, a real-time capable trajectory generating algorithm is designed to solve Problem 4.1 while fulfilling the Objectives 4.1. According to the classification outlined in Section 4.2, the developed strategy can be considered an *indirect* approach because the problem is treated as a control task and only the next timestep of the trajectory is considered. For this purpose, a nonlinear switching algorithm is designed which discretely switches between two different states to generate the trajectory values $(\mathbf{x}_\mathrm{d}[k], \dot{\mathbf{x}}_\mathrm{d}[k], \ddot{\mathbf{x}}_\mathrm{d}[k])$. The algorithm is presented in the following and the algorithm's behavior is investigated thereafter in Section 4.4 via simulations.

The proposed method is inspired by the works of Lloyd [44] and Huber et al. [36]. The trajectory for the pose $\mathbf{x}_\mathrm{d}[k]$ can be partitioned in a trajectory for the EE position $\mathbf{r}_\mathrm{d}[k]$ and a trajectory for the EE rotation angle $\varphi_\mathrm{d}[k]$ in the form

$$\mathbf{x}_\mathrm{d} = \begin{bmatrix} x_\mathrm{d} \\ y_\mathrm{d} \\ \varphi_\mathrm{d} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_\mathrm{d} \\ \varphi_\mathrm{d} \end{bmatrix} , \qquad \mathbf{x}_\mathrm{set} = \begin{bmatrix} x_\mathrm{set} \\ y_\mathrm{set} \\ \varphi_\mathrm{set} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_\mathrm{set} \\ \varphi_\mathrm{set} \end{bmatrix} . \tag{4.1}$$

The translation and the rotation of the EE are considered independently from each other and, thus, the trajectories $\mathbf{r}_\mathrm{d}[k]$ and $\varphi_\mathrm{d}[k]$ are generated independently. For the control strategy outlined in Chapter 3, the trajectory must be generated in the task space. To limit the motion of the EE, kinematic limits in the form

$$\|\mathbf{v}_\mathrm{d}\|_2 \leq v_\mathrm{max} \qquad \|\mathbf{a}_\mathrm{d}\|_2 \leq a_\mathrm{max} \qquad \|\mathbf{j}_\mathrm{d}\|_2 \leq M_r < \infty \tag{4.2}$$

$$|\omega_\mathrm{d}| \leq \omega_\mathrm{max} \qquad |\dot{\omega}_\mathrm{d}| \leq \dot{\omega}_\mathrm{max} \qquad |\ddot{\omega}_\mathrm{d}| \leq M_\varphi < \infty \tag{4.3}$$

are prescribed as constraints. Here, the desired EE velocity $\dot{\mathbf{r}}_\mathrm{d}$ is denoted $\mathbf{v}_\mathrm{d}$, the desired EE acceleration $\ddot{\mathbf{r}}_\mathrm{d}$ is denoted $\mathbf{a}_\mathrm{d}$ and the EE jerk $\dddot{\mathbf{r}}_\mathrm{d}$ is denoted $\mathbf{j}_\mathrm{d}$ along the trajectory. Furthermore, ω_d denotes the desired angular velocity $\dot{\varphi}_\mathrm{d}$. Enforcing dynamical limits associated with the cable force limits is a difficult task for the CDPR and challenging to implement online. For this reason, the kinematic limits v_max, a_max, ω_max and $\dot{\omega}_\mathrm{max}$ are chosen constant. The values for the kinematic limits must be chosen appropriately, to allow the controller outlined in Chapter 3 to track the desired trajectory. In addition, the jerk $\|\mathbf{j}_\mathrm{d}\|_2$ and the angular jerk $|\ddot{\omega}_\mathrm{d}|$ must be bounded by a finite constant for all motions. The acceleration $\|\mathbf{a}_\mathrm{d}\|_2$ and the angular acceleration $\dot{\boldsymbol{\omega}}_\mathrm{d}$ directly impact the motor torques $\mathbf{T}_\mathrm{d}$. Thus, by limiting the rate of change of the accelerations to a finite value, the rate of change of the torques $\dot{\mathbf{T}}_\mathrm{d}$ is also finite. This allows the cascaded motor controllers to more accurately track the desired torque values and minimizes unwanted oscillations in the drive-trains.

4.3.1 Translation

Consider the known position $\mathbf{r}_\mathrm{d}[k-1]$, velocity $\mathbf{v}_\mathrm{d}[k-1]$ and acceleration $\mathbf{a}_\mathrm{d}[k-1]$ from the previous timestep $k-1$. Because the acceleration of the previous timestep is known, the position and velocity can be computed via integration neglecting higher order terms in the form

$$\mathbf{v}_\mathrm{d}[k] = \mathbf{v}_\mathrm{d}[k-1] + \mathbf{a}_\mathrm{d}[k-1]T_s \tag{4.4}$$

$$\mathbf{r}_\mathrm{d}[k] = \mathbf{r}_\mathrm{d}[k-1] + \mathbf{v}_\mathrm{d}[k-1]T_s + \frac{1}{2}\mathbf{a}_\mathrm{d}[k-1]T_s^2 . \tag{4.5}$$

Thus, the aim of the OTG algorithm is to determine a suitable acceleration $\mathbf{a}_\mathrm{d}[k]$ for the current timestep k to continue the trajectory. By choosing the acceleration for each timestep, the trajectory is fully defined and the position and velocity values for the following timestep can be computed via integration using Eqs. (4.4) and (4.5). This procedure can be repeated iteratively to generate the trajectory.

Computing a new acceleration value from scratch at each timestep in general does not yield a trajectory with bounded jerk. Hence, to limit the jerk and obtain a sufficiently smooth trajectory, the desired acceleration values $\hat{\mathbf{a}}_\mathrm{d}[k]$ are computed and then filtered to obtain the actual acceleration values $\mathbf{a}_\mathrm{d}[k]$. Thus, the chosen acceleration value $\mathbf{a}_\mathrm{d}[k]$ is obtained by applying a filtering function $\mathbf{G}$ to the sequence of acceleration values $\hat{\mathbf{a}}_\mathrm{d}[j]$ where $j = 0, \dots, k$ in the form

$$\mathbf{a}_\mathrm{d}[k] = \mathbf{G}(\hat{\mathbf{a}}_\mathrm{d}[k], \hat{\mathbf{a}}_\mathrm{d}[k-1], \dots, \hat{\mathbf{a}}_\mathrm{d}[0]) \; . \tag{4.6}$$

Different kinds of filters and different filter orders can be applied to achieve the desired smoothness properties and achieve bounded derivatives of any desired order. Two different filtering methods are investigated in Section 4.3.3 and are implemented on the prototype robot.

The structure of the proposed OTG algorithm can be outlined by the following three steps which are repeated at each timestep k:

Step 1 Compute $\mathbf{r}_\mathrm{d}[k]$ and $\mathbf{v}_\mathrm{d}[k]$ via integration using Eqs. (4.4) and (4.5).

Step 2 Determine the desired acceleration for the current step $\hat{\mathbf{a}}_\mathrm{d}[k]$.

Step 3 Filter the acceleration $\hat{\mathbf{a}}_\mathrm{d}[k]$ to obtain $\mathbf{a}_\mathrm{d}[k]$ using Eq. (4.6).

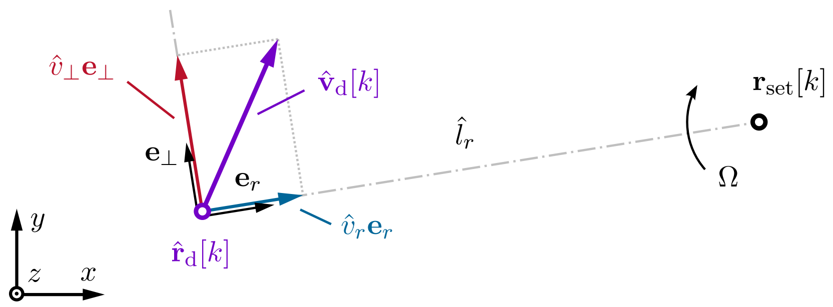
To find a suitable desired acceleration $\hat{\mathbf{a}}_\mathrm{d}[k]$ in Step 2, a series of assumptions and approximations is applied in the following. The proposed method for computing the acceleration $\hat{\mathbf{a}}_\mathrm{d}[k]$ can be divided into six steps which are described in detail in the following. To decouple the filtering operation applied in Step 3 from the computation of $\hat{\mathbf{a}}_\mathrm{d}[k]$, the algorithm uses the unfiltered trajectory velocity $\hat{\mathbf{v}}_\mathrm{d}[k]$ and the unfiltered position $\hat{\mathbf{r}}_\mathrm{d}[k]$ which are computed via integration from Eqs. (4.5) and (4.4), respectively. Thus, all quantities $(\cdot)$ denoted with the symbol $(\hat{\cdot})$ refer to the trajectory with unbounded jerk, which is generated independently. This trajectory is filtered to obtain a closely related trajectory with bounded jerk. In the following, the timestep k is omitted from all intermediate variables which are computed within each timestep but are not stored or used beyond a timestep.

I.) Local coordinate system

First, the velocity $\hat{\mathbf{v}}_\mathrm{d}[k]$ is separated into two components as depicted in Figure 4.1. A local coordinate system with the basis vectors $\mathbf{e}_r$ and $\mathbf{e}_\perp$ is defined, such that

$$\mathbf{e}_r = \frac{\mathbf{r}_\mathrm{set}[k] - \hat{\mathbf{r}}_\mathrm{d}[k]}{\|\mathbf{r}_\mathrm{set}[k] - \hat{\mathbf{r}}_\mathrm{d}[k]\|_2} \; , \qquad 0 = \mathbf{e}_\perp \cdot \mathbf{e}_r \; , \qquad \hat{\mathbf{v}}_\mathrm{d}[k] \cdot \mathbf{e}_\perp \geq 0 \; . \tag{4.7}$$

For the special case that $\hat{l}_r = \|\mathbf{r}_\mathrm{set}[k] - \hat{\mathbf{r}}_\mathrm{d}[k]\|_2 = 0$, the trajectory position equals the desired set-point position. If the velocity $\hat{\mathbf{v}}_\mathrm{d}[k] = \mathbf{0}$ at the same time, the target set-point is reached and the algorithm terminates. Otherwise, if $\hat{\mathbf{v}}_\mathrm{d}[k] \neq \mathbf{0}$, then $\mathbf{e}_r = \frac{\hat{\mathbf{v}}_\mathrm{d}[k]}{\|\hat{\mathbf{v}}_\mathrm{d}[k]\|_2}$ is used. Note that $\mathbf{e}_\perp$ is chosen such that it points in the direction of motion to ensure that $\hat{v}_\perp \geq 0$.



$\hat{v}_\perp \mathbf{e}_\perp$
$\hat{\mathbf{v}}_\mathrm{d}[k]$
$\mathbf{r}_\mathrm{set}[k]$
$\hat{l}_r$
$\mathbf{e}_\perp$
$\mathbf{e}_r$
Ω
y
$\hat{v}_r \mathbf{e}_r$
$\hat{\mathbf{r}}_\mathrm{d}[k]$
z
x

Figure 4.1: Splitting the velocity into radial ($\mathbf{e}_r$) and normal ($\mathbf{e}_\perp$) direction.

In the local coordinate system formed by $\mathbf{e}_\perp$ and $\mathbf{e}_r$, the velocity $\hat{\mathbf{v}}_\mathrm{d}(t)$ reads as

$$\hat{\mathbf{v}}_\mathrm{d}(t) = \hat{v}_\perp \mathbf{e}_\perp + v_r \mathbf{e}_r \ . \tag{4.8}$$

Note that the local coordinate system rotates with the angular velocity Ω given by

$$\Omega = \frac{\hat{v}_\perp}{\hat{l}_r} \ . \tag{4.9}$$

For this reason, the basis vectors $\mathbf{e}_\perp(t)$ and $\mathbf{e}_r(t)$ are functions of time. Thus, the total acceleration $\hat{\mathbf{a}}_\mathrm{tot}$ is obtained in the form

$$\hat{\mathbf{a}}_\mathrm{tot}(t) = \frac{\mathrm{d}\hat{\mathbf{v}}_\mathrm{d}}{\mathrm{d}t} = \underbrace{\frac{\mathrm{d}\hat{v}_\perp}{\mathrm{d}t}}_{\hat{a}_\perp} \mathbf{e}_\perp + \underbrace{\frac{\mathrm{d}\hat{v}_r}{\mathrm{d}t}}_{\hat{a}_r} \mathbf{e}_r + \hat{v}_\perp \underbrace{\frac{\mathrm{d}\mathbf{e}_\perp}{\mathrm{d}t}}_{\Omega \mathbf{e}_r} + \hat{v}_r \underbrace{\frac{\mathrm{d}\mathbf{e}_r}{\mathrm{d}t}}_{-\Omega \mathbf{e}_\perp} \ . \tag{4.10}$$

Substituting Eq. (4.9) into Eq. (4.10) yields the expression

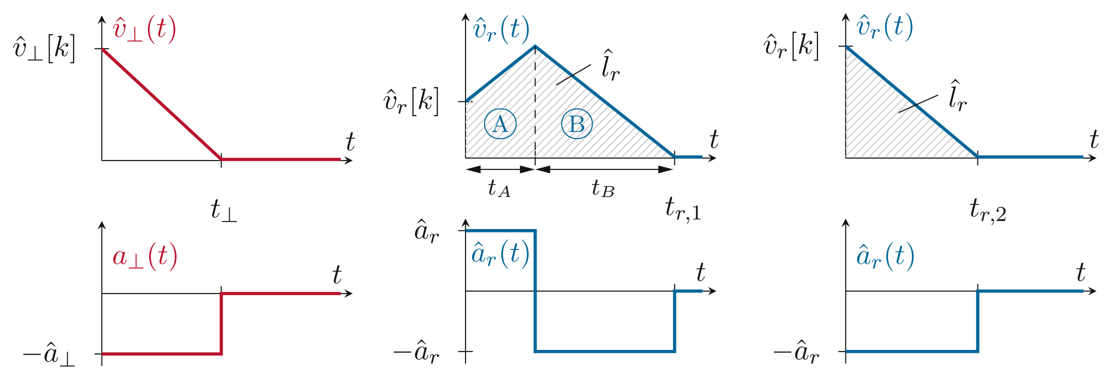
$$\hat{\mathbf{a}}_\mathrm{tot}(t) = \left(\hat{a}_\perp - \frac{\hat{v}_r \hat{v}_\perp}{\hat{l}_r}\right)\mathbf{e}_\perp + \left(\hat{a}_r + \frac{\hat{v}_\perp^2}{\hat{l}_r}\right)\mathbf{e}_r \ . \tag{4.11}$$

Due to the rotation of the local coordinate system, the additional acceleration terms $-\frac{\hat{v}_r \hat{v}_\perp}{\hat{l}_r}$ and $\frac{\hat{v}_\perp^2}{\hat{l}_r}$ in Eq. (4.11) arise in the non-inertial reference frame formed by $\mathbf{e}_\perp$ and $\mathbf{e}_r$.

II.) Profiles for velocity and acceleration

Second, the target profiles depicted in Figure 4.2 are chosen for the velocities $\hat{v}_\perp(t)$ and $\hat{v}_r(t)$ and the accelerations $\hat{a}_\perp(t)$ and $\hat{a}_r(t)$ within the local coordinate system.

In the normal direction $\mathbf{e}_\perp$, the profiles for $\hat{v}_\perp(t)$ and $\hat{a}_\perp(t)$ assume a linear reduction of the velocity until the velocity $\hat{v}_\perp$ reaches zero at the time $t_\perp$, as shown in Figure 4.2(a). For the radial direction $\mathbf{e}_r$, a triangular velocity profile for $\hat{v}_r(t)$ is assumed. Here, a distinction is made between the two cases visualized in Figures 4.2(b) and 4.2(c). In the first case shown in Figure 4.2(b), the radial velocity linearly increases in the acceleration phase labelled Ⓐ and subsequently linearly decreases in the braking phase labelled Ⓑ until the radial velocity $\hat{v}_r$ reaches a value of zero at the time $t_{r,1}$. During the motion, the



(a) Profiles in $\mathbf{e}_\perp$. (b) Acceleration case $\textcircled{I}$. (c) Braking case $\textcircled{II}$.

Figure 4.2: Velocity and acceleration profiles.

trajectory covers a distance of $\hat{l}_r$ in the direction $\mathbf{e}_r$ of the local coordinate system. This case is referred to as *acceleration case* $\textcircled{I}$ in the following.

In contrast, in the second case shown in Figure 4.2(c), only a linear decrease of the velocity $\hat{v}_r$ is assumed until it reaches zero at the time $t_{r,2}$. Similarly to the previous case, the trajectory covers a distance of $\hat{l}_r$ in direction $\mathbf{e}_r$ during the motion. Subsequently, this case is referred to as *braking case* $\textcircled{II}$.

Based on the assumed velocity profiles from Figure 4.2, the times $t_\perp$, $t_{r,1}$ and $t_{r,2}$ are computed in the following. The time $t_\perp$ depicted in Figure 4.2(a) depends on the velocity $\hat{v}_\perp[k]$ and the acceleration $\hat{a}_\perp$ and can be written in the form

$$t_\perp = \frac{\hat{v}_\perp[k]}{\hat{a}_\perp} \ . \tag{4.12}$$

The duration $t_{r,1}$ of the motion in the *acceleration case* $\textcircled{I}$ described by the profiles depicted in Figure 4.2(b) is fully defined by the starting velocity $\hat{v}_r[k]$, the acceleration $\hat{a}_r$ and the distance $\hat{l}_r$. Consider partitioning $t_{r,1}$ into the duration t_A of the acceleration phase $\textcircled{A}$ and the duration t_B of the braking phase $\textcircled{B}$, such that

$$t_{r,1} = t_A + t_B \ . \tag{4.13}$$

Because the velocity $\hat{v}_r$ must reach zero at the time $t_{r,1}$, the relation

$$\hat{v}_r[k] + t_A\hat{a}_r - t_B\hat{a}_r = 0 \tag{4.14}$$

holds. In addition, the covered distance $\hat{l}_r$ can be expressed as

$$\hat{l}_r = \hat{v}_r[k]t_A + \frac{1}{2}\hat{a}_r t_A^2 + \frac{1}{2}\hat{a}_r t_B^2 \ . \tag{4.15}$$

Combining Eqs. (4.13)-(4.15) leads to the expression

$$t_{r,1} = \frac{-\hat{v}_r[k] + \sqrt{4\hat{l}_r\hat{a}_r + 2\hat{v}_r^2[k]}}{\hat{a}_r} \ , \tag{4.16}$$

for the total duration of the motion.

The time $t_{r,2}$ required to complete the motion for the *braking case* (II) shown in Figure 4.2(c) and the corresponding acceleration $\hat{a}_r$ are fully defined by the velocity $\hat{v}_r[k]$ and the distance $\hat{l}_r$. In this case, the relations

$$t_{r,2} = \frac{2\hat{l}_r}{\hat{v}_r[k]} \;, \qquad\qquad \hat{a}_r = \frac{\hat{v}_r^2[k]}{2\hat{l}_r} \tag{4.17}$$

are obtained.

III.) Desired acceleration

Third, the previously derived durations $t_\perp$, $t_{r,1}$ and $t_{r,2}$ are utilized to calculate a suitable acceleration $\hat{\mathbf{a}}_\mathrm{d}[k]$. The *acceleration case* (I) and the *braking case* (II) are treated separately. In general, due to the acceleration limit a_max from Eq. (4.2), the velocity profiles from Figure 4.2 might not be feasible. When following the velocity profiles $\hat{v}_\perp(t)$ and $\hat{v}_r(t)$, the corresponding acceleration $\hat{\mathbf{a}}_\mathrm{d}(t)$ is given by Eq. (4.11). As explained previously, the additional acceleration terms $-\frac{\hat{v}_r\hat{v}_\perp}{\hat{l}_r}$ and $\frac{\hat{v}_\perp^2}{\hat{l}_r}$ in Eq. (4.11) arise due to the rotating reference frame formed by $\mathbf{e}_\perp$ and $\mathbf{e}_r$. These additional accelerations can be arbitrarily large because the distance $\hat{l}_r$ in the denominator can be arbitrarily small. Thus, the velocity and acceleration profiles from Figure 4.2 are used as approximations and the desired acceleration $\hat{\mathbf{a}}[k]$ is defined as

$$\hat{\mathbf{a}}_\mathrm{d}[k] = \hat{a}_\perp\mathbf{e}_\perp + \hat{a}_r\mathbf{e}_r \;, \tag{4.18}$$

by neglecting the nonlinear acceleration terms. The definition from Eq (4.18) is motivated by objective ***O3*** from Objectives 4.1. For the special case of straight motions, the velocity $\hat{v}_\perp \equiv 0$. Thus, in this case the nonlinear acceleration terms are identical to zero and the acceleration $\hat{\mathbf{a}}_\mathrm{d}$ defined in Eq (4.18) is identical to the acceleration $\hat{\mathbf{a}}_\mathrm{tot}$ from (4.11). As a result, the triangular velocity profiles from Figures 4.2(b) and 4.2(c) are feasible given the acceleration limit a_max. Because triangular velocity profiles are time-optimal for straight motions subject to acceleration constraints [20], objective ***O3*** is satisfied.

It can be shown, that for a time-optimal motion subject to acceleration and velocity constraints, either the velocity constraint or the acceleration constraint must be active at all times [48]. This means that the acceleration must be at its limit $\|\hat{\mathbf{a}}_\mathrm{d}\|_2 = a_\mathrm{max}$ unless the velocity limit is reached. For this reason, the relation

$$\hat{a}_\perp^2 + \hat{a}_r^2 = a_\mathrm{max}^2 \tag{4.19}$$

must hold. The acceleration components $\hat{a}_\perp$ and $\hat{a}_r$ must be chosen such that the motion satisfies objective ***O1***, i. e. that the trajectory does not spiral or oscillate. For this purpose, the deceleration $\hat{a}_\perp$ must be chosen sufficiently large such that Ω from Eq. (4.9) decreases sufficiently fast such that the trajectory does not spiral around the set-point $\mathbf{r}_\mathrm{set}$.

For the *acceleration case* (I), a choice of $\hat{a}_\perp$ and $\hat{a}_r$ is obtained by assuming that

$$t_\perp = \alpha_1\tilde{t}_{r,1} \;, \qquad\qquad \alpha_1 \leq 1 \;, \tag{4.20}$$

where $\tilde{t}_{r,1}$ denotes a lower bound for the duration $t_{r,1}$ and the dimensionless quantity α_1 is introduced as an additional damping. A lower bound for $t_{r,1}$ is given by

$$\tilde{t}_{r,1} = \min_{\hat{v}_r} \quad t_{r,1} = \sqrt{\frac{2\hat{l}_r}{\hat{a}_r}} \ . \tag{4.21}$$

Note that the lower bound $\tilde{t}_{r,1}$ corresponds to the special case of the *acceleration case* $\textcircled{I}$ visualized in Figure 4.2(b) where the starting velocity $\hat{v}_r$ has a value such that $t_A = 0$ and consequently the acceleration phase $\textcircled{A}$ vanishes. Substituting Eqs. (4.21), (4.20) and (4.12) into (4.19) and solving for $\hat{a}_r$ yields an expression for the acceleration case $\textcircled{I}$ denoted $\hat{a}_{r,1}$ in the form

$$\hat{a}_{r,1} = -\frac{\hat{v}_\perp[k]^2}{4\hat{l}_r\alpha_1^2} + \sqrt{a_{\max}^2 + \frac{\hat{v}_\perp^4[k]}{16\hat{l}_r^2\alpha_1^4}} \ . \tag{4.22}$$

By inserting Eq. (4.22) into Eq. (4.19) and rearranging, the component $\hat{a}_\perp$ can be calculated for the acceleration case $\textcircled{I}$ in the form

$$\hat{a}_{\perp,1} = -\sqrt{a_{\max}^2 - \hat{a}_r^2} \ . \tag{4.23}$$

Here, a suitable choice for $\hat{a}_\perp$ in the acceleration case $\textcircled{\hat{I}}$ is denoted $\hat{a}_{\perp,1}$. Hence, with Eqs. (4.22), (4.23) and (4.18), a closed-form expression for the desired acceleration $\hat{\mathbf{a}}_\mathrm{d}[k]$ for the acceleration case $\textcircled{I}$ is found.

For the *braking case* $\textcircled{II}$, the desired acceleration $\hat{a}_r$, given by Eq. (4.17) for following the profile from Figure 4.2(c) might not be feasible due to the acceleration limit $a_{\max}$. Note, that these cases can arise even for the special case of motions in a straight line, where $\hat{v}_\perp \equiv 0$. This corresponds to scenarios, where the set-point position $\mathbf{r}_\mathrm{set}[k]$ is too close to the current trajectory position $\hat{\mathbf{r}}_\mathrm{d}[k]$ and the current velocity component $\hat{v}_r[k]$ is too large to reach the set-point without overshooting.

To find an appropriate desired acceleration $\hat{a}_{\perp,2}$ for the direction $\mathbf{e}_\perp$, a similar strategy to the previous case is chosen. It is assumed that

$$t_\perp = \alpha_2 t_{r,2} \ , \qquad\qquad \alpha_2 \leq 1 \ . \tag{4.24}$$

Here, the dimensionless quantity α_2 is introduced to promote a sufficiently fast reduction of $\hat{v}_\perp$ to avoid spiralling motions. Substituting Eqs. (4.12) and (4.17) into Eq. (4.24) yields an expression for $\hat{a}_\perp$ in the form

$$\hat{a}_{\perp,2} = -\frac{\hat{v}_\perp[k]\hat{v}_r[k]}{2\hat{l}_r\alpha_2} \ . \tag{4.25}$$

A suitable component $\hat{a}_{\perp,2}$ for the braking case $\textcircled{II}$ can be calculated using Eq. (4.17).

To ensure that the chosen desired acceleration complies with the acceleration constraint from Eq. (4.2), a limit can be enforced in the form

$$\hat{\mathbf{a}}_\mathrm{d,lim} = \frac{\hat{\mathbf{a}}_\mathrm{d}}{\|\hat{\mathbf{a}}_\mathrm{d}\|_2}\min\{a_{\max}, \|\hat{\mathbf{a}}_\mathrm{d}\|_2\} \ . \tag{4.26}$$

By choosing sufficiently small constant values for α_1 and α_2, the algorithm satisfies Objectives 4.1. However, the required time for reaching the set-point can be reduced by adapting $\alpha_1[k]$ and $\alpha_2[k]$ at each timestep using the heuristic update law

$$\alpha_1[k] = \alpha_2[k] = \min\left(\frac{\|\hat{\mathbf{v}}[k]\|_2}{6\hat{v}_r[k]}, 1\right) . \tag{4.27}$$

Here, the damping factors $\alpha_1[k]$ and $\alpha_2[k]$ are varied between 1 and $\frac{1}{6}$ depending on $\hat{v}_r[k]$.

The motivation for the choice of Eq. (4.27) is to improve the behavior of the algorithm for small values of $\hat{v}_r[k]$. To obtain a closed-form expression for $\hat{a}_{r,1}$, the dependence on $\hat{v}_r$ was removed by using $\tilde{t}_{r,1}$ from Eq. (4.21) instead of $t_{r,1}$ from Eq. (4.16). However, for small values of $\hat{v}_r[k]$ the value $\tilde{t}_{r,1}$ is a poor approximation of $t_{r,1}$. This leads to the issue that the algorithm overly weighs braking in $\mathbf{e}_\perp$ direction instead of accelerating in $\mathbf{e}_r$ direction. To counteract this bias, the damping $\alpha_1[k]$ and $\alpha_2[k]$ can be increased for small values of $\hat{v}_r[k]$. While the algorithm performs well even for a constant choice of α_1 and α_2, the chosen law from Eq. (4.27) was found to significantly improve the behavior with respect to the motion time of the generated trajectories.

IV.) Velocity limit

So far, an appropriate acceleration vector was computed, which satisfies the acceleration constraint from Eq. (4.2) for both the *acceleration case* Ⓘ and the *braking case* Ⓘⓘ. In the following, the velocity limit is incorporated into the algorithm. If the velocity in the next timestep $\hat{\mathbf{v}}_\mathrm{d}[k+1]$ would violate the velocity limit v_max, the acceleration must be adapted to prevent this violation. The acceleration component $\hat{\mathbf{a}}_\perp$, acting in direction $\mathbf{e}_\perp$, always opposes the velocity component $\hat{\mathbf{v}}_\perp$. Thus, the component $\hat{\mathbf{a}}_\perp$ acts against any spiralling motion and is critical to satisfy objective ***O1***. At the same time, the component $\hat{\mathbf{a}}_\perp$ corresponds to a reduction of $\hat{v}_\perp$ and does not contribute to violating the velocity limit.

As a result, the possible violation of the velocity limit can be resolved by reducing the velocity component $\hat{\mathbf{a}}_r T_s$ while leaving the velocity component $\hat{\mathbf{a}}_\perp T_s$ unchanged. In case of a violation of the velocity limit in the next timestep $\|\hat{\mathbf{v}}_\mathrm{d}[k+1]\|_2 > v_\mathrm{max}$, the acceleration $\hat{a}_r$ can be chosen such that

$$\|\hat{\mathbf{v}}_\mathrm{d}[k+1]\|_2 = \|\hat{\mathbf{v}}_\mathrm{d}[k] + \hat{\mathbf{a}}_\perp T_s + \hat{a}_r \mathbf{e}_r T_s\|_2 = v_\mathrm{max} , \tag{4.28}$$

as visualized in Figure 4.3.

Considering the component $\hat{\mathbf{a}}_\perp$ fixed and solving Eq. (4.28) for $\hat{a}_r$ yields a quadratic equation which has two real solutions. Because the acceleration component $\hat{a}_r$ must be positive, the negative solution can be discarded. Thus, the solution for the acceleration component $\hat{a}_r$ can be written in the form

$$\hat{a}_r = \frac{1}{T_s}\left(-\mathbf{p}\cdot\mathbf{e}_r + \sqrt{(\mathbf{p}\cdot\mathbf{e}_r)^2 + v_\mathrm{max}^2 - \|\mathbf{p}\|_2^2}\right) , \qquad \mathbf{p} = \hat{\mathbf{v}}_\mathrm{d}[k] + \hat{\mathbf{a}}_\perp T_s . \tag{4.29}$$

The solution from Eq. (4.29) is visualized with green arrows in Figure 4.3 and the resulting limited velocity for the next timestep $\hat{\mathbf{v}}_\mathrm{d}[k+1]$ is depicted with a purple circle.

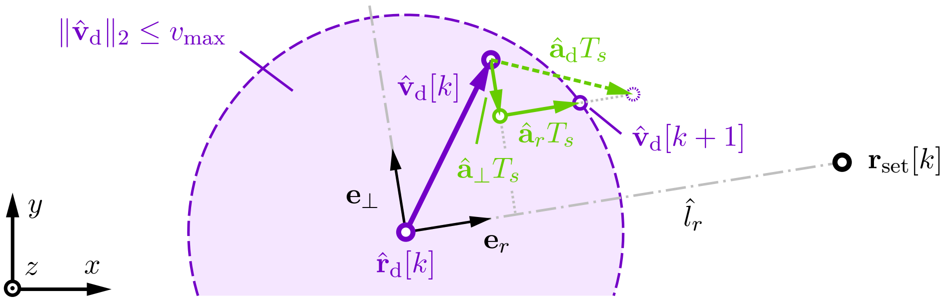

$\|\hat{\mathbf{v}}_\mathrm{d}\|_2 \leq v_\mathrm{max}$
$\hat{\mathbf{a}}_\mathrm{d} T_s$
$\hat{\mathbf{v}}_\mathrm{d}[k]$
$\hat{\mathbf{a}}_r T_s$
$\hat{\mathbf{v}}_\mathrm{d}[k+1]$
$\hat{\mathbf{a}}_\perp T_s$
$\mathbf{r}_\mathrm{set}[k]$
$\mathbf{e}_\perp$
$\hat{l}_r$
$\mathbf{e}_r$
$\hat{\mathbf{r}}_\mathrm{d}[k]$
y
z
x

Figure 4.3: Applying the velocity limit by reducing $\hat{\mathbf{a}}_r$.

V.) Switching between cases Ⓘ and Ⓘ Ⓘ

To switch between the previously discussed *acceleration case* Ⓘ and *braking case* (II), a suitable switching criterion is necessary. The criterion must decide if the set-point $\mathbf{r}_\mathrm{set}$ is sufficiently far away such that the trajectory should accelerate towards it or if it is necessary to brake to reach $\mathbf{r}_\mathrm{set}$ with zero velocity without overshooting, if possible.

The proposed switching criterion assumes at first that the *acceleration case* Ⓘ will be active. In this case, the desired acceleration $\hat{\mathbf{a}}_{\mathrm{d},1}$ is given by

$$\hat{\mathbf{a}}_{\mathrm{d},1}[k] = \hat{a}_{\perp,1}\mathbf{e}_\perp + \hat{a}_{r,1}\mathbf{e}_r \ , \tag{4.30}$$

where $\hat{a}_{r,1}$ can be calculated using Eq. (4.22) and $\hat{a}_{\perp,1}$ can be obtained from Eq. (4.23).

The idea behind the proposed switching method is to verify if using the *braking case* (II) will be possible in the next timestep $k+1$ without violating the acceleration limit a_max or not. In case the acceleration limit will be violated when applying the *braking case* (II) in the next timestep $k+1$, the *braking case* (II) must be chosen in the current timestep k to prevent overshooting the set-point $\mathbf{r}_\mathrm{set}$ in the future. Otherwise it is safe to choose the *acceleration case* Ⓘ for the current timestep k because braking at a later time will be sufficient.

Using the acceleration $\hat{\mathbf{a}}_{\mathrm{d},1}$ from Eq. (4.30), the velocity $\hat{\mathbf{v}}_\mathrm{d}[k+1]$ and the position $\hat{\mathbf{r}}_\mathrm{d}[k+1]$ in the next timestep read as

$$\hat{\mathbf{v}}_\mathrm{d}[k+1] = \hat{\mathbf{v}}_\mathrm{d}[k] + \hat{\mathbf{a}}_{\mathrm{d},1}T_s \ , \tag{4.31}$$

$$\hat{\mathbf{r}}_\mathrm{d}[k+1] = \hat{\mathbf{r}}_\mathrm{d}[k] + \hat{\mathbf{v}}_\mathrm{d}[k]T_s + \frac{1}{2}\hat{\mathbf{a}}_{\mathrm{d},1}T_s^2 \ . \tag{4.32}$$

A new local coordinate system defined by the basis vectors $\mathbf{e}_r[k+1]$ and $\mathbf{e}_\perp[k+1]$ can be created for the next timestep using Eq. (4.7). Separating $\hat{\mathbf{v}}_\mathrm{d}[k+1]$ into the components $\hat{v}_\perp[k+1]$ and $\hat{v}_r[k+1]$ and substituting into Eqs. (4.17) and (4.25) yields the desired acceleration $\hat{\mathbf{a}}_{\mathrm{d},2}[k+1]$ for braking in the next timestep in the form

$$\hat{\mathbf{a}}_{\mathrm{d},2}[k+1] = -\frac{\hat{v}_\perp[k+1]\hat{v}_r[k+1]}{2\hat{l}_r[k+1]\alpha_2}\mathbf{e}_\perp[k+1] + \frac{\hat{v}_r^2[k+1]}{2\hat{l}_r[k+1]}\mathbf{e}_r[k+1] \ . \tag{4.33}$$

The Euclidean norm of the required acceleration $\|\hat{\mathbf{a}}_{\mathrm{d},2}[k+1]\|_2$ reads as

$$\|\hat{\mathbf{a}}_{\mathrm{d},2}[k+1]\|_2 = \frac{|\hat{v}_r[k+1]|}{2\hat{l}_r[k+1]}\sqrt{\frac{\hat{v}_\perp^2[k+1]}{\alpha_2^2} + \hat{v}_r^2[k+1]}\ . \tag{4.34}$$

If the acceleration $\|\hat{\mathbf{a}}_{\mathrm{d},2}[k+1]\|_2 \leq a_{\max}$, the *acceleration case* $\textcircled{\small I}$ is selected for the current timestep k. Otherwise if $\|\hat{\mathbf{a}}_{\mathrm{d},2}[k+1]\|_2 > a_{\max}$, the *braking case* $\textcircled{\small II}$ is chosen for timestep k.

VI.) Termination

Because the proposed OTG algorithm computes the trajectory via numerical integration using Eqs. (4.4) and (4.5), small numerical errors can accumulate which can potentially cause the trajectory to oscillate when reaching the set-point and coming to a halt. To prevent these oscillations and enabling the trajectory to exactly reach the set-point even in the presence of numerical errors, a final termination condition is introduced. When the trajectory position is close enough to the set-point and the velocity is small enough to reach zero in the next sampling interval, the acceleration is chosen such that the velocity is identical to zero in the next iteration. Thus, the acceleration $\hat{\mathbf{a}}_\mathrm{d}[k]$ is chosen in the form

$$\hat{\mathbf{a}}_\mathrm{d}[k] = -\frac{\hat{\mathbf{v}}_\mathrm{d}[k]}{T_s}\ . \tag{4.35}$$

Summary

The complete algorithm derived and presented in this section is summarized in Algorithm 4.1. The individual operations of the algorithm are computationally inexpensive and straightforward to implement on a real-time computer system. The idea behind the algorithm is to compute a suitable acceleration $\hat{\mathbf{a}}_\mathrm{d}[k]$ at each timestep. Subsequently, the acceleration $\hat{\mathbf{a}}_\mathrm{d}$ is filtered to obtain a sufficiently smooth sequence of $\mathbf{a}_\mathrm{d}$, which fully defines the trajectory. For this purpose, the components of $\hat{\mathbf{a}}_\mathrm{d}[k]$ are computed in a local coordinate system. Thus, the choice of $\hat{\mathbf{a}}_\mathrm{d}[k]$ is transformed to a representation, where the acceleration is defined by the choice of the dimensionless damping quantities $\alpha_1[k]$ and $\alpha_2[k]$. These quantities regulate the convergence of the trajectory in normal direction and, thus, specify the trade-off between the deviation from a straight line and the convergence to the goal position.

Algorithm 4.1: Proposed online trajectory generator (OTG).

Input:	$\mathbf{r}_{\text{set}}[k]$	(desired set-point)
	$a_{\max}, v_{\max}$	(maximum acceleration, velocity)
	T_s	(sample time)
	$(\mathbf{r}_\text{d}, \mathbf{v}_\text{d}, \mathbf{a}_\text{d})[k-1]$	(previous trajectory values)
	$(\hat{\mathbf{r}}_\text{d}, \hat{\mathbf{v}}_\text{d}, \hat{\mathbf{a}}_\text{d})[k-1]$	(previous unfiltered values)
	$\mathbf{G}(\cdot) : \mathbb{R}^2 \to \mathbb{R}^2$	(filtering function)
Output:	$(\mathbf{r}_\text{d}, \mathbf{v}_\text{d}, \mathbf{a}_\text{d})[k]$	(current trajectory values)
	$(\hat{\mathbf{r}}_\text{d}, \hat{\mathbf{v}}_\text{d}, \hat{\mathbf{a}}_\text{d})[k]$	(current unfiltered values)

```
/* (Step 1) Compute $\mathbf{r}_\text{d}[k]$, $\hat{\mathbf{r}}_\text{d}[k]$ and $\mathbf{v}_\text{d}[k]$, $\hat{\mathbf{v}}_\text{d}[k]$ via integration */
$\mathbf{v}_\text{d}[k] = \mathbf{v}_\text{d}[k-1] + \mathbf{a}_\text{d}[k-1]T_s$ ; // See Eq. (4.4)
$\hat{\mathbf{v}}_\text{d}[k] = \hat{\mathbf{v}}_\text{d}[k-1] + \hat{\mathbf{a}}_\text{d}[k-1]T_s$ ; // See Eq. (4.4)
$\mathbf{r}_\text{d}[k] = \mathbf{r}_\text{d}[k-1] + \mathbf{v}_\text{d}[k-1]T_s + \frac{1}{2}\mathbf{a}_\text{d}[k-1]T_s^2$ ; // See Eq. (4.5)
$\hat{\mathbf{r}}_\text{d}[k] = \hat{\mathbf{r}}_\text{d}[k-1] + \hat{\mathbf{v}}_\text{d}[k-1]T_s + \frac{1}{2}\hat{\mathbf{a}}_\text{d}[k-1]T_s^2$ ; // See Eq. (4.5)

/* (Step 2) Determine the acceleration $\hat{\mathbf{a}}_\text{d}[k]$ */
// Create a local coordinate system
$\hat{l}_r \leftarrow \|\mathbf{r}_{\text{set}}[k] - \hat{\mathbf{r}}_\text{d}[k]\|_2$;
if $\hat{l}_r = 0$ then
    if $\hat{\mathbf{v}}_\text{d}[k] = \mathbf{0}$ then
        // The set-point is reached
        $\hat{\mathbf{a}}[k] \leftarrow \mathbf{0}$;
        Goto (Step 3);
    else
        $\mathbf{e}_r \leftarrow \frac{\hat{\mathbf{v}}_\text{d}[k]}{\|\hat{\mathbf{v}}_\text{d}[k]\|_2}$;
        Calculate $\mathbf{e}_\perp$ according to (4.7);
    end
else
    Calculate $\mathbf{e}_r$ and $\mathbf{e}_\perp$ according to (4.7);
end
$\hat{v}_r[k] \leftarrow \hat{\mathbf{v}}_\text{d}[k] \cdot \mathbf{e}_r$;
$\hat{v}_\perp[k] \leftarrow \hat{\mathbf{v}}_\text{d}[k] \cdot \mathbf{e}_\perp$;
// Compute the acceleration for acceleration case (I)
$\alpha_1 \leftarrow$ Choose damping value using Eq. (4.27);
$\hat{\mathbf{a}}_{r,1} \leftarrow$ Eq. (4.22);
$\hat{\mathbf{a}}_{\perp,1} \leftarrow$ Eq. (4.23);
$\hat{\mathbf{a}}_\text{d}[k] = \hat{a}_{\perp,1}\mathbf{e}_\perp + \hat{a}_{r,1}\mathbf{e}_r$; // See Eq. (4.30)
$\hat{\mathbf{v}}_\text{d}[k+1] \leftarrow \hat{\mathbf{v}}_\text{d}[k] + \hat{\mathbf{a}}_\text{d}[k]T_s$;
if $\|\hat{\mathbf{v}}_\text{d}[k+1]\|_2 > v_{\max}$ then
    // Limit acceleration so the velocity limit is not violated
    $\hat{a}_r \leftarrow$ Eq. (4.29);
    $\hat{\mathbf{a}}_\text{d}[k] = \hat{a}_{\perp,1}\mathbf{e}_\perp + \hat{a}_r\mathbf{e}_r$;
end
⋮
```

```
⋮
// Compute next velocity and position for the acceleration case (I)
$\hat{\mathbf{v}}_\mathrm{d}[k+1] \leftarrow \hat{\mathbf{v}}_\mathrm{d}[k] + \hat{\mathbf{a}}_\mathrm{d}[k]T_s$;
$\hat{\mathbf{r}}_\mathrm{d}[k+1] \leftarrow \hat{\mathbf{r}}_\mathrm{d}[k] + \hat{\mathbf{v}}_\mathrm{d}[k]T_s + \frac{1}{2}\hat{\mathbf{a}}_\mathrm{d}[k]T_s^2$;
Calculate $\mathbf{e}_r[k+1]$ and $\mathbf{e}_\perp[k+1]$ according to (4.7);
$\hat{v}_r[k+1] \leftarrow \hat{\mathbf{v}}_\mathrm{d}[k+1] \cdot \mathbf{e}_r[k+1]$;
$\hat{v}_\perp[k+1] \leftarrow \hat{\mathbf{v}}_\mathrm{d}[k+1] \cdot \mathbf{e}_\perp[k+1]$;
// Check if braking is still possible in the next step, if we choose
   the acceleration case (I) in the current step
$\alpha_2 \leftarrow$ Choose damping value using Eq. (4.27);
$a_\mathrm{thresh} \leftarrow \|\hat{\mathbf{a}}_{\mathrm{d},2}[k+1]\|_2$ from Eq. (4.34);
if $a_\mathrm{thresh} > a_\mathrm{max}$ then
    // Choose braking case (II)
    $\hat{a}_{\perp,2} \leftarrow$ Eq. (4.25);
    $\hat{a}_{r,2} \leftarrow$ Eq. (4.17);
    $\hat{\mathbf{a}}_\mathrm{d} \leftarrow \hat{a}_{r,2}\mathbf{e}_r + \hat{a}_{\perp,2}\mathbf{e}_\perp$;
    // Limit acceleration if necessary
    $\hat{\mathbf{a}}_\mathrm{d}[k] \leftarrow \frac{\hat{\mathbf{a}}_\mathrm{d}}{\|\hat{\mathbf{a}}_\mathrm{d}\|_2}\min\{a_\mathrm{max}, \|\hat{\mathbf{a}}_\mathrm{d}\|_2\}$ ; // See Eq. (4.26)
end
// Termination of the trajectory
if $\|\hat{\mathbf{v}}_\mathrm{d}[k]\|_2 \leq a_\mathrm{max}T_s$ and $\hat{l}_r \leq \frac{3}{2}a_\mathrm{max}T_s^2$ then
    // The set-point can be reached in the next timestep
    $\hat{\mathbf{a}}_\mathrm{d}[k] \leftarrow -\frac{\hat{\mathbf{v}}_\mathrm{d}[k]}{T_s}$ ; // See Eq. (4.35)
    if $\|\hat{\mathbf{v}}_\mathrm{d}[k]\|_2 = 0$ then
        // The set-point is reached
        $\hat{\mathbf{a}}_\mathrm{d}[k] \leftarrow \mathbf{0}$;
        $\hat{\mathbf{v}}_\mathrm{d}[k] \leftarrow \mathbf{0}$;
        $\hat{\mathbf{r}}_\mathrm{d}[k] \leftarrow \mathbf{r}_\mathrm{set}[k]$;
    end
end

/* (Step 3) Filter $\hat{\mathbf{a}}_\mathrm{d}$ to obtain $\mathbf{a}_\mathrm{d}$                                  */
$\mathbf{a}_\mathrm{d}[k] \leftarrow \mathbf{G}(\hat{\mathbf{a}}_\mathrm{d}[k])$ ; // Apply filtering function
```

4.3.2 Rotation

Because the CDPR possesses one rotational degree-of-freedom (DOF), the trajectory planning problem is 1-dimensional. This case can be interpreted as a special case of the 2-dimensional trajectory planning problem and the algorithm presented in Section 4.3.1 can be employed. In this case, any quantity $(\cdot)$ in the normal direction denoted $(\cdot)_\perp$ is identical to zero. Thus, the unfiltered trajectory with unbounded jerk is always time optimal as stated by objective ***O3*** from Objectives 4.1.

4.3.3 Filtering

The proposed OTG algorithm uses a filter in $\boxed{\text{Step 3}}$, to transform the trajectory with unbounded jerk, given by $(\hat{\mathbf{a}}_\mathrm{d}, \hat{\mathbf{v}}_\mathrm{d}, \hat{\mathbf{r}}_\mathrm{d})$ to a trajectory with limited jerk given by $(\mathbf{a}_\mathrm{d}, \mathbf{v}_\mathrm{d}, \mathbf{r}_\mathrm{d})$. Depending on the filter type and the filter order applied in this step, the desired smoothness, i. e. the number of continuous derivatives of the trajectory, can be chosen. In the following, two different filters are discussed, which are implemented on the prototype robot.

Moving average FIR filter

To achieve time-optimal motion with limited jerk, a moving average FIR filter can be used [42] to filter the acceleration $\hat{\mathbf{a}}_\mathrm{d}$. This filter converts a rectangular acceleration profile to a trapezoidal acceleration profile. To ensure time-optimality with an arbitrary acceleration profile, the filter length must be dynamically adapted as discussed in detail in [42]. In the present work, a moving average FIR filter with constant length is employed. The filter length is chosen in such a way, that the desired jerk j_step is reached, for a step-change from $\hat{\mathbf{a}}_\mathrm{d}[k-1] = \mathbf{0}$ to $\|\hat{\mathbf{a}}_\mathrm{d}[k]\|_2 = a_\mathrm{max}$. Thus, the static filter length n_FIR is given by the relation

$$N_\mathrm{FIR} = \left\lceil \frac{a_\mathrm{max}}{j_\mathrm{step} T_s} \right\rceil , \tag{4.36}$$

where $\lceil\cdot\rceil$ denotes the ceiling operator which rounds to the next full integer. The resulting moving average FIR filter can be defined via its impulse response matrix $\mathbf{H}[k]$ which reads as

$$\mathbf{H}[k] = \begin{cases} \frac{1}{N_\mathrm{FIR}}\mathbf{I}, & 0 \leq k < N_\mathrm{FIR} \\ \mathbf{0}, & \text{otherwise} , \end{cases} \tag{4.37}$$

where $\mathbf{I}$ denotes the identity matrix. The behavior of this filter is presented and discussed via simulations in Section 4.4.4.

IIR filter

To achieve a smooth motion and further reduce oscillations in the robot drivetrain, an IIR filter of order n can be employed alternatively. This filter smooths the acceleration profile and ensures that the filtered acceleration $\mathbf{a}_\mathrm{d}$ is $n-1$ times continuously differentiable. Such a filter can be designed using the continuous-time transfer function

$$G(s) = \frac{c_0}{s^n + c_{n-1}s^{n-1} + \cdots + c_1 s + c_0} . \tag{4.38}$$

Here, the filter coefficients c_i can be calculated by choosing appropriate poles for the transfer function from Eq. (4.38). To implement the filter on the robot controller, the continuous-time transfer function $G(s)$ can be converted to a continuous-time state space system and subsequently discretized. Thus, a discrete-time state-space system is obtained which can be easily implemented on the robot control system. In the following, the behavior of an IIR filter with $n = 3$ is investigated in the Section 4.4.4.

4.4 Simulation results

To assess the effectiveness of the proposed OTG algorithm, simulations for different scenarios are conducted to investigate the behavior of the algorithm. First, in Section 4.4.1, the behavior and performance of the OTG algorithm for generating a trajectory with different initial velocities is investigated. Second, the impact of the kinematic limits $v_{\max}$ and $a_{\max}$ on the behavior of the trajectories is studied in Section 4.4.2. Third, the temporal profiles of velocity and acceleration are examined in Section 4.4.3. Fourth, in Section 4.4.4 the behavior of the two different filters outlined in Section 4.3.3 is discussed.

The proposed OTG algorithm initially generates an acceleration-limited trajectory $(\hat{\mathbf{r}}_\mathrm{d}, \hat{\mathbf{v}}_\mathrm{d}, \hat{\mathbf{a}}_\mathrm{d})$ in $\boxed{\text{Step 1}}$ and $\boxed{\text{Step 2}}$ of the algorithm and subsequently uses a filter in $\boxed{\text{Step 3}}$ to obtain a jerk-limited trajectory $(\mathbf{r}_\mathrm{d}, \mathbf{v}_\mathrm{d}, \mathbf{a}_\mathrm{d})$. For this reason, the acceleration-limited and jerk-limited trajectories are studied separately. In Sections 4.4.1, 4.4.2 and 4.4.3, the filtering step conducted in $\boxed{\text{Step 3}}$ of the algorithm is omitted and the acceleration-limited trajectories are investigated. Subsequently, in Section 4.4.4 the effect of the filtering in $\boxed{\text{Step 3}}$ is discussed.

For all simulations conducted in the following, the proposed OTG algorithm operates using a sampling frequency of $f_s = 8\,\mathrm{kHz}$. This sampling frequency is also used by the robot controller of the prototype hardware.

4.4.1 Initial velocity

To study the behavior of the proposed OTG algorithm, different motions in 2D Euclidean space with different initial conditions are compared to each other. This corresponds to adapting a trajectory during a motion, where the velocity is non-zero and the adapted target set-point causes the trajectory to change the direction of motion. In this section, the acceleration-limited trajectories $(\hat{\mathbf{r}}_\mathrm{d}, \hat{\mathbf{v}}_\mathrm{d}, \hat{\mathbf{a}}_\mathrm{d})$ generated by the algorithm are examined and the filtering conducted in $\boxed{\text{Step 3}}$ is omitted.

To investigate the performance and quality of the trajectories generated by the proposed algorithm, the time-optimal trajectories are used as a reference for comparison. The time-optimal solution for each motion is computed offline using the `Rockit` numerical framework based on `CasADi` and `IPOPT`. For this purpose, the trajectory generation problem stated in Problem 4.1 is formulated as a dynamic optimization problem in the form

$$\begin{aligned} \min_{\mathbf{a}(t)} \quad & \int_0^T 1 dt \\ \text{s.t.} \quad & \dot{\mathbf{r}} = \mathbf{v}, \qquad \mathbf{r}(0) = \mathbf{r}_0, \qquad \mathbf{r}(T) = \mathbf{r}_{\text{set}}, \\ & \dot{\mathbf{v}} = \mathbf{a}, \qquad \mathbf{v}(0) = \mathbf{v}_0, \qquad \mathbf{v}(T) = \mathbf{0}, \\ & \|\mathbf{v}\|_2 \leq v_{\max}, \\ & \|\mathbf{a}\|_2 \leq a_{\max} \,. \end{aligned} \tag{4.39}$$

Here, the vectors $\mathbf{r}, \mathbf{v} \in \mathbb{R}^2$. The optimal solution $\mathbf{a}^*(t)$ and the corresponding time-optimal trajectory $\mathbf{r}^*(t), \mathbf{v}^*(t)$ of the optimal control problem from Eq. (4.39) are computed numerically on a grid of 200 points in time. The required computation time for each optimal trajectory ranges from approximately 0.5 s to 5 s. Hence, the optimal solution cannot be used in the real-time environment but only offline for reference.

Figure 4.4 shows the geometric path of the trajectories for different starting velocities in the xy-plane. Here, the trajectories generated by the proposed OTG algorithm are shown with solid lines while the optimal solution is depicted using dash-dotted lines for comparison. The initial velocity $\mathbf{v}_0$ is chosen as

$$\mathbf{v}_0 = \begin{bmatrix} \cos\alpha_0 \\ \sin\alpha_0 \end{bmatrix} \text{m/s} \,, \tag{4.40}$$

with different angles $\alpha_0 \in [0, \pi]$. The boundary conditions and the constraints are chosen as

$$\mathbf{r}_0 = \mathbf{0}\text{m} \,, \qquad \mathbf{r}_{\text{set}} = [1, 0]^{\text{T}}\text{m} \,, \qquad v_{\max} = 1\,\text{m/s} \,, \qquad a_{\max} = 1\,\text{m/s}^2 \,, \tag{4.41}$$

for the trajectories presented in Figure 4.4. The geometric paths of the trajectories, which are generated by the proposed OTG algorithm are generally similar to the optimal paths. For all starting angles, the proposed OTG algorithm produces smooth trajectories without any overshooting or unnecessary spiral motion. For the straight path with $\alpha_0 = 180°$, the trajectory generated by the OTG algorithm matches the optimal trajectory. To quantify the performance with respect to the time required to reach the target set-point $\mathbf{r}_{\text{set}}$, the optimal time for each trajectory t_{opt} is compared to the necessary time t_{OTG} for completing the trajectory generated by the OTG algorithm. The relative excess time Δt_{rel} is defined as

$$\Delta t_{\text{rel}} = \frac{t_{\text{OTG}} - t_{\text{opt}}}{t_{\text{opt}}} \,, \tag{4.42}$$

and computed for a range of trajectories with different starting angles as defined in Eq. (4.40). The resulting relative excess time in relation to the starting angle α_0 is visualized in Figure 4.5. The results from Figure 4.5 show that the OTG algorithm generates time-optimal motion profiles for straight motions i. e. for the cases $\alpha_0 = 0°$ and $\alpha_0 = 180°$. For all other motions, the resulting trajectory requires around 3-5% more time than the optimal trajectory to complete each motion.

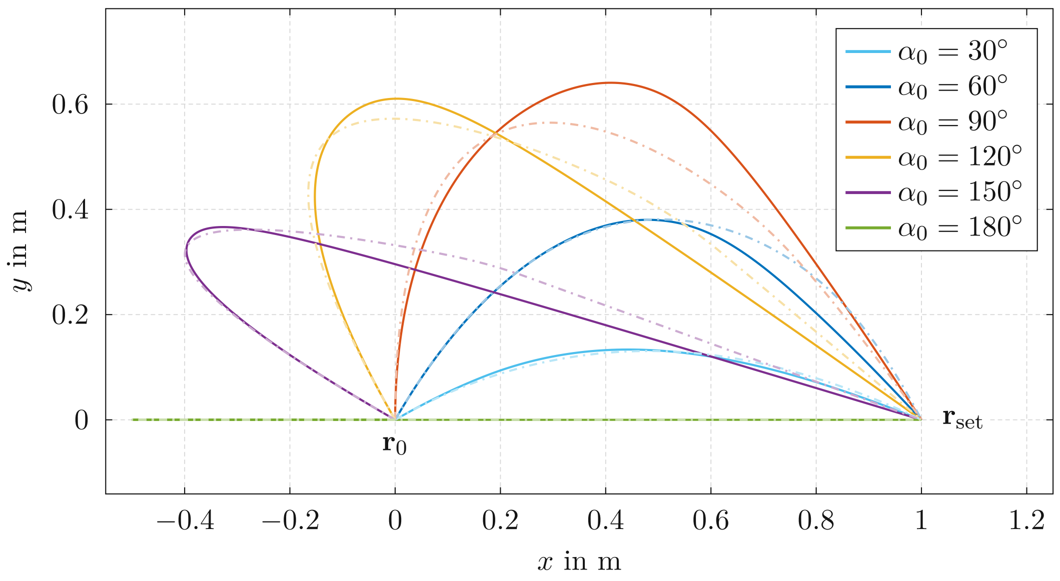


Figure 4.4: Trajectories for different starting angles. Proposed OTG algorithm (solid lines) compared to the optimal solution (dash-dotted lines).

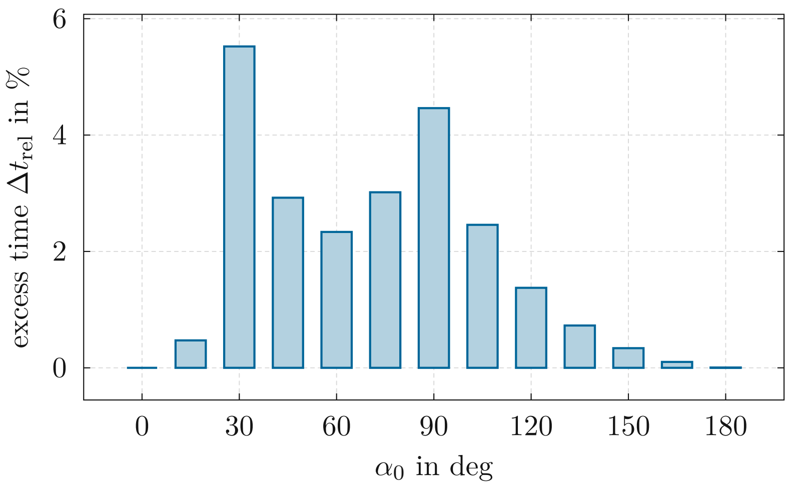
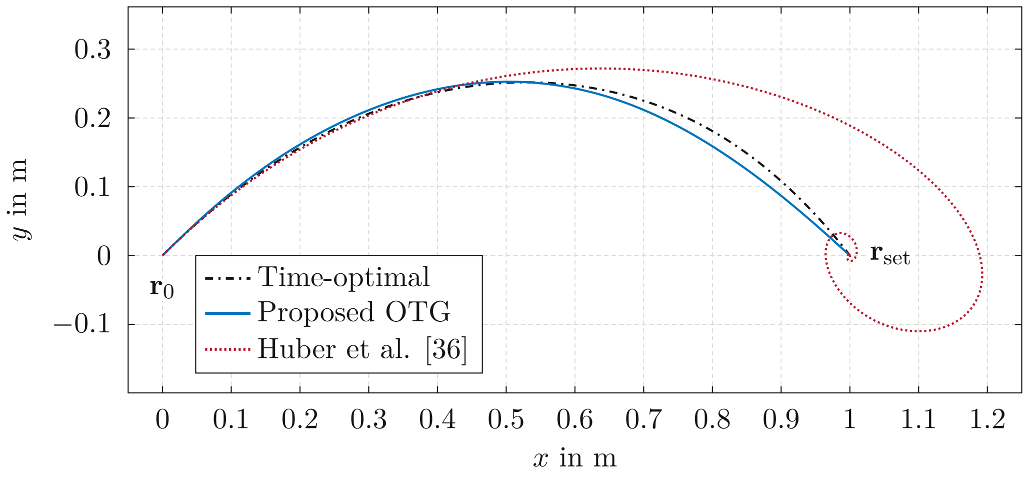


Figure 4.5: Relative excess time required by the OTG algorithm.

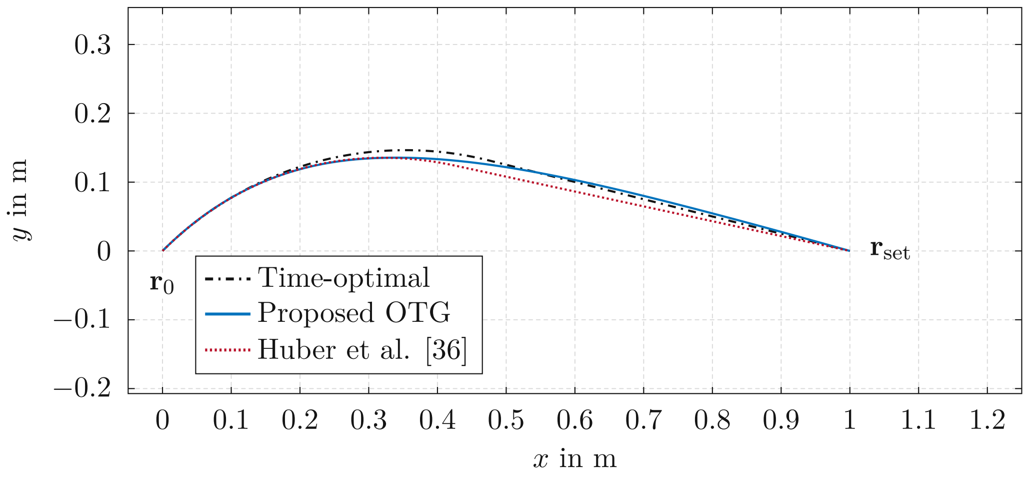
4.4.2 Kinematic limits

In this section, the behavior of the OTG algorithm and the resulting trajectory for different kinematic limits $v_{\max}$ and $a_{\max}$ is investigated. The proposed OTG is compared to the time-optimal solution of the optimization problem Eq. (4.39) and the OTG algorithm proposed by Huber et al. [36], found in the literature. The filtering in Step 3 of the algorithm is omitted for a better comparison.

A comparison of the generated trajectories with two different acceleration limits $a_{\max}$ is shown in Figure 4.6. The initial velocity for both motions is chosen with a magnitude of $1\,\mathrm{m/s}$ and an angle of $\alpha_0 = 45°$, see Eq. (4.40). For the first motion, depicted in



(a) Trajectories for $\alpha_0 = 45°$, $v_{\max} = 1\,\mathrm{m/s}$ and $a_{\max} = 1\,\mathrm{m/s^2}$.



(b) Trajectories for $\alpha_0 = 45°$, $v_{\max} = 1\,\mathrm{m/s}$ and $a_{\max} = 2\,\mathrm{m/s^2}$.

Figure 4.6: Comparison of the behavior with different acceleration limits.

Figure 4.6(a), the constraints are chosen as stated in Eq. (4.41). For this choice, the velocity constraint $\|\mathbf{v}\|_2 \leq v_{\max}$ is never active for any of the generated trajectories. Thus, the motion in this case is dominated by the acceleration constraint $\|\mathbf{a}\|_2 \leq a_{\max} = 1\,\mathrm{m/s^2}$.

For the second motion shown in Figure 4.6(b), the acceleration limit $a_{\max} = 2\,\mathrm{m/s^2}$ is chosen. For this choice, the trajectories reach the velocity limit $v_{\max}$ and as a result the motion in this case is dominated by the velocity constraint. The results visualized in Figure 4.6 demonstrate the different behavior of the trajectories for motions dominated by the acceleration constraint compared to motions dominated by the velocity constraint. When the acceleration limit is the dominant limit, the time-optimal trajectory $\mathbf{r}^*(t)$ describes an arc with relatively constant curvature. Towards the end of the motion, close to the target set-point, the curvature of the optimal trajectory slightly increases as the velocity is reduced to reach the target set-point with zero velocity. This behavior is visible in Figure 4.6(a). On the other hand, for a motion which is dominated by the velocity constraint, the time-optimal trajectory exhibits a larger curvature at the beginning of the trajectory and subsequently extends almost in a straight path towards the set-point. This is a result of reaching the velocity limit during the motion. When the velocity limit is reached, the optimal trajectory $\mathbf{r}^*(t)$ comprises paths with lower curvature which reduce the distance covered by the trajectory. This behavior can be observed in Figure 4.6(b).

For both, a dominating acceleration limit and a dominating velocity limit, the trajectory generated by the proposed OTG algorithm shows small deviations from the optimal trajectory. In comparison, the algorithm proposed by Huber et al. generates a trajectory which spirals around the target set-point when the motion is dominated by the acceleration limit as shown in Figure 4.6(a). In the presence of a sufficiently small velocity limit, the algorithm proposed by Huber et al. is well-behaved and produces trajectories which are close to the optimal trajectory as depicted in Figure 4.6(b). It should be noted that the primary focus of the work presented by Huber et al. lies in planning algorithms for both position and orientation of a robot in a human-robot interaction scenario where the velocity limit is usually dominant. Thus, the algorithm can be used in a range of applications even despite generating spiralling trajectories for motions without a sufficiently small velocity limit. However, for the dart catching application which is investigated in the present work, the OTG algorithm proposed by Huber et al. is not suitable.

4.4.3 Profiles of velocity and acceleration

In this section, the behavior of the velocities and accelerations of the trajectories generated by the proposed OTG algorithm are analyzed in more detail. For this purpose, the boundary conditions and constraints from Eq. (4.41) are used and the starting velocity is chosen with a magnitude of $1\,\mathrm{m/s}$ and an angle of $120°$ as defined in Eq. (4.40). The optimal solution $(\mathbf{a}^*, \mathbf{v}^*, \mathbf{r}^*)$ of the corresponding dynamical optimization problem from Eq. (4.39) is computed numerically and compared to the acceleration limited trajectory $(\hat{\mathbf{a}}, \hat{\mathbf{v}}, \hat{\mathbf{r}})$, which is generated by the proposed OTG algorithm. Here, the filtering in Step 3 of the algorithm is omitted.

Figure 4.7 visualizes the results for the motion. Here, the geometric path of the optimal solution $\mathbf{r}^*$ is compared to the geometric path $\hat{\mathbf{r}}$ generated by the proposed OTG in Figure 4.7(a). In addition, the corresponding acceleration vectors $\mathbf{a}^*$ and $\hat{\mathbf{a}}$ are shown at equidistant times as arrows in the xy-plane. To compare the temporal evolution of the time-optimal velocity vector $\mathbf{v}^*$ to the evolution of the velocity vector $\hat{\mathbf{v}}$ generated by the proposed OTG algorithm, the Euclidean norm and the angle α_v is computed for both vector quantities. Here, the angle α_v of the vector quantity $\mathbf{v}$ in the xy-plane in defined as

$$\alpha_v = \text{atan2}(v_y, v_x) \tag{4.43}$$

using the four-quadrant arc-tangent function, [20]. The velocity profiles $\|\mathbf{v}^*\|_2$ and $\|\hat{\mathbf{v}}\|_2$ are depicted in Figure 4.7(b), while the profiles of α_{v^*} and $\alpha_{\hat{v}}$ are shown in Figure 4.7(c). Similarly, the acceleration profiles $\|\mathbf{a}^*\|_2$ and $\|\hat{\mathbf{a}}\|_2$ are visualized in Figure 4.7(d) whereas the profiles of α_{a^*} and $\alpha_{\hat{a}}$ are shown in Figure 4.7(e). The results for the present motion demonstrate the behavior and subtle nuances of the optimal solution of the control problem from Eq. (4.39). The motion can be divided into 4 sections which are labelled ① - ④ in Figure 4.7. A necessary condition for a time-optimal motion subject to kinematic constraints is, that at least one constraint is always active [48]. In other words, the optimal motion will always keep the control variable i. e. acceleration at its limit $\|\mathbf{a}^*\|_2 = a_{\max}$ until a different state, i. e. the velocity $\mathbf{v}^*$, reaches the corresponding limit. For the present motion, it is possible to steer the point mass in such a way, that the acceleration stays at its limit even when the velocity reaches the limit $\|\mathbf{v}^*\|_2 = v_{\max}$. To achieve this, the optimal trajectory $\mathbf{r}^*$ follows a path which initially curves with a relatively small radius of curvature in section ①, as depicted in Figure 4.7(a), while the velocity $\|\mathbf{v}^*\|_2$ decreases as shown in Figure 4.7(b). When the velocity is at its lowest magnitude between ① and ②, the rate of change of the direction α_{v^*} reaches its maximum value as visible in Figure 4.7(c).

Subsequently, in Section ②, the velocity of the optimal trajectory increases and the rate of change of the direction α_{v^*} decreases as the trajectory prematurely stops changing the direction of motion towards the target set-point $\mathbf{r}_{\text{set}}$. In section ③, the velocity limit $\|\mathbf{v}^*\|_2 = v_{\max}$ is reached by the optimal trajectory, but the acceleration $\|\mathbf{a}^*\|_2$ stays at the limit value to complete the turn towards the target set-point and to continue changing the direction of motion α_{v^*}. Thus, the direction α_{a^*} of the acceleration changes direction in Section ③. Finally, in Section ④, the velocity along the optimal trajectory reduces to reach the target set-point $\mathbf{r}_{\text{set}}$ with zero velocity. This is achieved by rapidly changing the direction α_{a^*} of the acceleration.

In summary, reaching the velocity limit $v_{\max}$ must be anticipated to generate the time-optimal trajectory. As a consequence, the optimal trajectory follows a path which is not oriented towards the set-point in Sections ① and ②. Only when the velocity limit is reached in Section ③, the velocity of the optimal trajectory is oriented towards the set-point.

Due to its structure, the proposed OTG algorithm does not anticipate reaching the velocity limit and consequently cannot generate the complex maneuver which is conducted by the time-optimal trajectory. Thus, the magnitude of the acceleration $\|\hat{\mathbf{a}}\|_2$ is reduced by the algorithm, when the velocity limit $\|\hat{\mathbf{v}}\|_2 = v_{\max}$ is reached. Despite this limitation, the trajectory generated by the proposed OTG qualitatively approximates the time-optimal trajectory well and produces a time-efficient sub-optimal trajectory.

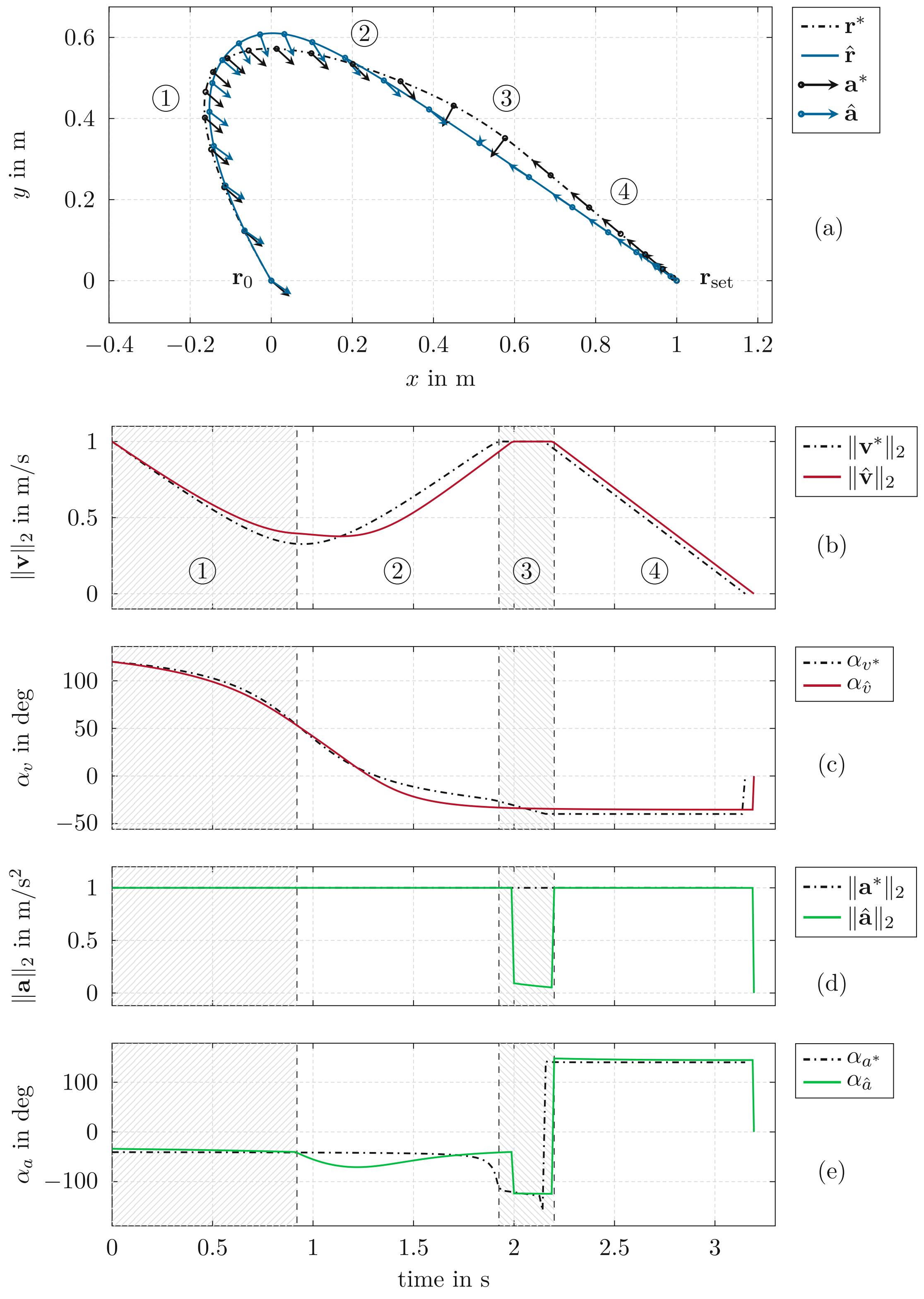


Figure 4.7: Trajectory for $\alpha_0 = 120°$, $v_{\max} = 1\,\text{m/s}$ and $a_{\max} = 1\,\text{m/s}^2$.

4.4.4 Filter behavior

In this section, the impact of the filtering performed in $\boxed{\text{Step 3}}$ of the proposed OTG algorithm is discussed. For this purpose, the two filter types outlined in Section 4.3.3 are parameterized such that they are suitable for use in the dart catching application using the prototype robot. More analysis on designing and using filtering algorithms to obtain smooth trajectories can be found in [41–43].

The linear time-invariant filters are applied to the acceleration $\hat{\mathbf{a}}_\mathrm{d}$ to obtain $\mathbf{a}_\mathrm{d}$ in $\boxed{\text{Step 3}}$ of the proposed OTG algorithm. The corresponding velocities $\hat{\mathbf{v}}, \mathbf{v}$ and positions $\hat{\mathbf{r}}, \mathbf{r}$ are computed via integration using Eqs. (4.4) and (4.5) in $\boxed{\text{Step 1}}$. Due to linearity, the order of these operations could also be swapped without changing the results. In other words, integrating first and then filtering yields the same results as filtering first and then integrating.

To analyze the filter behavior, a typical motion for catching a dart is chosen. Here, the boundary conditions and kinematic limits are chosen as

$$\mathbf{r}_0 = \mathbf{0}\mathrm{m}\ , \qquad \mathbf{r}_\mathrm{set} = [0.3, 0]^\mathrm{T}\mathrm{m}\ , \qquad v_\mathrm{max} = 2.5\,\mathrm{m/s}\ , \qquad a_\mathrm{max} = 30\,\mathrm{m/s}^2\ , \tag{4.44}$$

which corresponds to a motion of the dartboard by more than a full radius of the dartboard in x-direction. This is equivalent of catching a dart, which would miss the dartboard, such that it hits the bulls-eye. The FIR filter from Eq. (4.37) is used with a filter length N_FIR given by Eq. (4.36). The desired jerk is chosen as $j_\mathrm{step} = 3000\,\mathrm{m/s}^3$, which results in a filter length of $N_\mathrm{FIR} = 80$ samples at a sampling frequency of $f_s = 8\,\mathrm{kHz}$. Thus, the moving average FIR filter averages over a period of 10 ms.

To compute the coefficients c_i for the IIR filter from Eq. (4.38), a filter order $n = 3$ is selected and the poles of the transfer function $G(s)$ are chosen as

$$p_1 = p_2 = p_3 = -400\,\mathrm{s}^{-1}\ . \tag{4.45}$$

The resulting profiles of velocity v, acceleration a and jerk j in x-direction are compared to each other in Figure 4.8. Here, the values for the jerk $j[k]$ are computed from the acceleration values $a[k]$ at each timestep k using the backwards difference quotient

$$j[k] = \frac{a[k] - a[k-1]}{T_s}\ , \tag{4.46}$$

with the sample time $T_s = \frac{1}{f_s} = 125\,\mathrm{\mu s}$. Figure 4.8(a) depicts the velocity profiles generated by the proposed OTG for different filters. The unfiltered velocity profile $\hat{v}_x$ has a trapezoidal shape, while the velocities of the filtered trajectories $v_{x,FIR}$ and $v_{x,IIR}$ are smoothed by the respective filtering operations. The acceleration profiles generated by the algorithm are shown in Figure 4.8(b). Here, the unfiltered acceleration $\hat{a}_x$ is piece-wise constant and switches between the limit a_max, zero, and $-a_\mathrm{max}$. The filtered accelerations $a_{x,FIR}$ and $a_{x,IIR}$ demonstrate the step-response of the FIR and IIR filters, respectively. The FIR filter has a trapezoidal step response while the third order IIR filter exhibits third order low-pass behavior. With the chosen parameters for the filters, the rise-time of the FIR filter is 10 ms while the rise-time of the IIR filter has a value of approximately

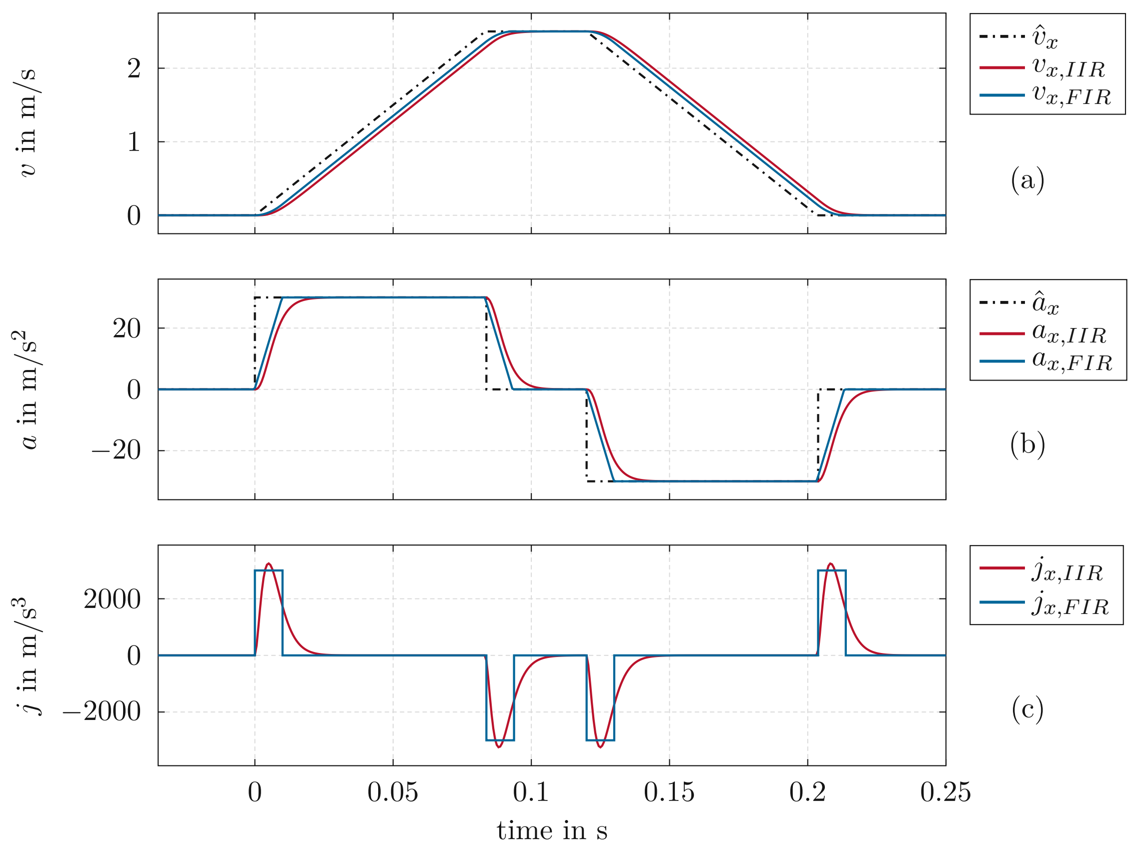


Figure 4.8: Behavior of different filters.

20 ms. Consequently, the motion requires an additional time of 10 ms when applying the FIR filter and an additional time of approximately 20 ms in the case of the IIR filter.

Figure 4.8(c) visualizes the jerk profiles of the motion when using the FIR and IIR filters, respectively. These profiles correspond to the impluse responses of the filters. The jerk $j_{x,FIR}$ has the shape of rectangular pulses while the jerk pulses exhibit a smoother shape $j_{x,IIR}$. For the chosen filter parameters, the peak jerk values when using the FIR filter and the IIR filter are similar.

5 Input shaping for vibration reduction

5.1 Motivation

Due to its mechanical design, the experimental prototype CDPR setup shown in Figure 1.1 is prone to structural vibrations. These vibrations are excited by the reaction forces caused by the rapid acceleration of the tournament dartboard, which is attached to the EE. The natural resonant frequencies of the structural components of the robot are investigated and measured in Section 7.3, where one dominant eigenfrequency at approximately 6 Hz can be observed. This resonant frequency corresponds to a sideways oscillation mode of the robot frame in x-direction and is a result of the large height to width ratio of the robot structure and the comparably low stiffness of the support structure. The aspect ratio of the robot structure was chosen to satisfy the design objectives of a lightweight mobile robot that can catch darts at the official tournament regulation height. Figure 5.1 shows the discrete Fourier transform (DFT) of the acceleration for the motion generated by the trajectory generator presented in Section 4.3 for the boundary conditions and limits

$$\mathbf{r}_0 = \mathbf{0}\mathrm{m}\ , \qquad \mathbf{r}_{\mathrm{set}} = [0.1, 0]^{\mathrm{T}}\mathrm{m}\ , \qquad v_{\max} = 2.5\,\mathrm{m/s}\ , \qquad a_{\max} = 30\,\mathrm{m/s^2}\ . \tag{5.1}$$

The motion defined by Eq. (5.1) is a typical positioning task for catching a dart.

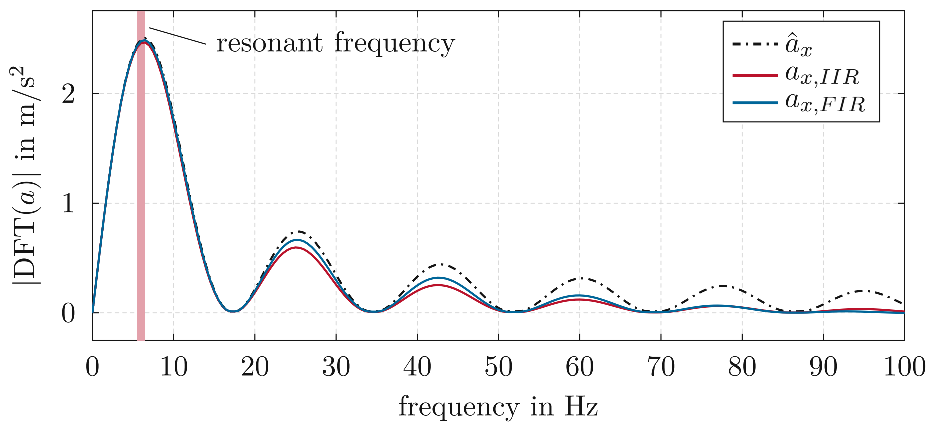


Figure 5.1: DFT of the acceleration for a typical motion.

The DFT of the accelerations depicted in Figure 5.1 show that the filters presented in Section 4.3.3 which are used to limit the jerk along the trajectory effectively reduce high frequency components in the acceleration of the EE. However, the primary frequency content in the range of 0 Hz-15 Hz is almost unaffected by this filtering. In addition, the dominant structural resonance frequency (6 Hz) lies in this range and is therefore strongly excited by the EE motion.

For this reason, it is desirable to reduce the mechanical oscillations via a suitable control method. In general, a vibration reduction can be achieved either by feedback control or by feedforward control of the EE motion [49]. For feedback vibration control, the vibration of the mechanical structure must be measured or observed and the EE must be moved accordingly to actively reduce vibrations. In a feedback vibration control scheme, special care must be taken to ensure stability of the closed-loop system, which can be challenging due to uncertainties in the system and spillover effects of unmodelled system dynamics. In contrast, feedforward vibration control methods inherently do not cause any stability issues for a stable system. For this reason, feedforward control is employed in the present work.

A popular and widely used approach for feedforward vibration control is so-called *input shaping* [50], where the actuator motion is shaped in such a way, that oscillations are avoided or compensated. There exist a variety of real-time capable input shaping techniques in the literature, see, e. g., [50, 51]. In this work, three different input shaping strategies are investigated and implemented. First, in Section 5.2, the possibility of using a notch-filter for removing resonant frequency components and thereby shaping the control signal is discussed. Second, the so-called *zero vibration* (ZV) input shaping method is investigated and compared to the notch-filter in Section 5.3. Third, in Section 5.4, the possibility of using more robust input shaping strategies is briefly outlined.

5.2 Notch filter

A so-called *notch filter* is a filter which is designed to remove a specific undesirable frequency from a signal. By applying a notch filter to the trajectory $(\mathbf{r}, \mathbf{v}, \mathbf{a})$, the resonant frequency component of the acceleration can be attenuated. Thus, a notch filter can be used to shape the trajectory of the EE and reduce structural vibrations.

A digital second order IIR notch filter with a single notch, located at the normalized angular frequency Ω_n, can be synthesized using the discrete-time transfer function [52]

$$H_n(z) = \beta \frac{1 - 2\cos\Omega_n z^{-1} + z^{-2}}{1 - 2\beta\cos\Omega_n z^{-1} + (2\beta - 1)z^{-2}} \,. \tag{5.2}$$

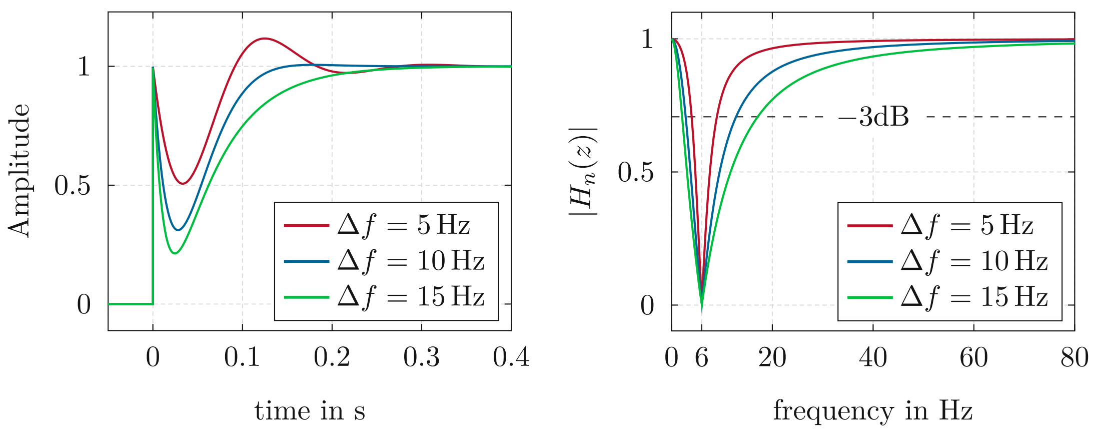
Here, the coefficient β influences the bandwidth i. e. sharpness of the resulting notch. The normalized angular frequency Ω_n of the notch is obtained by normalizing the angular frequency $\omega_n = 2\pi f_n$ using the sampling frequency f_s of the system in the form

$$\Omega_n = \frac{\omega_n}{f_s} = \frac{2\pi f_n}{f_s} \,. \tag{5.3}$$

The relation of the -3dB bandwidth Δf and the coefficient β is given by

$$\beta = \tan\left(\pi \frac{\Delta f}{f_s}\right) \,. \tag{5.4}$$

Using relations from Eq. (5.2)-(5.4), a notch filter with a notch frequency of $f_n = 6\,\mathrm{Hz}$ and any desired bandwidth Δf can be designed. These formulas are implemented in Matlab in the form of the `iirnotch` command. Figure 5.2 visualizes the impact of the filter bandwidth on the behavior of the notch filter. The step response for different bandwidth values Δf is shown in Figure 5.2(a). In comparison, the dependence of the frequency response on the bandwidth is depicted in Figure 5.2(b). Due to the reciprocal relation of the time domain and the frequency domain, a smaller filter bandwidth in the frequency domain corresponds to a longer duration of the step response in the time domain. The step response of the notch filter can be interpreted as a step, which is superimposed with an additional compensation of the notch frequency. The smaller the bandwidth of the compensated notch frequency, the higher the damping of the compensation signal on top of the step signal.



(a) Step response. (b) Frequency response.

Figure 5.2: Notch filter with $f_n = 6\,\text{Hz}$ and different bandwidth values Δf.

A notch bandwidth of $\Delta f = 10\,\text{Hz}$ is chosen to obtain a suitable settling time of the filter. Thus, a reasonably fast response of the filter can be achieved without excessive ringing.

5.3 Zero vibration input shaping

The basic idea behind the zero vibration input shaping method is to split the desired input signal into two steps, where the second step is delayed by half the period of the resonant frequency of the system. Thus, any vibration induced by the first step is cancelled by the second step of the signal [53]. The method is based on the vibration model of a damped linear oscillator, with the impulse response $y(t)$ given by

$$y(t) = A\frac{\omega_0}{\sqrt{1-\zeta^2}}e^{-\zeta\omega_0 t}\sin\left(t\omega_0\sqrt{1-\zeta^2}\right). \tag{5.5}$$

Here, A is the amplitude of the impulse excitation, $\omega_0 = 2\pi f_0$ denotes the undamped angular frequency and ζ the damping ratio of the oscillator. For the present application, the input corresponds to the reaction force caused by the acceleration of the EE and the output corresponds to the displacement of the structural frame of the prototype robot.

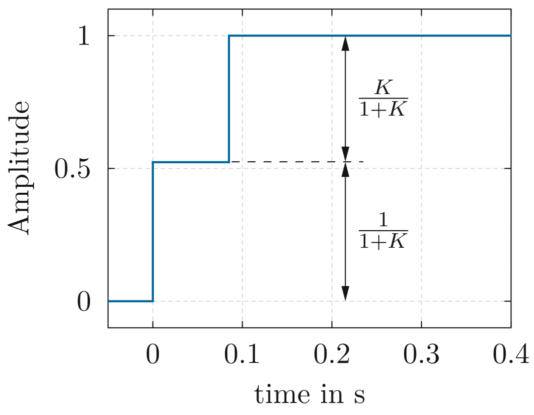
Based on the vibration model from Eq. (5.5), a discrete time ZV input shaping filter can be realized in the form of a linear FIR filter with the transfer function

$$H_{\mathrm{ZV}}(z) = \frac{1}{1+K} + \frac{K}{1+K} z^{-n_\mathrm{d}} \,, \tag{5.6}$$

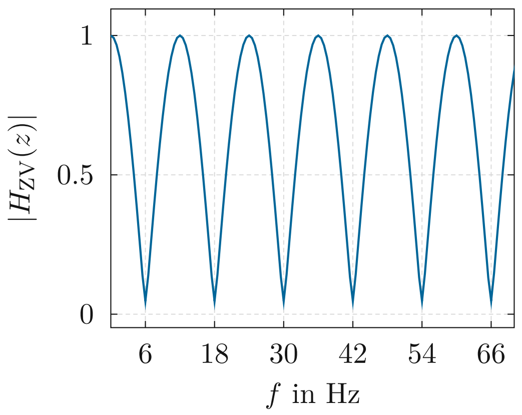
$$n_\mathrm{d} = \left\lfloor \frac{2 f_s}{f_0 \sqrt{1-\zeta^2}} \right\rceil , \qquad K = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} \,, \tag{5.7}$$

where $\lfloor\cdot\rceil$ denotes the rounding operator which rounds to the nearest integer. A detailed derivation of the method and a discussion of its advantages, shortcomings and possible extensions can be found in [53].

Using Eqs. (5.6) and (5.7), a ZV input shaping filter can be designed for the dominating structural resonance discussed in Section 7.3. Here, the values $f_0 = 6\,\mathrm{Hz}$ and $\zeta = 0.03$ are applied for the prototype robot and a sampling frequency of $f_s = 8\,\mathrm{kHz}$ is used. Figure 5.3 shows behavior of the resulting ZV input shaping filter.



(a) Step response.



(b) Frequency response.

Figure 5.3: ZV input shaping filter behavior.

The step response depicted in Figure 5.3(a) shows the two separate delayed steps conducted by the input shaping algorithm. With the parameters used for the prototype robot, the delay has a duration of 83 ms.

The frequency response of the input shaper shows, that the frequency content of the resonant frequency (6 Hz) and all uneven multiples is removed. Any even multiple of the resonant frequency is unaffected by the input shaping filter. Note that $|H(z)|$ is slightly larger than zero at the resonance frequency because only a damped oscillation with the exact damping ζ is exactly cancelled by the filter.

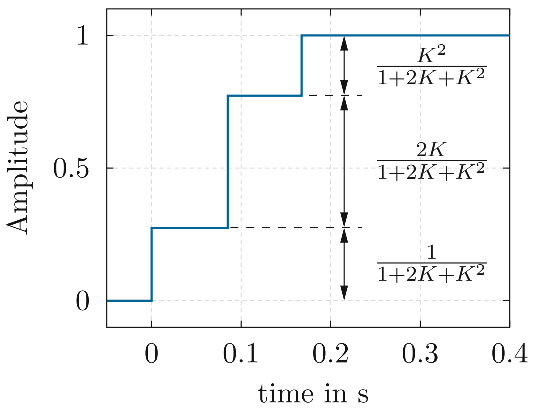
5.4 Robust input shaping

Both the notch filter presented in Section 5.2 and the ZV input shaping discussed in Section 5.3 rely on a precise model of the vibrations, in particular of the frequency of the oscillation. In the literature, there exist a variety of alternative input shaping methods which offer greater robustness to parameter uncertainties. Singer and Seering [53] present an extension of the ZV method by setting the derivative of the frequency response to zero at the resonance frequency. The resulting input shaper is referred to as *zero vibration derivative* (ZVD) method and requires separating the input signal into three steps. This concept can be extended to methods with more steps, to gain even more robustness to uncertain or changing parameters.

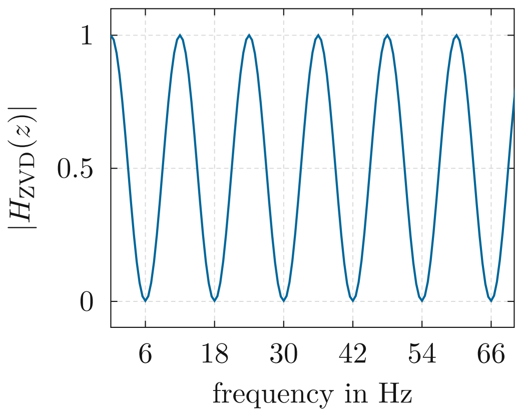
Based on the vibration model from Eq. (5.5), a ZVD input shaper can be implemented as a digital FIR filter in the form

$$H_{\mathrm{ZVD}}(z) = \frac{1}{1+2K+K^2} + \frac{2K}{1+2K+K^2} z^{-n_{\mathrm{d}}} + \frac{K^2}{1+2K+K^2} z^{-2n_{\mathrm{d}}}\,, \tag{5.8}$$

with K and n_{d} from Eq. (5.7). The resulting step response and frequency response for the ZVD input shaper using $f_0 = 6\,\mathrm{Hz}$ and $\zeta = 0.03$ is shown in Figure 5.4. Here, the step response shown in Figure 5.4(a) shows that the step is separated into three separate steps which require a total duration of 167 ms to complete the shaped signal. The frequency response depicted in Figure 5.4(b) shows the cancellation of the resonant frequency (6 Hz) and all uneven multiples, where the derivative of $|H_{\mathrm{ZVD}}(z)|$ with respect to the frequency is also zero.



(a) Step response.



(b) Frequency response.

Figure 5.4: ZVD input shaping filter behavior.

There also exist similar methods with three and more steps, which offer higher robustness to parameter uncertainties by not aiming for full cancellation of the vibration but a desired tolerable attenuation of vibrations across a wider frequency range [54]. A comparison of the performance of different robust input shaping methods can be found in [51]. With any input shaping method, there is an inherent trade-off between the robustness with respect to parameter uncertainties and the duration of the shaped input signal. To achieve a higher robustness, a longer duration of the shaped signal is required. For the application of catching a dart, robust input shaping techniques demand a relatively long duration of the shaped signals and are therefore considered impractical.

6 Flight prediction

6.1 Overview

In this chapter, the flight prediction algorithm is outlined. This algorithm uses the measurement data provided by the optical tracking system to track the flight of the dart and predict the impact location. The goal of the algorithm is to compute a target set-point $\mathbf{x}_{\text{set}}$ for the robot pose, such that the dartboard can be positioned accordingly and the dart hits the desired segment on the dartboard.

The developed architecture of the flight prediction algorithm consists of four intermediate steps as shown in Figure 6.1. These steps are briefly outlined in the following and elaborated in more detail in the subsequent sections.

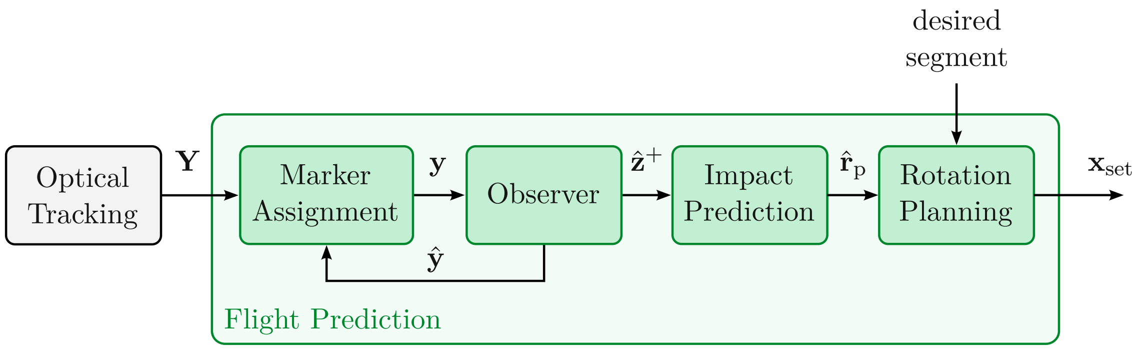


Figure 6.1: Internal steps of the flight prediction stage.

The Optical Tracking system is used to capture the position of three reflective markers, which are attached to the dart. The flight prediction algorithm receives the raw measurement data $\mathbf{Y}$ from the optical tracking system. This data contains the coordinates of all optical markers which were detected in the current frame. First, in the Marker Assignment step, the raw measurement data $\mathbf{Y}$ is processed and the optical marker coordinates $\mathbf{y}$, which correspond to the dart, are identified and extracted from the raw data. Second, the Observer uses the measured data $\mathbf{y}$ and a mathematical model of the dart dynamics to compute an estimate $\hat{\mathbf{z}}^+$ of the dart state. If available, a prediction of the expected marker positions $\hat{\mathbf{y}}$ is provided to the marker assignment algorithm to increase reliability in case marker signals are missing in the raw tracking data. In the third step, the model of the dart is used for the Impact Prediction. Fourth, the predicted impact position $\hat{\mathbf{r}}_\text{p}$ and the desired segment on the dartboard are used to compute a suitable combination of rotation and translation of the dartboard in the Rotation Planning step. Thus, a set-point for the pose $\mathbf{x}_{\text{set}}$ is obtained.

6.2 Mathematical model

In the following, a mathematical model of the dart is presented. The hardware components employed in the prototype robot are discussed in Section 7.1. The geometry of the dart and the individual components are depicted in Figure 6.2. Here, the locations of the three optical markers, which are used to track the dart, are illustrated in orange color.

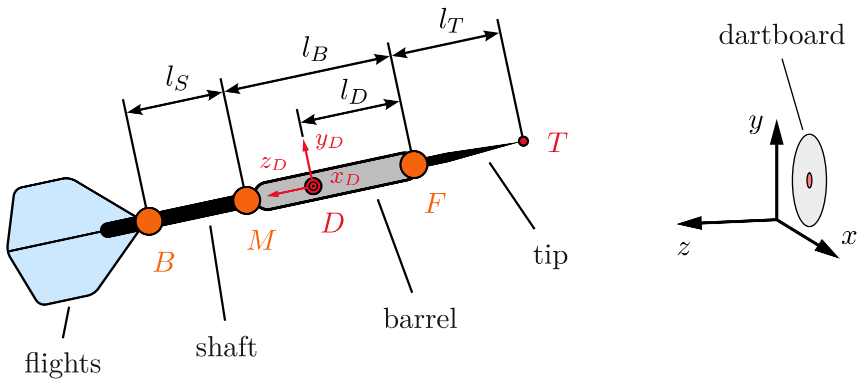


Figure 6.2: Dart geometry and optical marker location.

The center of mass of the dart is denoted D and lies inside the so-called barrel of the dart. The three markers are denoted F, M and B and the sharp point at the end of the so-called tip of the dart is denoted T. In the rear part of the dart, the so-called flights are installed, which stabilize the orientation of the dart in the air.

6.2.1 Kinematics

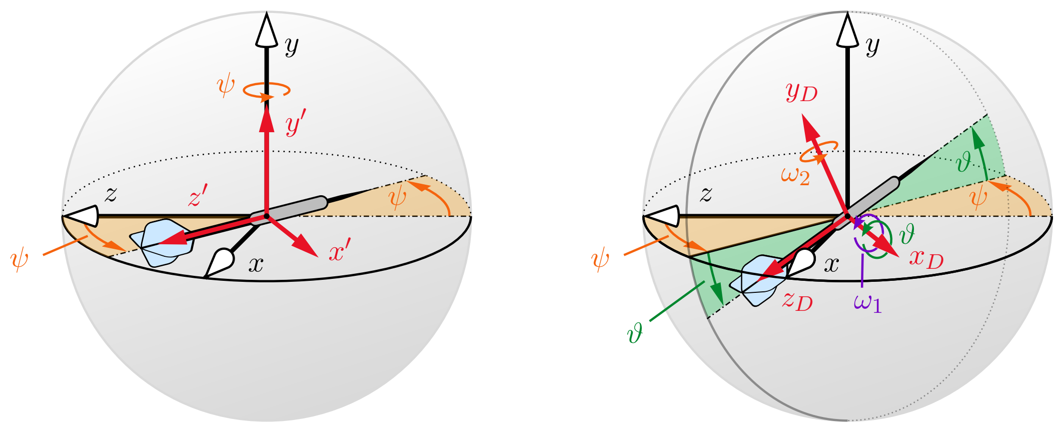
To model the motion and flight trajectory of the dart, the dart is considered a rigid body and any deformations are neglected. Furthermore, the spin of the dart around the z_D-axis and the influence of the angular spin velocity on the flight trajectory is assumed to be insignificant. Thus, the rotation of the dart around the z_D-axis is neglected and the model for the dart has 5 degrees-of-freedom (DOFs), i. e. 3 DOFs of translation and 2 DOFs of rotation. Note, that for the chosen marker placement shown in Figure 6.2, the spin of the dart around the z_D-axis does not affect the marker positions and is therefore not observable.

To describe the 3 translational DOFs, the position vector $\mathbf{r}_D$, which points from the origin to the center of gravity D, is used. The position vector $\mathbf{r}_D$ reads as

$$\mathbf{r}_D = \begin{bmatrix} x_D & y_D & z_D \end{bmatrix}^{\mathrm{T}} . \tag{6.1}$$

Here, the robot coordinate system is used as inertial reference frame which is located at the center of the CDPR as shown in Figure 2.1. Thus, the dartboard moves in the xy-plane as indicated in Figure 6.2.

To describe the 2 rotational DOFs, the orientation of the dart is parameterized using the two angles ϑ and ψ as visualized in Figure 6.3. The dart coordinate frame is defined by the basis vectors $\mathbf{e}_{x_D}$, $\mathbf{e}_{y_D}$ and $\mathbf{e}_{z_D}$ and centered at point D. To obtain the orientation of the dart coordinate frame, a rotation around the y-axis by the angle ψ is conducted to obtain the intermediate basis vectors $\mathbf{e}_{x'}$, $\mathbf{e}_{y'}$ and $\mathbf{e}_{z'}$ as shown in Figure 6.3(a). Thereafter, the intermediate coordinate system is rotated around the x'-axis by the angle ϑ to obtain the dart coordinate frame $\mathbf{e}_{x_D}$, $\mathbf{e}_{y_D}$ and $\mathbf{e}_{z_D}$ as depicted in Figure 6.3(b).



(a) Angle ψ. (b) Angle ϑ.

Figure 6.3: Parameterization of the dart orientation.

Thus, an arbitrary vector $(\mathbf{r})_D$ in the dart coordinate frame, given in the basis $(\mathbf{e}_{x_D}, \mathbf{e}_{y_D}, \mathbf{e}_{z_D})$ can be transformed to the basis $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ via the rotation matrix $\mathbf{R}^D$ in the form

$$\mathbf{r} = \mathbf{R}^D (\mathbf{r})_D \ . \tag{6.2}$$

The rotation matrix $\mathbf{R}^D$ reads as

$$\mathbf{R}^D = \mathbf{R}_{y,\psi}\mathbf{R}_{x',\vartheta} = \begin{bmatrix} \cos\psi & \sin\psi\sin\vartheta & \sin\psi\cos\vartheta \\ 0 & \cos\vartheta & -\sin\vartheta \\ -\sin\psi & \cos\psi\sin\vartheta & \cos\psi\cos\vartheta \end{bmatrix} , \tag{6.3}$$

using the elementary rotation matrices

$$\mathbf{R}_{x',\vartheta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\vartheta & -\sin\vartheta \\ 0 & \sin\vartheta & \cos\vartheta \end{bmatrix} , \qquad \mathbf{R}_{y,\psi} = \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix} . \tag{6.4}$$

In other words, the columns of the rotation matrix $\mathbf{R}^D$ correspond to the basis vectors of the dart coordinate frame

$$\mathbf{R}^D = \begin{bmatrix} \mathbf{e}_{x_D} & \mathbf{e}_{y_D} & \mathbf{e}_{z_D} \end{bmatrix} , \tag{6.5}$$

as observed from the inertial reference frame.

The relation of the angular velocities ω_1 and ω_2 in the dart coordinate frame (see Figure 6.3(b)) to the angular velocities $\dot{\vartheta}$ and $\dot{\psi}$ is given by

$$\begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} = (\mathbf{R}^D)^{\mathrm{T}} \dot{\mathbf{R}}^D \ . \tag{6.6}$$

A detailed derivation of Eq. (6.6) can be found in [20]. Calculating $\dot{\mathbf{R}}^D$ and inserting Eq. (6.3) into Eq. (6.6) yields the relations

$$\begin{aligned}\omega_1 &= \dot{\vartheta} \\ \omega_2 &= \dot{\psi}\cos\vartheta \ .\end{aligned} \tag{6.7}$$

To describe the motion of the dart and subsequently derive a dynamic model in Section 6.2.2, the kinematic state $\mathbf{z}$ of the dart is defined as

$$\mathbf{z} = \begin{bmatrix} x_D & y_D & z_D & \dot{x}_D & \dot{y}_D & \dot{z}_D & \vartheta & \psi & \omega_1 & \omega_2 \end{bmatrix}^{\mathrm{T}} . \tag{6.8}$$

Measuring the position of the three markers via the optical tracking system yields the output vector $\mathbf{y}$, which is defined as

$$\mathbf{y} = \begin{bmatrix} \mathbf{r}_F \\ \mathbf{r}_M \\ \mathbf{r}_B \end{bmatrix} = \begin{bmatrix} x_F & y_F & z_F & x_M & y_M & z_M & x_B & y_B & z_B \end{bmatrix}^{\mathrm{T}} . \tag{6.9}$$

Using the dart geometry depicted in Figure 6.2 and the rotation matrix $\mathbf{R}^D$ from Eq. (6.3), the positions of the optical markers, i. e. $\mathbf{r}_F$, $\mathbf{r}_M$ and $\mathbf{r}_B$ can be calculated. Thus, the output $\mathbf{y}$ from Eq. (6.9) can be written as a function $\mathbf{h}(\mathbf{z})$ in the form

$$\mathbf{y} = \mathbf{h}(\mathbf{z}) = \begin{bmatrix} \mathbf{r}_D - l_D \mathbf{e}_{z_D} \\ \mathbf{r}_D + (l_B - l_D)\mathbf{e}_{z_D} \\ \mathbf{r}_D + (l_B - l_D + l_S)\mathbf{e}_{z_D} \end{bmatrix} \qquad \mathbf{e}_{z_D} = \mathbf{R}^D(\vartheta,\psi)\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} . \tag{6.10}$$

6.2.2 Dynamics

To describe the flight dynamics of the dart, a model for the temporal evolution of the kinematic state $\mathbf{z}(t)$ is presented in this section. The flight of the dart consists of two stages as visualized in Figure 6.4. To account for the structural change between the two stages, the switching variable $u \in \{0, 1\}$ is utilized in the mathematical model.

In the handheld stage ① ($u = 0$), the human player holds the dart such that the hand moves around with the dart freely. In this stage, the hand compensates the gravitational acceleration and the dart is accelerated unpredictably as the human moves around and prepares for throwing the dart. In this work, the motion patterns followed by the human player are not further modelled and the best estimate for the acceleration of the dart is zero. Hence, the dynamical model for the handheld stage ① is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{z} = \mathbf{f}(\mathbf{z}, u = 0) = \begin{bmatrix} \dot{x}_D & \dot{y}_D & \dot{z}_D & 0 & 0 & 0 & \omega_1 & \frac{\omega_2}{\cos\vartheta} & 0 & 0 \end{bmatrix}^{\mathrm{T}} . \tag{6.11}$$

When the dart is thrown and detaches from the hand, the dart is in the airborne stage ② ($u = 1$). In this stage, the dart is subject to the gravitational acceleration and experiences aerodynamic forces which stabilize the rotation of the dart and change the flight path. In the following, the motion of an airborne dart is analyzed to derive a model for the airborne stage ②.

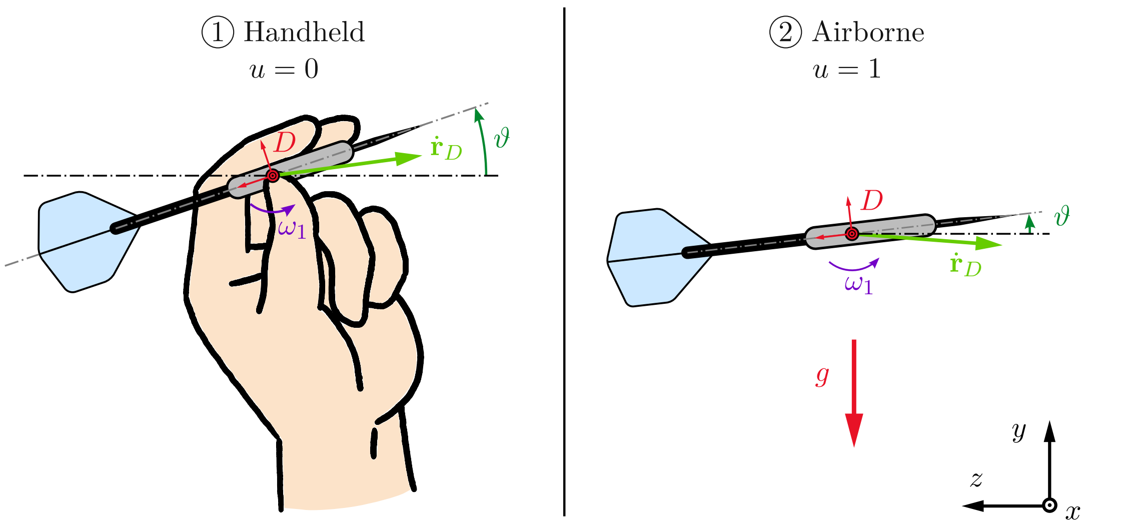


Figure 6.4: Switching between the handheld stage ① and the airborne stage ②.

The flight trajectory $\mathbf{r}_D(t)$ of a tournament dart closely resembles a parabolic ballistic trajectory, which is the result of the gravitational acceleration acting on the dart in negative y-direction. However, the aerodynamic forces acting on the dart change the shape of the trajectory and significantly impact the behavior of the orientation of the dart. The flights located at the rear of the dart cause the aerodynamic center of the dart to be located behind the center of mass. As a result, any deviation of the orientation of the dart from the flight direction causes an opposing torque. Thus, the flights stabilize the dart and attempt to align the z_D-axis with the velocity $\dot{\mathbf{r}}_D$. When the dart is thrown by the player with an initial angular velocity, this effect causes the orientation of the dart to oscillate around the direction of flight.

A typical trajectory of a dart thrown by an amateur player is shown in Figure 6.5. The image shown in Figure 6.5 was created by overlaying multiple frames captured with a high-speed camera. Here, the oscillation of the angle ϑ is visible. The dart is thrown such

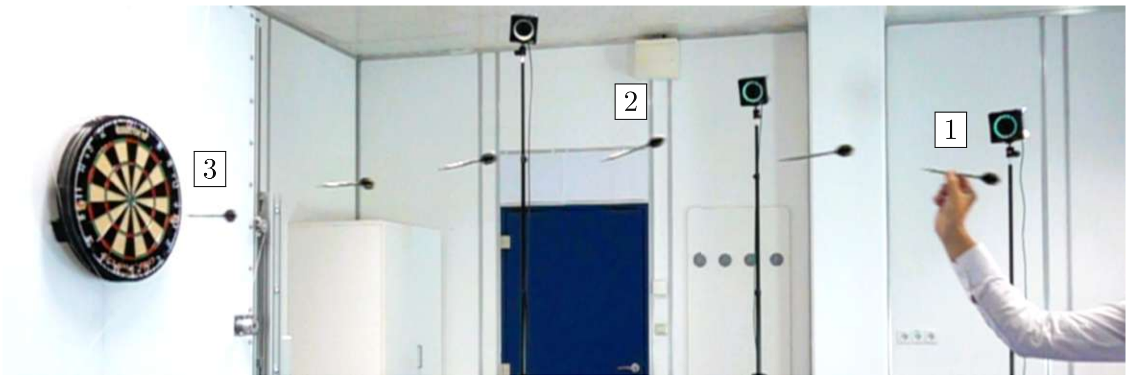
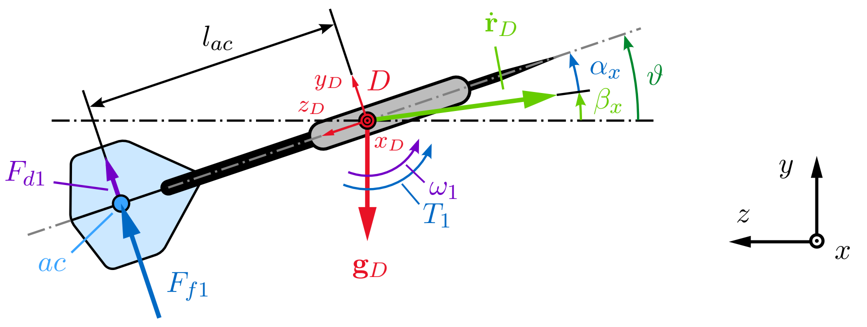


Figure 6.5: Typical trajectory of a dart thrown by an amateur player.

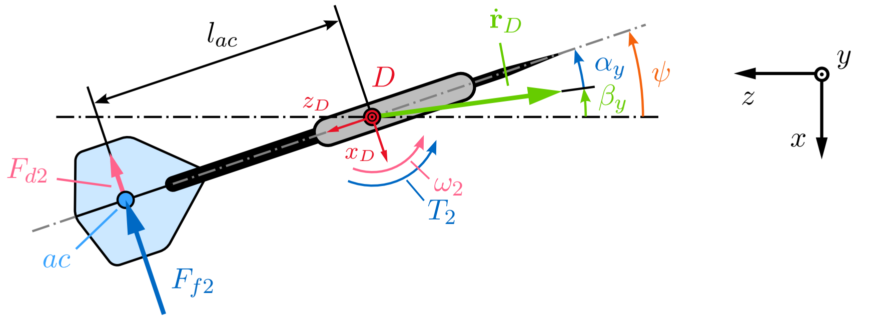
that initially $\vartheta > 0$ at point $\boxed{1}$. Due to the negative initial angular velocity, the dart pitches downward such that $\vartheta < 0$ at $\boxed{2}$. Finally, the aerodynamic forces cause the dart to pitch upward again such that $\vartheta \approx 0$ at point $\boxed{3}$.

To accurately predict the impact position of the tip of the dart and reliably hit small segments such as the bullseye, the dynamics of the orientation must be modelled and aerodynamic effects must be considered. The flowfield over a flying dart is a complex combination of different fundamental flow shapes. In [55], the airflow over a dart is studied in detail using smoke flow visualization in a wind channel. It is found, that the flow can be characterized by different counter rotating vortex systems which form due to the separation of air around the barrel, shaft and at the leading edge of the flights. The results indicate, that the flights behave similar to a delta-wing and efficiently create lift, even at large angles of attack.

For the aerodynamic model developed in the present thesis, it is assumed that the angles ϑ and ψ are small when the dart is airborne. Consequently, the coupling between the dynamics of ϑ and ψ is neglected and each rotational degree-of-freedom is treated independently. The modelled forces acting on the dart are visualized in Figure 6.6.



(a) Forces acting on the airborne dart in the yz-plane for $\psi \approx 0$.



(b) Forces acting on the airborne dart in the xz-plane for $\vartheta \approx 0$.

Figure 6.6: Forces acting on the dart for small angles ϑ and ψ.

The gravitational force $\mathbf{g}_D$ acts at the center of gravity D and reads as

$$\mathbf{g}_D = \begin{bmatrix} 0 & -m_D g & 0 \end{bmatrix}^{\mathrm{T}}, \tag{6.12}$$

with the gravitational acceleration g and the mass of the dart m_D.

Aerodynamic forces acting on the dart are created as a result of the interaction of the dart with the surrounding air as the dart moves through the air. Here, the motion of the rigid dart relative to the air is described by the velocity $\dot{\mathbf{r}}_D$ and the angular velocities ω_1 and ω_2. For better readability, the euclidean norm of the velocity $\|\dot{\mathbf{r}}\|_2$ is denoted v_D in the following. For simplicity, the effect of the linear motion of the whole dart with the velocity v_D is treated separately from the effect of the angular velocities ω_1 and ω_2, respectively.

It is assumed, that all aerodynamic forces act at the aerodynamic center denoted ac as shown in Figure 6.6, where the distance between the center of mass D and the aerodynamic center ac is denoted l_{ac}.

The lift and drag forces generated by the motion with a relative velocity of v_D are modelled using the forces F_{f1} acting in $\mathbf{e}_{y_D}$ direction and F_{f2} acting in negative $\mathbf{e}_{x_D}$ direction, respectively. These forces are caused by the flights of the dart, which act as wings and generate lift. Thus, the forces $F_{f1}\mathbf{e}_{y_D}$ and $-F_{f2}\mathbf{e}_{x_D}$ depend on the angle of attack α_x shown in Figure 6.6(a) and α_y shown in Figure 6.6(b), respectively. In the proposed model, the forces F_{f1} and F_{f2} are given by

$$F_{f1} = C_{0f} v_D^2 \sin\alpha_x \ , \qquad F_{f2} = C_{0f} v_D^2 \sin\alpha_y \ , \tag{6.13}$$

with the constant parameter C_{0f}. The angles of attack α_x and α_y are defined as

$$\alpha_x = \vartheta - \operatorname{atan2}(\dot{y}_D, -\dot{z}_D) \ , \qquad \alpha_y = \psi - \operatorname{atan2}(-\dot{x}_D, -\dot{z}_D) \ . \tag{6.14}$$

In addition, the angular velocities ω_1 and ω_2 cause additional aerodynamic forces, which act as damping and oppose the angular velocities. These damping effects are modelled using the force $F_{d1}\mathbf{e}_{y_D}$ and $-F_{d2}\mathbf{e}_{x_D}$, respectively. The proposed damping forces are given by the relations

$$F_{d1} = C_{0d} v_D \omega_1 \ , \qquad F_{d2} = C_{0d} v_D \omega_2 \ , \tag{6.15}$$

with the constant parameter C_{0d}.

The balance of momentum can be written by transforming the forces from Eqs. (6.13) and (6.15) from the dart coordinate system to the reference coordinate system via the rotation matrix $\mathbf{R}^D$ from Eq. (6.3). Thus, the relation

$$\ddot{\mathbf{r}}_D = \mathbf{g}_D + \frac{1}{m_D}\mathbf{R}^D \begin{bmatrix} -F_{f2} - F_{d2} \\ F_{f1} + F_{d1} \\ 0 \end{bmatrix} \tag{6.16}$$

is obtained.

Due to the offset l_{ac}, the aerodynamic forces cause the torques T_1 around the $\mathbf{e}_{x_D}$-axis and T_2 around the $\mathbf{e}_{y_D}$-axis, respectively, as shown in Figure 6.6. As discussed earlier, these torques stabilize the rotation of the dart. The balance of angular momentum in the dart coordinate frame reads as

$$\begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ 0 \end{bmatrix} = -\mathbf{I}_D^{-1} l_{ac} \begin{bmatrix} F_{f1} + F_{d1} \\ F_{f2} + F_{d2} \\ 0 \end{bmatrix} \ , \tag{6.17}$$

where $\mathbf{I}_D$ denotes the inertia tensor in the dart coordinate frame. The z_D-axis of the dart coordinate system is aligned with the symmetry axis of the dart. Due to the symmetry of the dart, the axes of the dart coordinate system form principal axes for the inertia tensor and the tensor can be written in the form

$$\mathbf{I}_D = \begin{bmatrix} I_{D\perp} & 0 & 0 \\ 0 & I_{D\perp} & \\ 0 & 0 & I_{D\|} \end{bmatrix} , \tag{6.18}$$

where $I_{D\|}$ denotes the mass moment of inertia around the z_D-axis and $I_{D\perp}$ denotes the mass moment of inertia around the x_D and y_D axes, respectively. Note that due to symmetry the latter two are identical. Substituting Eq. (6.18) into Eq. (6.17) yields

$$\dot{\omega}_1 = -\frac{l_{ac}}{I_{D\perp}}(F_{f1} + F_{d1}) , \qquad \dot{\omega}_2 = -\frac{l_{ac}}{I_{D\perp}}(F_{f2} + F_{d2}) . \tag{6.19}$$

The unknown constant parameters C_{0f}, C_{0d}, m_D, $I_{D\perp}$ and l_{ac} can be lumped into new parameters in the form

$$C_f = \frac{C_{0f}}{m_D} , \qquad C_d = \frac{C_{0d}}{m_D} , \qquad C_l = \frac{l_{ac} m_D}{I_{D\perp}} , \tag{6.20}$$

which can be identified and calibrated using measurement data. The model parameter values used in this work are summarized in Table A.1 in Appendix A.

Combining the equations of motions from Eqs. (6.16) and (6.19) and using the constants from Eq. (6.20), the dart dynamics for the airborne stage $\textcircled{2}$ can be written in the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{z} = \mathbf{f}(\mathbf{z}, u = 1) = \begin{bmatrix} \dot{x}_D \\ \dot{y}_D \\ \dot{z}_D \\ \sin\psi \sin\vartheta A_1 - \cos\psi A_2 \\ -g + \cos\vartheta A_1 \\ \cos\psi \sin\vartheta A_1 + \sin\psi A_2 \\ \omega_1 \\ \frac{1}{\cos\vartheta}\omega_2 \\ -C_l A_1 \\ -C_l A_2 \end{bmatrix} , \qquad \begin{aligned} A_1 &= (F_{f1} + F_{d1}) , \\ A_2 &= (F_{f2} + F_{d2}) . \end{aligned} \tag{6.21}$$

It can be shown that the behavior of the dynamic model for the airborne dart given by Eq. (6.21) with respect to the dart oscillation is consistent with the small perturbation model presented in [56]. In this work, the oscillations of the pitch angle ϑ and the angle of attack α_x are studied using high-speed video footage from 225 dart throws from 19 different amateur players.

6.3 Marker Assignment

The marker assignment stage preprocesses the raw data frames $\mathbf{Y}$, which are received from the optical tracking system and extracts the relevant marker signals $\mathbf{y}$. The optical tracking system uses multiple cameras to capture the position of reflective markers. For this purpose, the camera images are processed by a proprietary tracking software running in the Windows operating system to triangulate the position of all markers found in the camera images. The time when a set of camera images is captured is denoted t_j and the corresponding data frame is denoted $\mathbf{Y}[j] = \mathbf{Y}(t_j)$. In the following, all signals which relate to measurements are represented as sequences using the measurement index $j = 0, 1, 2, \ldots$ corresponding to the measurement time t_j. As soon as a set of camera images has been processed by the tracking software, it is passed from the Windows environment to the robot controller. Due to latencies in the tracking software, the Windows operating system and the transmission system, the data $\mathbf{Y}[j]$ are received at irregular control timesteps k. Thus, there exists no simple relationship between the measurement steps denoted j and the controller steps k.

Each raw data frame $\mathbf{Y}[j]$ has a fixed length and contains data corresponding to the the first n_{mk} markers, which were found in the tracked volume. For each marker, the position $\mathbf{r}_{\text{mk},i}$ with $i = 1, \ldots, n_{\text{mk}}$ is contained in the data frame. In addition, a timestamp of the captured time t_j is available for each raw data frame $\mathbf{Y}[j]$. Furthermore, additional measurement related data for each marker is contained in the raw data such as the marker size and the triangulation residual. However, the present marker assignment algorithm does not use this additional data.

The goal of the marker assignment algorithm is to find the indices (i_F, i_M, i_B), which correspond to the markers attached to the dart as shown in Figure 6.2. Thus, the measurement $\mathbf{y}$ is obtained in the form

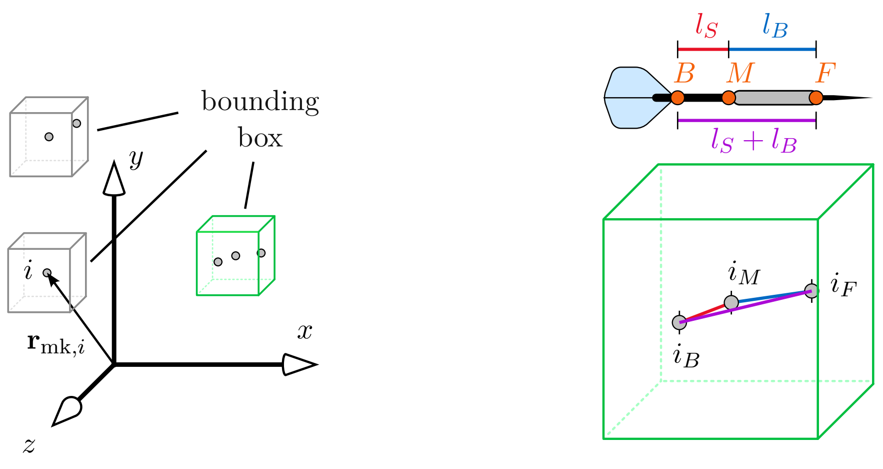
$$\mathbf{y} = \begin{bmatrix} \mathbf{r}_{\text{mk},i_F} & \mathbf{r}_{\text{mk},i_M} & \mathbf{r}_{\text{mk},i_B} \end{bmatrix}^{\mathrm{T}} . \tag{6.22}$$

The measurement data is subject to measurement errors and occasionally marker signals can be missing due to bad visibility of the optical markers. Hence, the marker assignment algorithm must be capable of assigning 1, 2 or all 3 marker positions to the corresponding measurement signals. In the present work, a heuristic marker assignment algorithm was developed, which is briefly outlined in the following.

Two different cases are distinguished by the marker assignment algorithm. In the first case, no previous knowledge about the position of the dart is available and the dart is not yet tracked by the observer presented in the following Section 6.4. Hence, the algorithm searches for 3 markers which match the dart geometry and assigns the marker signals accordingly. Figure 6.7 visualizes the assignment procedure without any state estimate. The idea behind the algorithm is to use bounding boxes centered around each marker to select candidates for a dart and remove outliers as illustrated in Figure 6.7(a). If a bounding box with exactly 3 markers inside the box is found, these markers are selected as candidates for a dart. The 3 distances between each pair of marker signals are compared to the expected distances l_S, l_B and $l_S + l_B$. If the measured distances match the dart definition within a predefined tolerance, the three markers are accepted as a valid dart and the indices i_F, i_M and i_B are assigned accordingly. This step is illustrated in Figure 6.7(b).

Note, that if no previous estimate for the dart state $\hat{\mathbf{z}}$ is available, the marker assignment only accepts a matching group of 3 markers as a valid measurement.

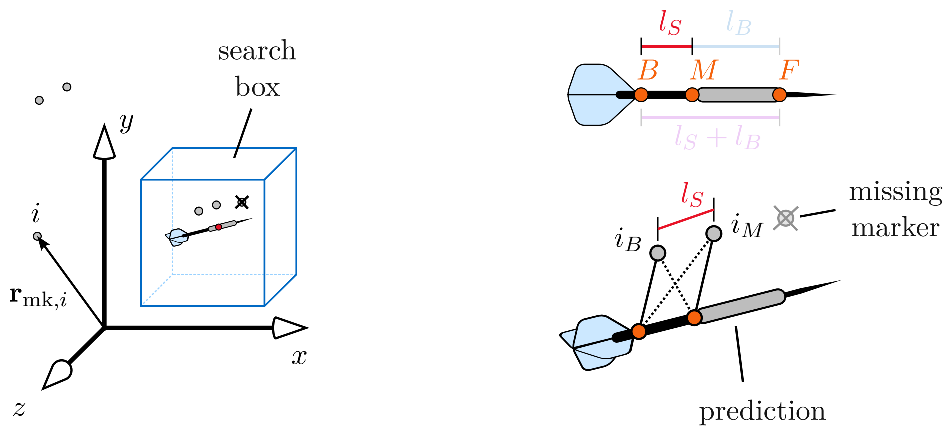
In the second case, the dart has been found in the previous frame $j - 1$ and a prediction



(a) Bounding boxes around markers. (b) Comparing marker distances to the model.

Figure 6.7: Marker assignment procedure without previous knowledge.

for the expected marker locations $\hat{\mathbf{y}}[j]$ is available. The marker assignment procedure in this case is visualized in Figure 6.8.



(a) Search box. (b) Finding best fit to the prediction.

Figure 6.8: Marker assignment procedure with a state estimate $\hat{\mathbf{z}}$.

In the first step, a search box centered around the predicted center of gravity $\hat{\mathbf{r}}_D[j]$ is used to find marker candidates and remove outliers. Here, the search box is chosen considerably larger than the bounding boxes in the previous case, to account for prediction errors as illustrated in Figure 6.8(a). If 1, 2 or 3 markers are found within the search box, they are used as candidates for the marker signals. In the example shown in Figure 6.8, only two valid markers were found and one marker signal is missing in the raw data $\mathbf{Y}[j]$. In the next step, the available marker candidates are compared to the prediction $\hat{\mathbf{y}}[j]$ to find the best fit as illustrated in Figure 6.8(b). If the distance between the markers matches the dart model within a predefined tolerance and the prediction error is below a predefined threshold, the markers are accepted as valid signals and the indices are assigned according to the best fit. In the example shown in Figure 6.8(b), the markers are identified as M and B because their distance matches the expected value l_S. The two possible assignment combinations are compared to the prediction and the best fit with the lower prediction error is chosen.

Note that the outlined procedure accepts marker signals with only 1 or 2 markers if their distance matches the dart definition and they are sufficiently close to the predicted positions. Thus, lost data points can be handled and the algorithm is robust with respect to missing data. However, if the marker signals are lost for an extended period of time and the prediction error becomes too large, the dart can no longer be tracked and a reliable estimation of the dart impact location is no longer possible.

If no valid match for the dart signals can be found for a predefined time T_{lost}, the state prediction $\hat{\mathbf{z}}$ is discarded because it is considered invalid.

6.4 Observer Design

To track the motion of the dart, a state observer is used. The observer uses the model derived in Section 6.2 and the measurement data $\mathbf{y}$, provided by the marker assignment algorithm, to estimate the kinematic state $\mathbf{z}$ of the dart. In the following, a discrete-time extended Kalman filter (EKF) is designed for this purpose. A classical Kalman filter is based on a linear system model. The idea behind an EKF is to extend the classical Kalman filter to nonlinear system models by linearizing the system at each step j.

The EKF is based on the nonlinear, discrete-time system model

$$\begin{aligned} \mathbf{z}[j+1] &= \mathbf{F}(\mathbf{z}[j], u[j], \mathbf{w}[j], \Delta t_j) \\ \mathbf{y}[j] &= \mathbf{h}(\mathbf{z}[j]) + \mathbf{v}[j] \ , \end{aligned} \tag{6.23}$$

where $\mathbf{w} \in \mathbb{R}^{10}$ denotes the unknown disturbance vector and $\mathbf{v} \in \mathbb{R}^{9}$ denotes the measurement noise. The output function $\mathbf{h}(\mathbf{z})$ is given by Eq. (6.10). To obtain the transition function $\mathbf{F}(\mathbf{z}, u, \mathbf{w}, \Delta t)$, the nonlinear continuous-time model given by Eqs. (6.11) and (6.21) is discretized using the explicit Euler method. Thus, the transition function is given by

$$\mathbf{F}(\mathbf{z}[j], u[j], \mathbf{w}[j], \Delta t_j) = \mathbf{z}[j] + \mathbf{f}(\mathbf{z}[j], u[j])\Delta t_j + \mathbf{w}[j] \ . \tag{6.24}$$

The unknown disturbance $\mathbf{w}[j]$ can be interpreted as the effect of modelling errors, unmodelled phenomena and parameter uncertainties in the model. It is assumed in the discrete model from Eq. (6.24), that the disturbance $\mathbf{w}[j]$ additively acts on the state $\mathbf{z}[j]$.

For the design of the EKF, it is assumed that the disturbance $\mathbf{w}$ and the measurement noise $\mathbf{v}$ are uncorrelated white noise sequences with zero mean. Thus, the relations

$$\mathrm{E}(\mathbf{w}[j]) = \mathbf{0} \qquad \mathrm{E}(\mathbf{w}[j]\mathbf{w}^{\mathrm{T}}[k]) = \mathbf{Q}[j]\delta_{j,k} \tag{6.25}$$

$$\mathrm{E}(\mathbf{v}[j]) = \mathbf{0} \qquad \mathrm{E}(\mathbf{v}[j]\mathbf{v}^{\mathrm{T}}[k]) = \mathbf{R}[j]\delta_{j,k} \tag{6.26}$$

$$\mathrm{E}(\mathbf{w}[j]\mathbf{v}^{\mathrm{T}}[k]) = \mathbf{0} \tag{6.27}$$

hold, where $\mathrm{E}(\cdot)$ denotes the expected value and $\delta_{j,k}$ denotes the Kronecker symbol. Here, $\mathbf{R} \in \mathbb{R}^{9\times 9}$ is the covariance matrix of the measurement noise and $\mathbf{Q} \in \mathbb{R}^{10\times 10}$ is the covariance matrix of the disturbance. It is assumed that the measurement uncertainty for each measured marker position $\mathbf{r}_{\mathrm{mk},i}$ is proportional to the triangulation residual ρ_i. If a marker signal is missing because the marker was not found by the optical tracking system, then the corresponding uncertainty is set to the large value $\rho_i = 10^9\mathrm{m}$. Hence, the noise covariance matrix is given in the form

$$\mathbf{R}[j] = R_{\mathrm{mk}}\mathrm{diag}(\rho_F[j], \rho_F[j], \rho_F[j], \rho_M[j], \rho_M[j]\rho_M[j], \rho_B[j], \rho_B[j], \rho_B[j]) \ , \tag{6.28}$$

with the constant parameter R_{mk}.

Furthermore, it is assumed that the disturbance depends on the flight stage, which is defined by the switching variable $u[j]$. In the handheld stage ① ($u[j] = 0$), the unmodelled forces exerted by the human can be very large and thus the disturbances are assumed to be large. In the airborne stage ② ($u[j] = 1$), the disturbances are caused by modelling errors and parameter uncertainties which are assumed to be smaller in comparison. Hence, the disturbance covariance matrix $\mathbf{Q}[j]$ is set according to the switching variable $u[j]$ in each timestep in the form

$$\mathbf{Q}[j] = \begin{cases} \mathbf{Q}_1 \ , & \text{for } u[j] = 0 \\ \mathbf{Q}_2 \ , & \text{for } u[j] = 1 \ . \end{cases} \tag{6.29}$$

Here, the two matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are constant design parameters. The parameter values chosen for the observer in this work are summarized in Table A.2 in Appendix A.

Because no a priori knowledge about the initial state of the dart is available, the EKF is initialized using

$$\hat{\mathbf{z}}[0] = \hat{\mathbf{z}}^+[0] = \mathbf{0} \qquad\qquad \mathbf{P}[0] = \mathbf{P}^+[0] = 10^6\mathbf{I} \ . \tag{6.30}$$

The estimation error covariance matrix $\mathbf{P} \in \mathbb{R}^{10\times10}$ is initialized as a diagonal matrix with very large values because the uncertainty of the initial guess $\hat{\mathbf{z}}[0]$ is very large. The update procedure for a discrete-time EKF is given by the following steps [57].

Ⓐ Compute the transition matrix $\mathbf{\Phi}[j-1]$:

$$\mathbf{\Phi}[j-1] = \frac{\partial}{\partial \mathbf{z}}\mathbf{F}\Big(\hat{\mathbf{z}}^+[j-1], u[j-1], \mathbf{0}, t_j - t_{j-1}\Big) \ . \tag{6.31}$$

Ⓑ Extrapolate state estimate $\hat{\mathbf{z}}[j]$, estimation error covariance matrix $\mathbf{P}[j]$ and output estimate $\hat{\mathbf{y}}[j]$ for the current timestep j:

$$\hat{\mathbf{z}}[j] = \mathbf{F}\Big(\hat{\mathbf{z}}^+[j-1], u[j-1], \mathbf{0}, t_j - t_{j-1}\Big) \ , \tag{6.32}$$

$$\mathbf{P}[j] = \mathbf{\Phi}[j-1]\mathbf{P}^+[j-1]\mathbf{\Phi}^{\mathrm{T}}[j-1] + \mathbf{Q}[j] \ , \tag{6.33}$$

$$\hat{\mathbf{y}}[j] = \mathbf{h}(\hat{\mathbf{z}}[j]) \ . \tag{6.34}$$

Ⓒ Compute the output matrix $\mathbf{C}[j]$ for the predicted state:

$$\mathbf{C}[j] = \frac{\partial}{\partial \mathbf{z}}\mathbf{h}(\hat{\mathbf{z}}[j]) \ . \tag{6.35}$$

Ⓓ Perform the measurement update to obtain the corrected state estimate $\hat{\mathbf{z}}^{+}[j]$ and the corrected estimation error covariance matrix $\mathbf{P}^{+}[j]$:

$$\mathbf{K}[j] = \mathbf{P}[j]\mathbf{C}^{\mathrm{T}}[j]\Big(\mathbf{C}[j]\mathbf{P}[j]\mathbf{C}^{\mathrm{T}}[j] + \mathbf{R}[j]\Big)^{-1} , \tag{6.36}$$

$$\hat{\mathbf{z}}^{+}[j] = \hat{\mathbf{z}}[j] + \mathbf{K}[j]\Big(\mathbf{y}[j] - \hat{\mathbf{y}}[j]\Big) , \tag{6.37}$$

$$\mathbf{P}^{+}[j] = \Big(\mathbf{I} - \mathbf{K}[j]\mathbf{C}[j]\Big)\mathbf{P}[j] . \tag{6.38}$$

In steps Ⓐ and Ⓑ, new estimates for the state $\hat{\mathbf{z}}[j]$, error covariance matrix $\mathbf{P}[j]$ and output $\hat{\mathbf{y}}[j]$ are predicted via extrapolation using the linearized system dynamics. These estimates are referred to as *a priori* estimates because they are computed before acquiring the measurement $\mathbf{y}[j]$. In steps Ⓒ and Ⓓ, the measurement $\mathbf{y}[j]$ is used to obtain improved estimates $\hat{\mathbf{z}}^{+}[j]$ and $\mathbf{P}^{+}[j]$. These estimates are called *a posteriori* estimates because they are calculated after the measurement is obtained.

The marker assignment algorithm outlined in Section 6.3 is executed between step Ⓑ and step Ⓒ. Here, the marker assignment algorithm uses the a priori estimate $\hat{\mathbf{y}}[j]$ to robustly find $\mathbf{y}[j]$ in the case of missing marker signals.

A state-machine is used to coordinate the different procedures involved in tracking the dart. The state-machine detects a valid throw of the dart based on the history of the distance to the dartboard and the velocity towards the dartboard. Let j_{a} denote the timestep, where the state-machine detects that the dart is airborne. Consequently, the switching variable $u[j_{\mathrm{a}}] = 1$ is set, to switch the model dynamics accordingly for the current timestep j_{a} and all subsequent timesteps. During the throwing process, the human player rapidly accelerates the dart towards the dartboard. This leads to a relatively large prediction error immediately after the throw. To improve the prediction when switching from the handheld stage ① to the airborne stage ②, the prediction error covariance matrix $\mathbf{P}[j_{\mathrm{a}}]$ is set to a large value. As a result, the EKF discards the possibly erroneous estimate and puts more weight on the following measurement data. Thus, the model switchover procedure from the handheld stage ① to the airborne stage ② can be summarized as:

$$u[j] = \begin{cases} 0\,, & \text{for } j < j_{\mathrm{a}} \\ 1\,, & \text{for } j \geq j_{\mathrm{a}}\,, \end{cases} \qquad \mathbf{P}[j_{\mathrm{a}}] = 10^{8}\mathbf{I} . \tag{6.39}$$

6.5 Impact prediction

An impact of the dart on the dartboard occurs, when the tip of the dart T arrives at the surface of the dartboard. Because the dartboard moves in the xy-plane, the impact position $\mathbf{r}_{\mathrm{p}}$ on the impact surface always has the same z-coordinate denoted z_{p}, which is given by the distance of the dartboard's surface to the xy-plane.

The position $\mathbf{r}_T$ of the tip T depends on the dart state $\mathbf{z}$ defined in Eq. (6.8) and can be written in the form

$$\mathbf{r}_T(\mathbf{z}) = \mathbf{r}_D - l_T\mathbf{e}_{z_D} = \begin{bmatrix} x_D - l_T\sin\psi\cos\vartheta \\ y_D + l_T\sin\vartheta \\ z_D - l_T\cos\psi\cos\vartheta \end{bmatrix} . \tag{6.40}$$

The impact of the dart occurs when the z-coordinate of the tip position $\mathbf{r}_T$ reaches the value z_p. Using the distance from the impact surface

$$h_\mathrm{p}(\mathbf{z}) = z_D - l_T \cos\vartheta \cos\psi - z_\mathrm{p} \tag{6.41}$$

as a so-called event function, the dart impact can be formulated as an event location problem in the form

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{z} &= \mathbf{f}(\mathbf{z}, u=1) + \mathbf{w} \\ \mathbf{z}(0) &= \mathbf{z}_0 \\ h_\mathrm{p}(\mathbf{z}(t_\mathrm{p})) &= 0 \; . \end{aligned} \tag{6.42}$$

Here, the impact time t_p and the corresponding impact state $\mathbf{z}(t_\mathrm{p})$ describe the impact of the dart at the dartboard. The vector field $\mathbf{f}(\mathbf{z}, u=1)$ from Eq. (6.21) describes the dynamics of the airborne dart and $\mathbf{w}$ denotes the disturbance. Using the solution $\mathbf{z}(t_\mathrm{p})$ of Eq. (6.42), the impact location $\mathbf{r}_\mathrm{p}$ is obtained by substituting $\mathbf{z}(t_\mathrm{p})$ into Eq. (6.40)

$$\mathbf{r}_\mathrm{p} = \mathbf{r}_T(\mathbf{z}(t_\mathrm{p})) \; . \tag{6.43}$$

To find an approximation $\hat{\mathbf{r}}_\mathrm{p}$ of the exact impact location $\mathbf{r}_\mathrm{p}$, the event location problem from Eq. (6.42) is discretized using Heun's method. Heun's method is a two-stage Runge-Kutta scheme and corresponds to using the trapezoid rule for integrating an ordinary differential equation (ODE) [58]. Thus, Heun's method is exact for integrating ODEs which have a linear derivative, which corresponds to a quadratic solution. The dart trajectory closely resembles a parabola due to the dominating influence of the gravitational acceleration. For this reason, Heun's method is very computationally efficient for numerically integrating the dart model. The event function is used as an inequality constraint, to find the step n_p where the impact has occurred. Thus, the discrete approximation of the event location problem from Eq. (6.42) reads as

$$\hat{\mathbf{z}}_\mathrm{p}[0] = \hat{\mathbf{z}}^+[j] \; , \tag{6.44a}$$

$$\begin{aligned} \hat{\mathbf{z}}_\mathrm{p}[n+1] &= \hat{\mathbf{z}}_\mathrm{p}[n] + \frac{1}{2}(\Delta\mathbf{z}_1 + \Delta\mathbf{z}_2) \; , \\ \Delta\mathbf{z}_1 &= \mathbf{f}(\hat{\mathbf{z}}_\mathrm{p}[n], 1) T_\mathrm{p}[j] \; , \\ \Delta\mathbf{z}_2 &= \mathbf{f}(\hat{\mathbf{z}}_\mathrm{p}[n] + \Delta\mathbf{z}_1, 1) T_\mathrm{p}[j] \; , \end{aligned} \tag{6.44b}$$

$$h_\mathrm{p}(\hat{\mathbf{z}}_\mathrm{p}[n_\mathrm{p}]) \leq 0 \; . \tag{6.44c}$$

The discrete event location problem from Eq. (6.44) is solved to predict the impact location as follows. In each measurement step j the updated a posteriori state estimate $\hat{\mathbf{z}}^+[j]$ is used to initialize $\hat{\mathbf{z}}_\mathrm{p}[0]$ as stated by Eq. (6.44a). Then, the iteration law from Eq. (6.44b) is applied until the condition from Eq. (6.44c) is satisfied. Thus, the iteration step n_p where the impact occurred is found. An approximation for the impact position is computed by linearly interpolating between the step n_p and the previous step $n_\mathrm{p}-1$ in the form

$$\bar{\mathbf{z}}_\mathrm{p}[j] = \hat{\mathbf{z}}_\mathrm{p}[n_\mathrm{p}] - \frac{\hat{\mathbf{z}}_\mathrm{p}[n_\mathrm{p}] - \hat{\mathbf{z}}_\mathrm{p}[n_\mathrm{p}-1]}{h_\mathrm{p}(\hat{\mathbf{z}}_\mathrm{p}[n_\mathrm{p}]) - h_\mathrm{p}(\hat{\mathbf{z}}_\mathrm{p}[n_\mathrm{p}-1])} h_\mathrm{p}(\hat{\mathbf{z}}_\mathrm{p}[n_\mathrm{p}]) \tag{6.45}$$

Finally, the predicted impact state $\bar{\mathbf{z}}_\mathrm{p}$ can be substituted into Eq. (6.40) to obtain an estimate for the impact position in the form

$$\hat{\mathbf{r}}_\mathrm{p}[j] = \begin{bmatrix} \hat{x}_\mathrm{p}[j] \\ \hat{y}_\mathrm{p}[j] \end{bmatrix} = \mathbf{r}_T(\bar{\mathbf{z}}_\mathrm{p}[j]) \ . \tag{6.46}$$

To determine a suitable time step $T_\mathrm{p}[j]$ for the iterative procedure, a preliminary estimate for the total duration of the dart flight $\hat{T}_\mathrm{tot}$ is calculated. This estimate is obtained from the state estimate $\hat{\mathbf{z}}^+[j]$ by assuming that the velocity in z-direction $\hat{v}_z^+[j]$ remains constant. Thus the duration can be estimated in the form

$$\hat{T}_\mathrm{tot}[j] = \frac{h_\mathrm{p}(\hat{\mathbf{z}}^+[j])}{\hat{v}_z^+[j]} \ . \tag{6.47}$$

This total duration is divided by the desired number of steps N_p which is a fixed design parameter. Thus, the time step T_p is given by

$$T_\mathrm{p}[j] = \frac{\hat{T}_\mathrm{tot}}{N_\mathrm{p}} \ . \tag{6.48}$$

6.6 Rotation planning

The CDPR has 3 degrees-of-freedom for moving the dartboard attached to the EE in the task space. The dartboard can be positioned in the xy-plane and rotated around the z-axis. Hence, the desired set-point pose $\mathbf{x}_\mathrm{set}$ for hitting a desired segment of the dartboard is not unique.

In this section, a simple heuristic rotation planning method for choosing a suitable set-point $\mathbf{x}_\mathrm{set}$ is presented. The aim of the planning algorithm is to choose a visually appealing combination of rotation and translation with a low computational effort.

To define the chosen target segment, the target impact point is parameterized in polar coordinates using the radius r_t and the angle φ_t as illustrated in Figure 6.9. Here, the triple 5 segment is shown as an example. The values for r_t and φ_t are stored in a lookup table for each segment on the dartboard such that the user can choose any desired segment as a target. The algorithm calculates a suitable set-point pose $\mathbf{x}_\mathrm{set}$ in two steps as visualized in Figure 6.10.

In the first step, the angle φ_set is chosen in the form

$$\varphi_\mathrm{set}[j] = K_\varphi(\varphi_\mathrm{p}[j] - \varphi_\mathrm{t}) \ , \tag{6.49a}$$

$$\begin{aligned} \varphi_\mathrm{p}[j] &= \begin{cases} \operatorname{atan2}(-\hat{x}_\mathrm{p}[j], \hat{y}_\mathrm{p}[j]) \ , & \text{if} \quad \hat{x}_\mathrm{p}^2[j] + \hat{y}_\mathrm{p}^2[j] > r_\mathrm{min}^2 \\ \varphi_\mathrm{p}[j-1] \ , & \text{otherwise} \ . \end{cases} \\ \varphi_\mathrm{p}[0] &= 0 \ , \end{aligned} \tag{6.49b}$$

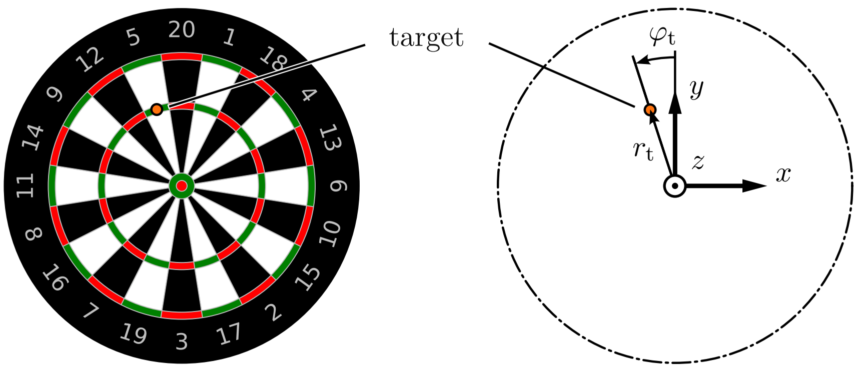
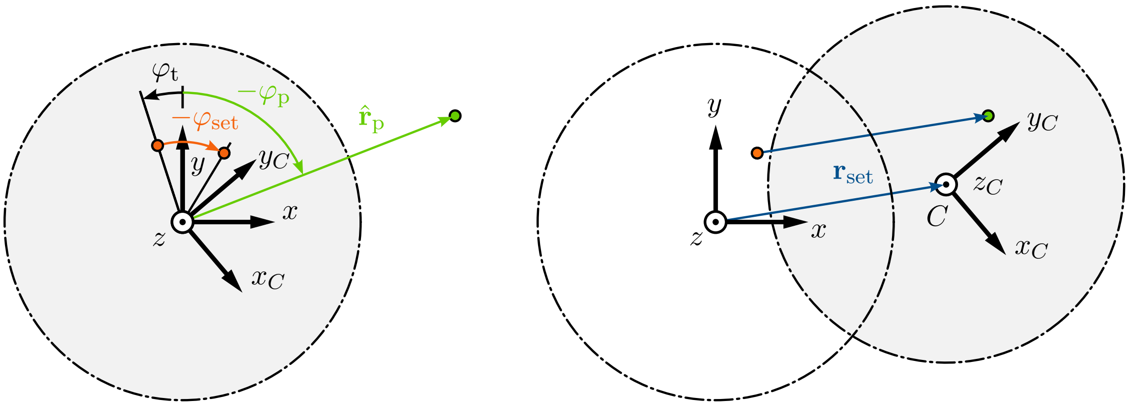


Figure 6.9: Parameterization of the desired segment (e. g. triple 5) on the dartboard.



(a) Rotation φ_set. (b) Translation $\mathbf{r}_\mathrm{set}$.

Figure 6.10: Rotation planning algorithm.

where $K_\varphi \in [0, 1]$ is a constant design parameter. This step is illustrated in Figure 6.10(a). The choice of the parameter K_φ determines the amount of rotation chosen by the algorithm. For the choice $K_\varphi = 0$, only pure translation is performed. In contrast, when choosing $K_\varphi = 1$ the algorithm will align the rotation of the target segment on the dartboard with the polar angle of the impact position. If the predicted impact position $\hat{\mathbf{r}}_\mathrm{p}$ is very close to the origin, then the angle φ_p can vary greatly due to measurement noise. Thus, the angle cannot be calculated reliably and the last accepted value is used instead. The constant parameter r_min is used as a threshold. In the special case of choosing the bullseye as desired segment, the angle φ_t is undefined because the bullseye is located in the center of the dartboard. In this special case, the set-point rotation angle is set to $\varphi_\mathrm{set}[j] = 0$.

In the second step of the rotation planning algorithm, the set-point $\mathbf{r}_\mathrm{set}[j]$ is chosen accordingly, such that the target impact position given by r_t and φ_t coincides with the predicted impact position $\hat{\mathbf{r}}_\mathrm{p}[j]$. The required remaining translational movement reads as

$$\mathbf{r}_\mathrm{set}[j] = \hat{\mathbf{r}}_\mathrm{p}[j] - r_\mathrm{t} \begin{bmatrix} -\sin\left(\varphi_\mathrm{set}[j] + \varphi_\mathrm{t}\right) \\ \cos\left(\varphi_\mathrm{set}[j] + \varphi_\mathrm{t}\right) \end{bmatrix}, \tag{6.50}$$

as visualized in Figure 6.10(b).

7 Experimental results

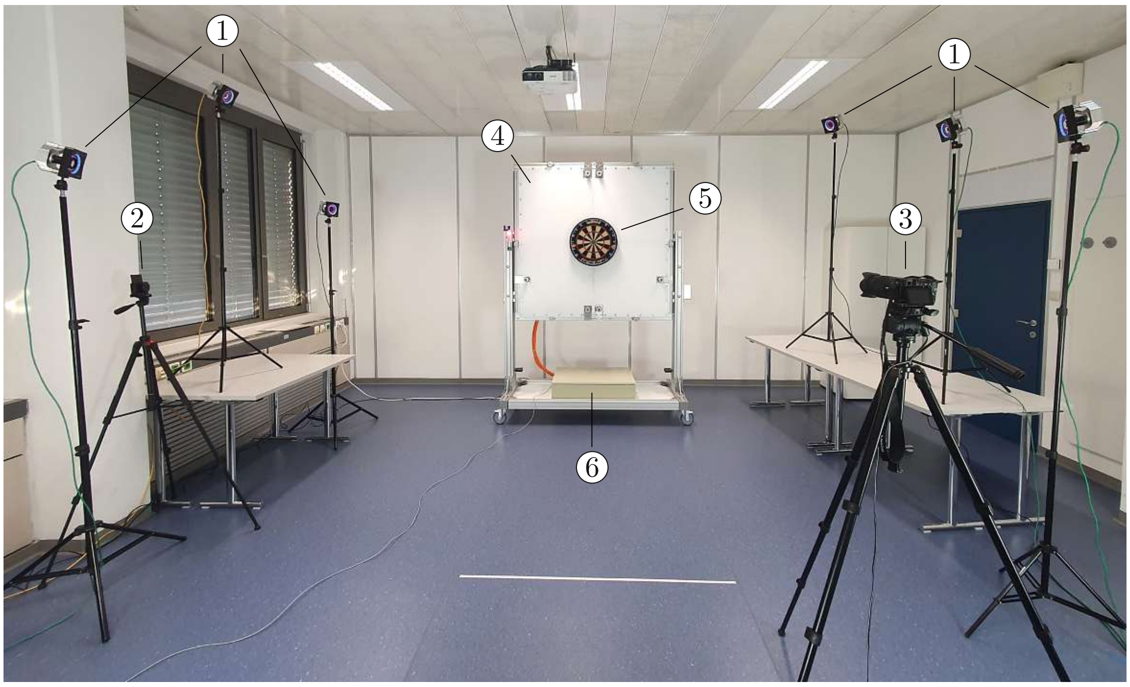
In this chapter, experimental results are presented which validate the theoretical considerations and demonstrate the effectiveness and performance of the developed system. First, in Section 7.1 the experimental setup is presented and the employed hardware components are outlined. Second, in Section 7.2 the controller tracking error is investigated for highly dynamic positioning tasks. Third, in Section 7.3, the structural vibrations of the prototype robot are analyzed. Furthermore, the vibration reduction achieved by the methods discussed in Chapter 5 are presented. Fourth, the behavior of the dart tracking algorithm and the accuracy of the impact prediction are investigated in Section 7.4. Finally, the reliability of the dart catching robot is demonstrated in Section 7.5.

7.1 Dartrobot experimental setup

The hardware setup used to perform experiments is shown in Figure 7.1. Here, some relevant components are labelled using the nubers ① - ⑥. To move the dartboard, the cable-driven parallel robot (CDPR), designed and described in [10], is employed. A *Unicorn Eclipse Pro* tournament dartboard is attached to the end effector of the CDPR. The IR cameras (*Optitrack Prime 17W*) which are used to capture the position of the flying dart are positioned at different heights on tripods. Each IR camera is connected via Ethernet to a switch (*Netgear GS728TPP*), which is not visible in Figure 7.1. The switch connects all 6 cameras to the PC system contained inside the electrical cabinet of the robot via a single Ethernet cable. To achieve high flexibility and good performance, a custom PC based on the *Intel i7-12700K* CPU was fitted into the electrical cabinet. This PC processes the camera data provided by the IR cameras using the *Optitrack Motive* software running in the Windows operating system. The robot controller is implemented on the same PC system using the *Beckhoff TWINCAT 3 eXtended Automation Runtime (XAR)*. Thus, the robot controller is executed independently from the Windows operating system to achieve the required real-time behavior. The robot controller sends torque and enable commands to the motor controllers (*Beckhoff AX8206*) via EtherCAT. For this purpose, the PC is equipped with a suitable network card compatible with the EtherCAT protocol.

To document and record the flight and the impact location of the dart, a DSLR camera and a compact high-speed camera are used.

In this work, *Winmau Barbarian 20g* stainless steel darts are used. Two different IR marker designs are tested, as shown in Figure 7.2. In the first design shown in Figure 7.2(a), reflective tape is wrapped around the dart to create areas which are highly reflective. The second design, shown in Figure 7.2(b) is created by attaching spherical reflective markers to the dart.



①	IR cameras	④	Cable-driven parallel robot
②	High-speed camera	⑤	Dartboard attached to end effector
③	Camera	⑥	Electrical cabinet

Figure 7.1: Experimental setup for catching a flying dart.

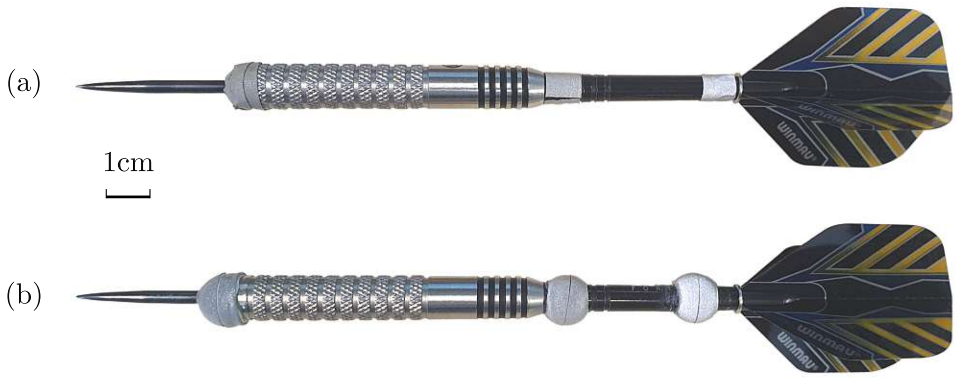


Figure 7.2: (a) Dart with reflective tape. (b) Dart with spherical reflective markers.

7.2 Controller tracking error

In this section, the dynamical tracking error of the controller is studied experimentally. Furthermore, the behavior of the controller is investigated by examining the forces and torques obtained from the control law.

7.2.1 Translation

To investigate the controller tracking error for translational motion, an experiment consisting of a positioning task with the parameters

$$\mathbf{x}_0 = \mathbf{0}\,, \quad \mathbf{x}_{\mathrm{set}} = \begin{bmatrix} 0.2\,\mathrm{m} & 0.2\,\mathrm{m} & 0\,\mathrm{rad} \end{bmatrix}^{\mathrm{T}}\,, \quad v_{\max} = 2.5\,\mathrm{m/s}\,, \quad a_{\max} = 30\,\mathrm{m/s^2}\,, \tag{7.1}$$

is conducted. The trajectory is generated by the OTG algorithm presented in Chapter 4 using the FIR filter presented in Section 4.4.4. Figure 7.3 visualizes the trajectory tracking error defined in Eq. (3.12) along the trajectory. In Figure 7.3(a), the distance travelled by the EE is shown. Here, the EE accelerates in Section Ⓐ and decelerates in Section Ⓑ. Due to the symmetry of the motion, the trajectory satisfies $x_{\mathrm{d}}(t) = y_{\mathrm{d}}(t) = r_{\mathrm{d}}(t)$. Figure 7.3(a) shows that the actual EE positions x_C and y_C closely follow the desired trajectory.

The velocity profile of the motion is depicted in Figure 7.3(b). Here, the desired velocity is denoted $\dot{x}_{\mathrm{d}}(t) = \dot{y}_{\mathrm{d}}(t) = v_{\mathrm{d}}(t)$. While the actual velocities $\dot{x}_C$ and $\dot{y}_C$ follow the trajectory closely, there is some visible velocity tracking error at the beginning and at the end of the acceleration and deceleration phases. This velocity tracking error occurs when the acceleration changes rapidly i. e. when the jerk is large.

The controller tracking error from Eq. (3.12) is shown in Figure 7.3(c). Both components e_x and e_y of the trajectory tracking error exhibit the largest values during phases of large jerk, which occur at the beginning and end of Sections Ⓐ and Ⓑ. The trajectory tracking error remains below 1.5 mm at all times. Furthermore, the component e_y initially has a value of $\approx 0.5\,\mathrm{mm}$ before the motions starts due to static friction. In addition, the component e_x shows the effect of structural vibrations of the robot frame. When the motion stops at the end of Section Ⓑ, the frame vibrates in x-direction. This vibration causes reaction forces on the EE which result in a small oscillation of e_x as visible in Figure 7.3(c).

Figure 7.3(d) shows the velocity tracking error $\dot{e}_x$ and $\dot{e}_y$. Similarly to the behavior of the trajectory tracking error, the peak values of the velocity tracking error are reached during phases of large jerk. The effect of the structural vibrations in x-direction are visible in $\dot{e}_x$ after the deceleration phase Ⓑ.

To investigate the behavior of the controller presented in Section 3.2, the PD+ control law for the EE is examined in detail. The desired task space force $\mathbf{f}_{\mathrm{d}}$ is given by Eq. (3.13) and can be partitioned in a feedforward component and a feedback component in the form

$$\mathbf{f}_{\mathrm{d}} = \begin{bmatrix} F_{x,\mathrm{d}} & F_{y,\mathrm{d}} & M_{z,\mathrm{d}} \end{bmatrix}^{\mathrm{T}} = \underbrace{\mathbf{M}_{\mathrm{EE}}\ddot{\mathbf{x}}_{\mathrm{d}} + \mathbf{g}}_{\text{feedforward}} \underbrace{- \mathbf{K}_{\mathrm{P}}\mathbf{e}_{\mathrm{x}} - \mathbf{K}_{\mathrm{D}}\dot{\mathbf{e}}_{\mathrm{x}}}_{\text{feedback}} \tag{7.2}$$

The forces calculated by the controller are visualized in Figure 7.4. Here, the stacked area plots from Figures 7.4(a) and 7.4(b) illustrate the individual terms highlighted in Eq. (7.2)

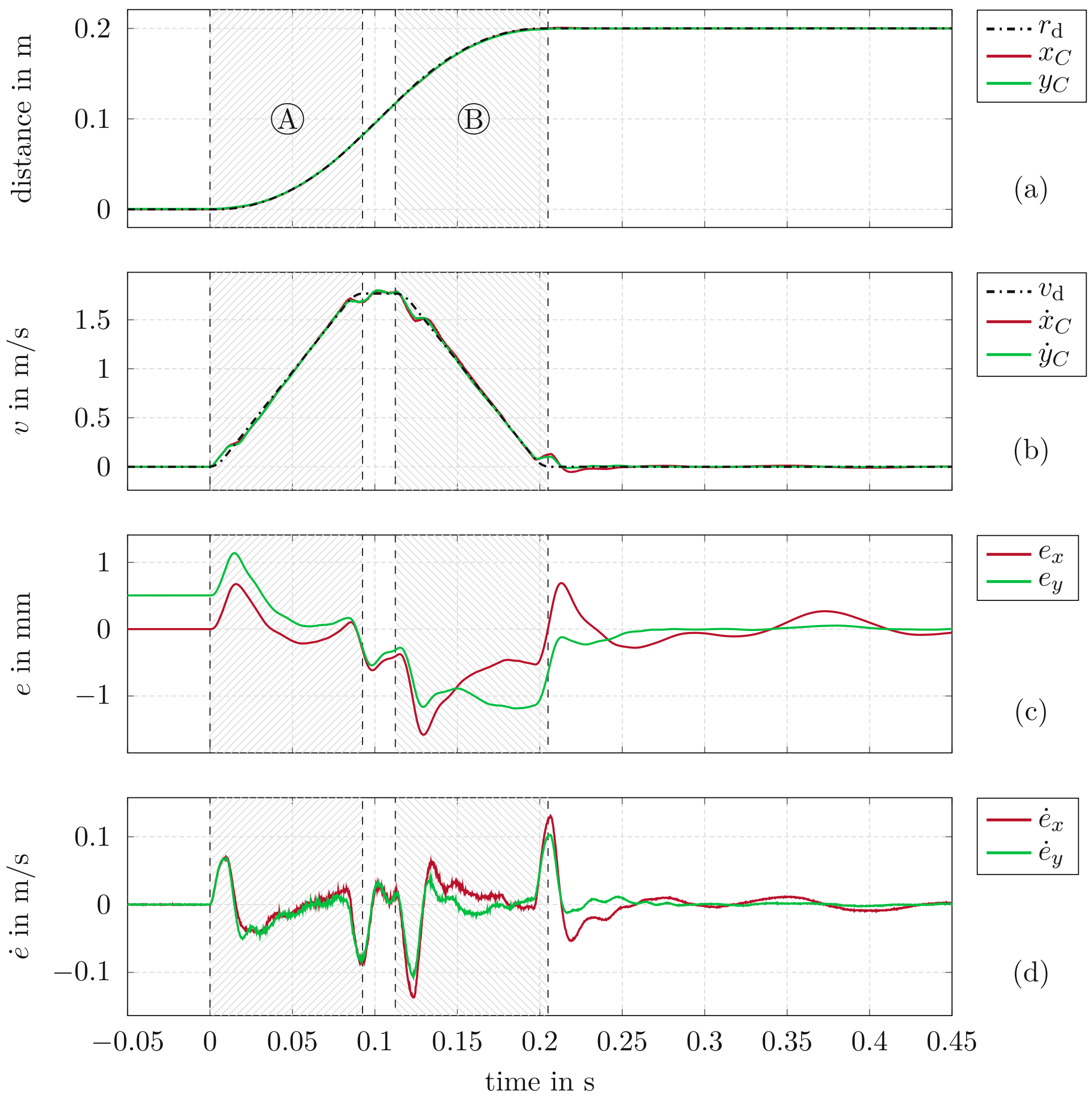


Figure 7.3: Experimental results: (a) Trajectory x_C and y_C. (b) Velocity $\dot{x}_C$ and $\dot{y}_C$. (c) Position error e_x and e_y. (d) Velocity error $\dot{e}_x$ and $\dot{e}_y$.

for the desired task space forces $F_{x,\mathrm{d}}$ and $F_{y,\mathrm{d}}$, respectively. In each stacked area plot, the shaded areas correspond to the individual force component of the controller and the lines show the cumulative sum up to the corresponding force component. By inspecting the task space force components that make up the total force $\mathbf{f}_\mathrm{d}$, the behavior of the controller can be investigated in detail.

The cable forces τ_i are shown in Figure 7.4(c) and the resulting motor torques given by the control law Eq. (3.15) are depicted in Figure 7.4(d). The majority of the task space force is provided by the feedforward components while the feedback terms correct for model mismatch and disturbances. Initially, the control error e_y is positive due to static friction. Thus, the P-feedback term $-K_\mathrm{P} e_y$ acts against the gravity compensation term $m_\mathrm{EE} g$ because the static friction supports the EE and acts against gravity. During the acceleration phase $\textcircled{A}$, the feedforward forces $\mathbf{M}_\mathrm{EE}\ddot{\mathbf{x}}_\mathrm{d}$ are too large and the feedback terms are negative to correct the mismatch. This mismatch in the feedforward components can be explained by two effects. First, the model parameters are subject to uncertainties. In particular, the inertia of the rotating parts of the motors and cable spools is relatively inaccurate. Second, the torque generated by the electric motors is not measured directly but estimated from the motor currents by the motor controllers using an observer. The motor model used by this observer is prone to uncertainties and errors. Especially the temperature dependence of the magnetic flux provided by the permanent magnets induces relatively large uncertainties. Due to the relatively low continuous load during operation of the CDPR, the motor temperatures are comparatively low. Thus, the motor model tends to underestimate the generated torques and the actual torques tend to be larger than the estimated torques.

The force distribution algorithm employed in the controller ensures, that the cable forces shown in Figure 7.4(c) are strictly positive. For the chosen coordinate system and sign convention, positive cable forces correspond to negative motor torques in the static case as visible in Figure 2.2. However, due to the winch inertia compensation in Eq. (3.15), the motor torques T_3 and T_4 are positive during the acceleration phase $\textcircled{A}$ as shown in Figure 7.4(d).

7.2.2 Rotation

To study the trajectory tracking error for rotational motion, the positioning task

$$\mathbf{x}_0 = \mathbf{0}\ , \quad \mathbf{x}_\mathrm{set} = \begin{bmatrix} 0\,\mathrm{m} & 0\,\mathrm{m} & \frac{\pi}{2}\mathrm{rad} \end{bmatrix}^\mathrm{T}\ , \quad \omega_\mathrm{max} = 12\,\mathrm{rad/s}\ , \quad \dot{\omega}_\mathrm{max} = 120\,\mathrm{rad/s^2}\ , \tag{7.3}$$

is performed. Similarly to the previous experiment, the trajectory is generated by the OTG algorithm presented in Chapter 4 using the FIR filter from Section 4.4.4. Figure 7.5 visualizes the trajectory tracking behavior. In Figure 7.5(a), the rotation angle is illustrated. The angular velocity increases in the acceleration phase $\textcircled{A}$ and decreases in the deceleration phase $\textcircled{B}$. The actual rotation angle φ closely follows the trajectory φ_d. Figure 7.5(b) depicts the angular velocity profile and shows small deviations between the trajectory $\dot{\varphi}_\mathrm{d}$ and the actual angular velocity $\dot{\varphi}$ of the EE.

The trajectory tracking error e_φ is shown in Figure 7.5(c). Here, a constant error with a magnitude of approximately 8 mrad can be observed before the motion starts and after

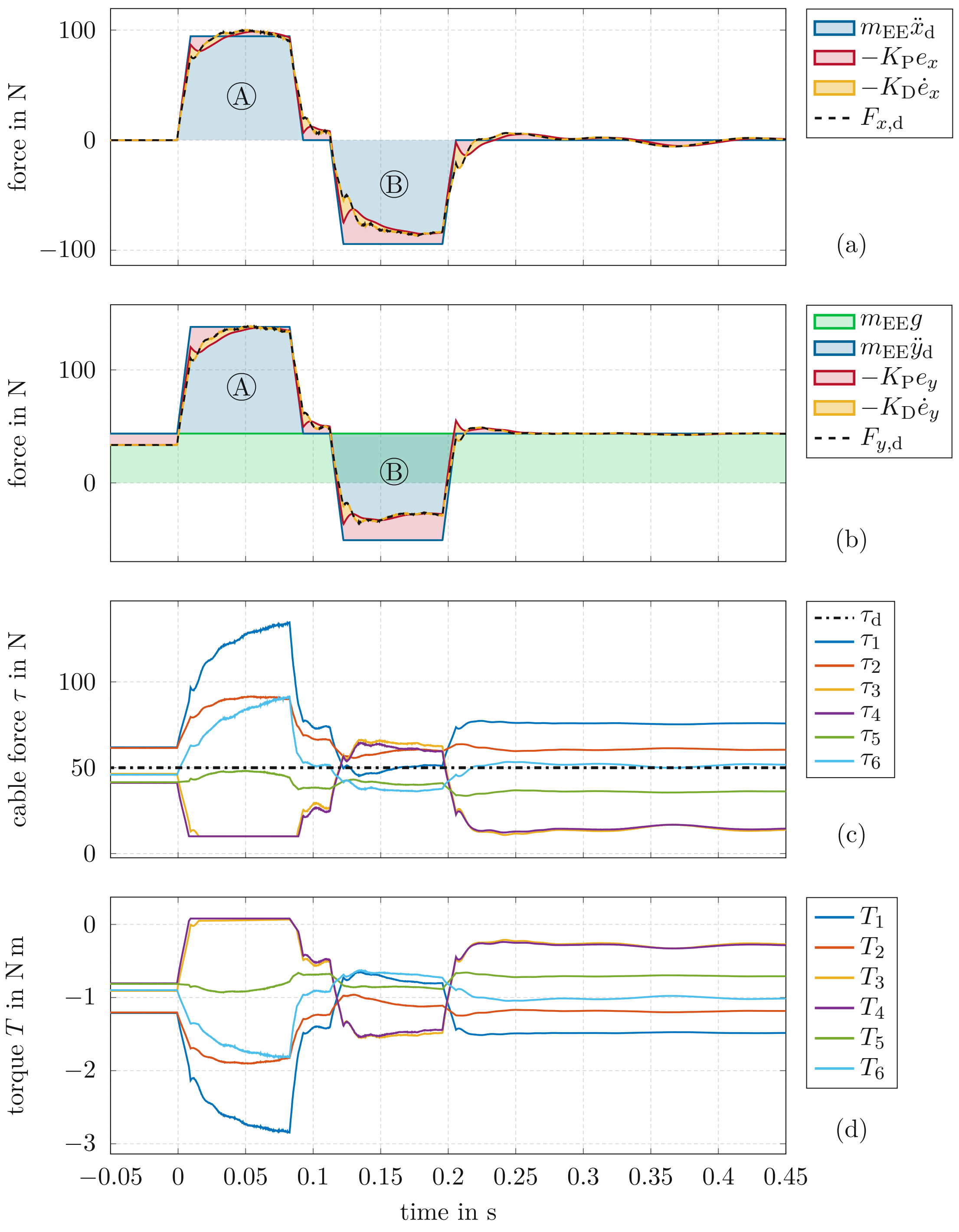


Figure 7.4: Experimental results: (a) Task space force components in x-direction. (b) Task space force components y-direction. (c) Cable forces τ. (d) Motor torques T.

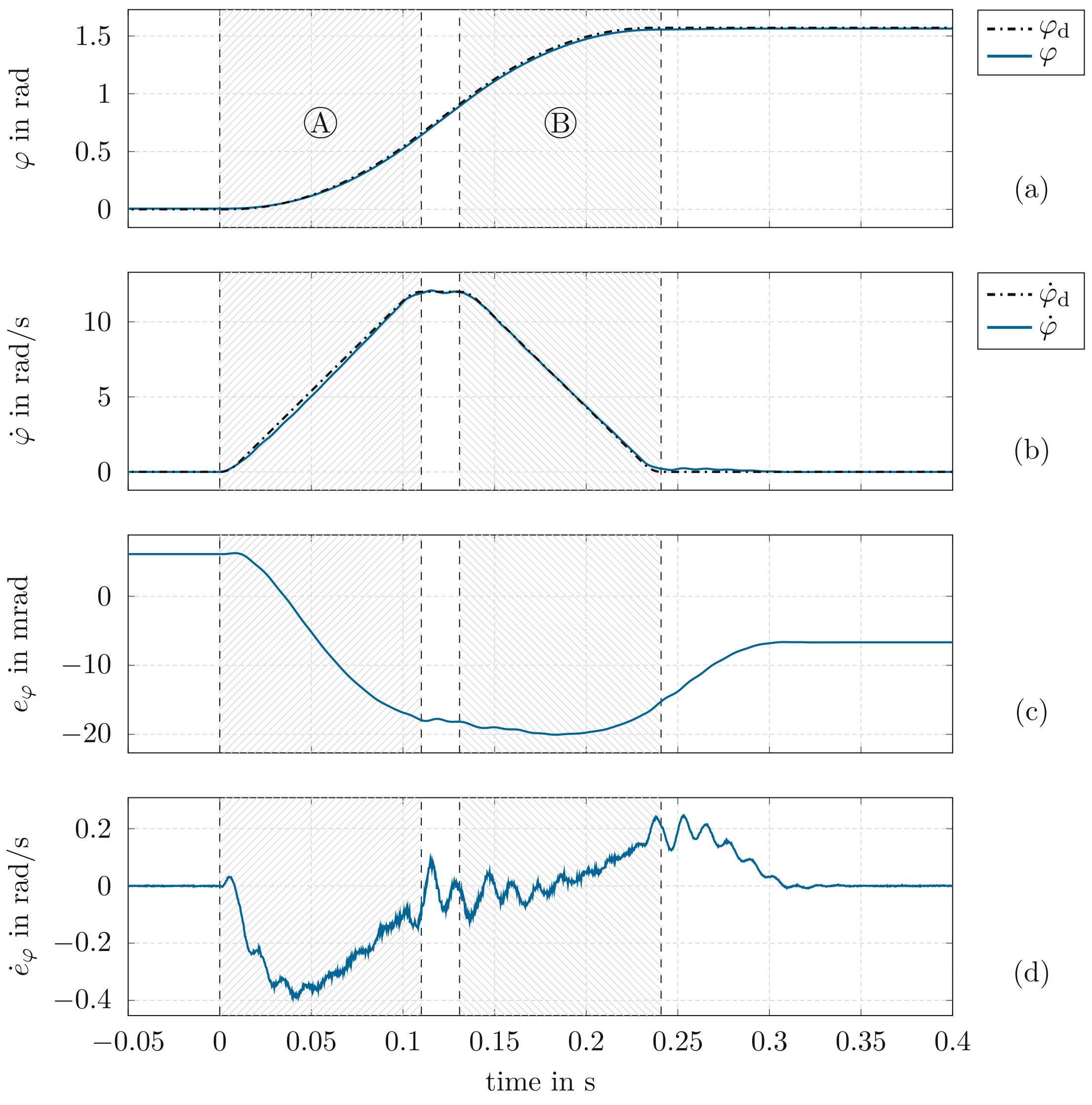


Figure 7.5: Experimental results: (a) Trajectory φ. (b) Angular velocity $\dot{\varphi}$. (c) Position error e_φ. (d) Angular velocity error $\dot{e}_\varphi$.

the motion is completed. This error is caused by static friction between the EE and the guiding surface. The trajectory following error e_φ is negative during the majority of the motion indicating that the EE angle φ slightly lags behind the trajectory φ_d. Figure 7.5 depicts the angular velocity tracking error $\dot{e}_\varphi$. Here, oscillations with a frequency in the range of 60-70 Hz can be observed. These oscillations likely correspond to the natural frequency of the elastic cables, which form a vibratory system with the inertia of the EE.

Figure 7.6 visualizes the behavior of the controller for rotational motion. The individual components of the desired task space torque $M_{z,\mathrm{d}}$ are illustrated in Figure 7.6(a) as a stacked area plot. Here, each shaded area corresponds to a task space torque component and each line visualizes the cumulative sum up to the corresponding torque component. Furthermore, the cable forces τ_i are shown in Figure 7.6(b). Figure 7.6(c) shows the corresponding motor torques T_i calculated via the control law from Eq. (3.15). The

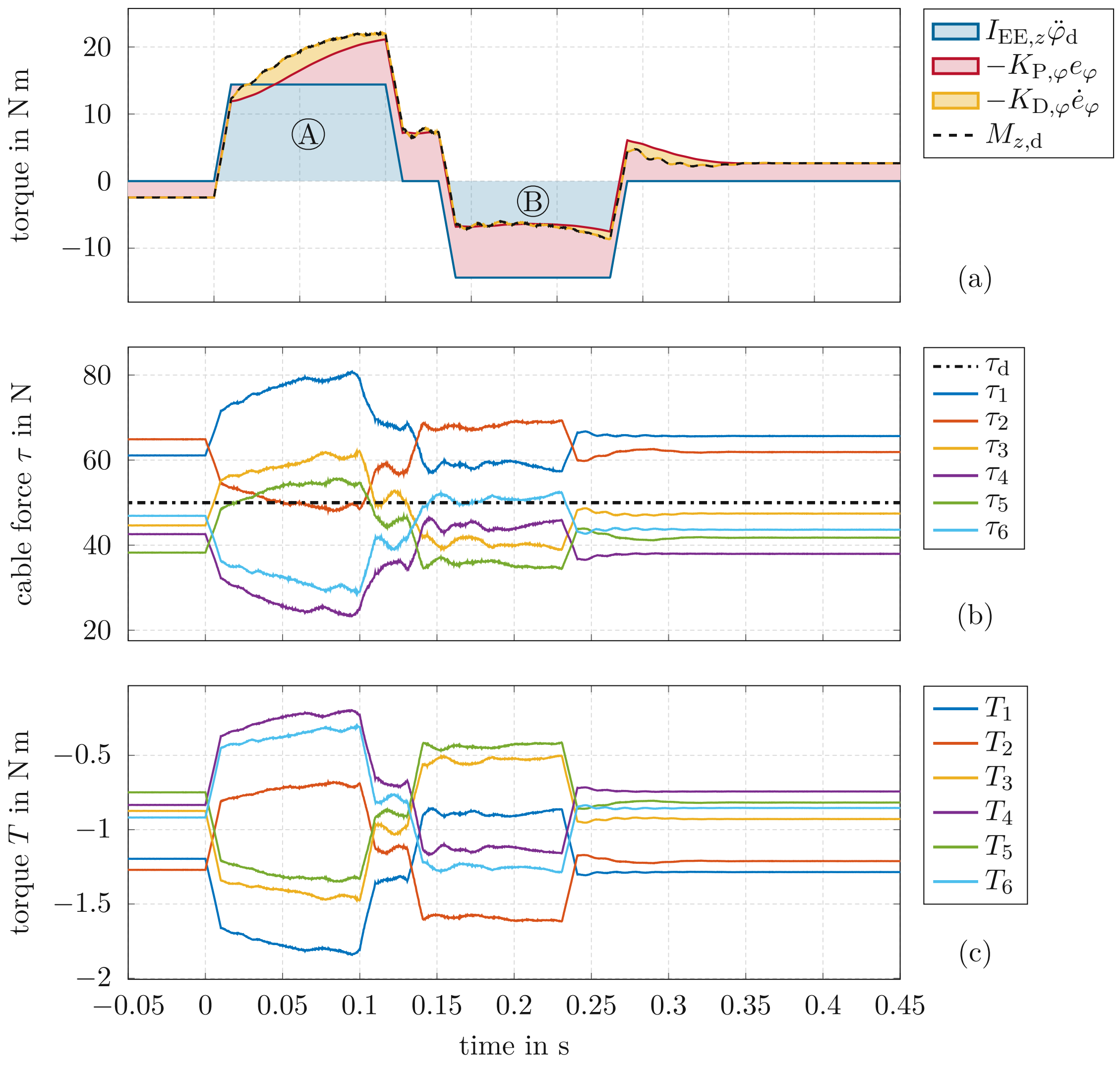


Figure 7.6: Experimental results: (a) Task space torque components. (b) Cable forces τ. (c) Motor torques T.

experimental results depicted in Figure 7.6(a) show, that the friction torque is similar to the inertial torque necessary to achieve the desired angular acceleration. For translational motion, the feedforward component provides the majority of the necessary task space force. However, due to the comparatively large effect of friction for the chosen rotational motion, the magnitude of the feedback component is similar to the magnitude of the feedforward component for the rotational motion. During the acceleration phase Ⓐ, the feedforward torque $I_{\mathrm{EE},z}\ddot{\varphi}_{\mathrm{d}}$ is smaller than the required torque to follow the trajectory due to the lack of a friction model. Thus, the feedback components $-K_{\mathrm{P},\varphi}e_{\varphi}$ and $-K_{\mathrm{D},\varphi}\dot{e}_{\varphi}$ assume positive values to provide the additional torque necessary to overcome friction. In the deceleration phase Ⓑ the negative feedforward braking torque is too large because unmodelled friction effects additionally contribute to a deceleration of the EE rotation. Consequently, the feedback component $-K_{\mathrm{P},\varphi}e_{\varphi}$ reduces the magnitude of the negative braking torque to follow the trajectory.

7.3 Vibration reduction

In this section, the structural vibrations of the CDPR frame are analyzed and measured. Subsequently, the vibration reduction achieved using the methods from Chapter 5 is investigated.

To excite the dominant horizontal vibration mode of the structure, a horizontal motion in x-direction with large acceleration is executed. The trajectory for the motion is generated using the OTG algorithm from Chapter 4 using the parameters

$$\mathbf{x}_0 = \mathbf{0}\ , \quad \mathbf{x}_{\mathrm{set}} = \begin{bmatrix} 0.1\,\mathrm{m} & 0\,\mathrm{m} & 0\,\mathrm{rad} \end{bmatrix}^{\mathrm{T}}\ , \quad v_{\mathrm{max}} = 2.5\,\mathrm{m/s}\ , \quad a_{\mathrm{max}} = 30\,\mathrm{m/s^2}\ . \tag{7.4}$$

Note that the motion described by Eq. (7.4) is identical to the motion from Eq. (5.1) for better comparability. This motion is a typical positioning task for catching a dart.

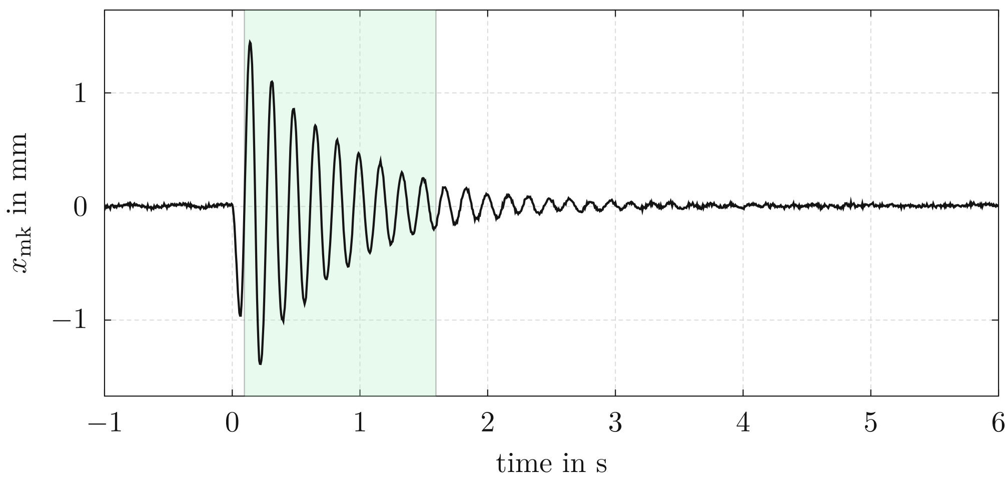
The motion of the frame is captured by attaching an optical marker to the square frame and measuring its position $(x_{\mathrm{mk}}, y_{\mathrm{mk}}, z_{\mathrm{mk}})$ using the IR camera system. This data can be used to find suitable parameters for the linear oscillator model from Eq. (5.5). Figure 7.7 shows the structural vibration of the CDPR. The horizontal motion x_{mk} of the marker attached to the frame is depicted in Figure 7.7(a). A portion of the signal is selected and the Matlab `fit` function is used to fit the linear vibration model from Eq. (5.5) to the data using the Levenberg-Marquardt method. The parameters

$$A = 1.58\,\mathrm{mm}\ , \qquad \omega_0 = 36.9\,\mathrm{s}^{-1}\ , \qquad \zeta = 0.0373\ , \tag{7.5}$$

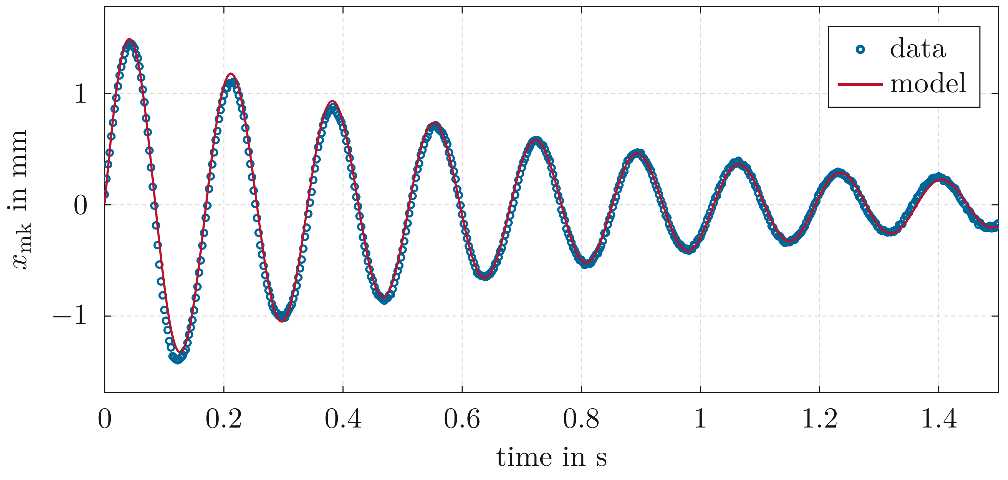
are obtained for the model. The response of the linear oscillator model with the parameters from Eq.(7.5) is shown in Figure 7.7(b). Here, it can be seen that the data matches the linear oscillator model very well. The resulting damped natural oscillation frequency of the dominating mode of vibration reads as

$$f_d = \frac{\omega_0}{2\pi}\sqrt{1-\zeta^2} = 5.87\,\mathrm{Hz}\ . \tag{7.6}$$

It should be noted that the height of the CDPR can be adjusted using a hand crank and the robot is equipped with casters. As a result, the vibration behavior of the structural



(a) Displacement of the CDPR frame.



(b) Linear vibration model.

Figure 7.7: Structural vibration of the CDPR.

components of the robot highly depends on the height, the properties of the ground and the position of the casters. Thus, the conservatively rounded values

$$f_0 = \frac{\omega_0}{2\pi} = 6\,\mathrm{Hz}\ , \qquad \zeta = 0.03\ , \tag{7.7}$$

are used to configure the input shaping methods as described in Chapter 5. To evaluate the effect of the input shaping methods, the motion from Eq. (7.4) is performed three times. First, no input shaping is used. Second, the notch filter from Section 5.2 is employed. Third, the zero vibration (ZV) input shaping method from Section 5.3 is used. The resulting EE motion x_C for all three cases is compared in Figure 7.8. Here, the EE position is recorded using the IR camera system and the forward kinematics. The forward kinematics calculates the position of the EE based on the cable lengths measured by the motor encoders. Thus, the motion of the robot frame does not impact the forward kinematics measurement. In comparison, the IR camera tracking system measures the motion of the EE relative to an inertial reference frame, which is calibrated such that it is aligned with the robot coordinate system used by the forward kinematics when the robot is at rest. The robot coordinate system is shown in Figure 2.1. For this reason, the motion of the robot frame impacts the camera tracking measurement.

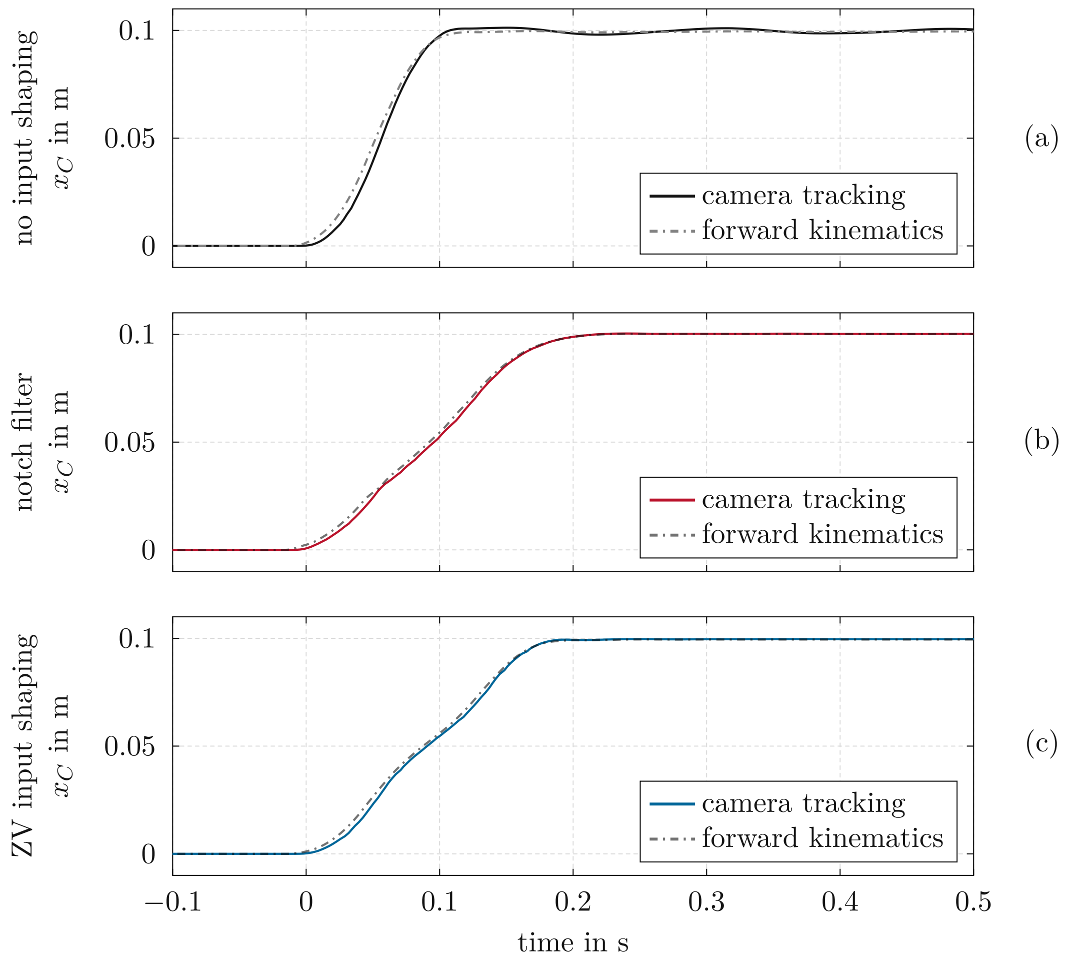


Figure 7.8: Comparison of EE motion with different input shaping methods. (a) No input shaping. (b) Notch filter. (c) ZV input shaping.

Figure 7.8(a) illustrates the EE motion without any input shaping. The trajectory requires approximately 110 ms to complete and there is a delay of a few milliseconds between the forward kinematics data and the camera tracking data caused by latencies in the camera tracking pipeline. In addition, the vibration of the robot frame causes the EE to oscillate relative to the inertial reference frame used by the camera system. Consequently, there are oscillations in the EE position measured by the camera system after the completion of the positioning task.

Figure 7.8(b) shows the effect of the notch filter on the motion. When using the notch filter, the trajectory requires approximately 220 ms to reach the set-point. After the motion has completed, the camera tracking data matches the forward kinematics data indicating greatly reduced vibrations of the robot frame.

When using the ZV input shaping method as depicted in Figure 7.8(c), the trajectory requires approximately 190 ms to complete. This duration is a result of the fixed duration of 83 ms of the input shaping FIR filter. Similarly to the notch filter, a vibration reduction of the robot frame can be observed after the motion has completed.

The horizontal vibrations of the marker x_{mk} attached to the robot frame are visualized in Figure 7.9. Figure 7.9(a) shows the vibration without any input shaping. In comparison, the resulting vibrations when using the notch filter are shown in Figure 7.9(b) and the vibrations when using the ZV input shaping method are depicted in Figure 7.9(c). Both

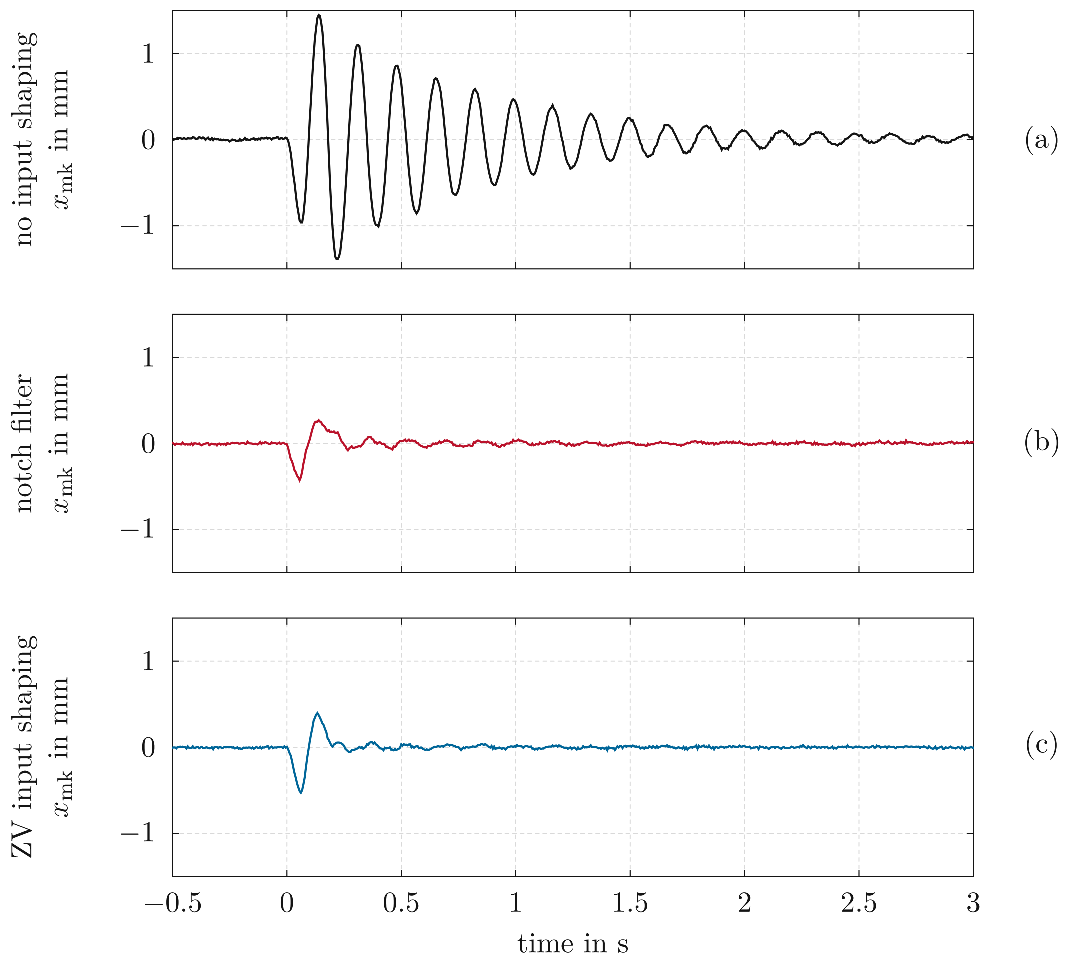


Figure 7.9: Effect of input shaping methods in time domain.
(a) No input shaping. (b) Notch filter. (c) ZV input shaping.

the notch filter and the ZV input shaping method cause a significant reduction of the structural vibrations of the robot frame. A displacement of the robot frame at the beginning of the motion is visible, which is caused by the reaction forces due to the acceleration of the EE.

To conclude the analysis of the achieved vibration reduction, the effect of the implemented input shaping methods on the frame vibrations is visualized in the frequency domain in Figure 7.10. Here, the discrete Fourier transform (DFT) of the marker dis-

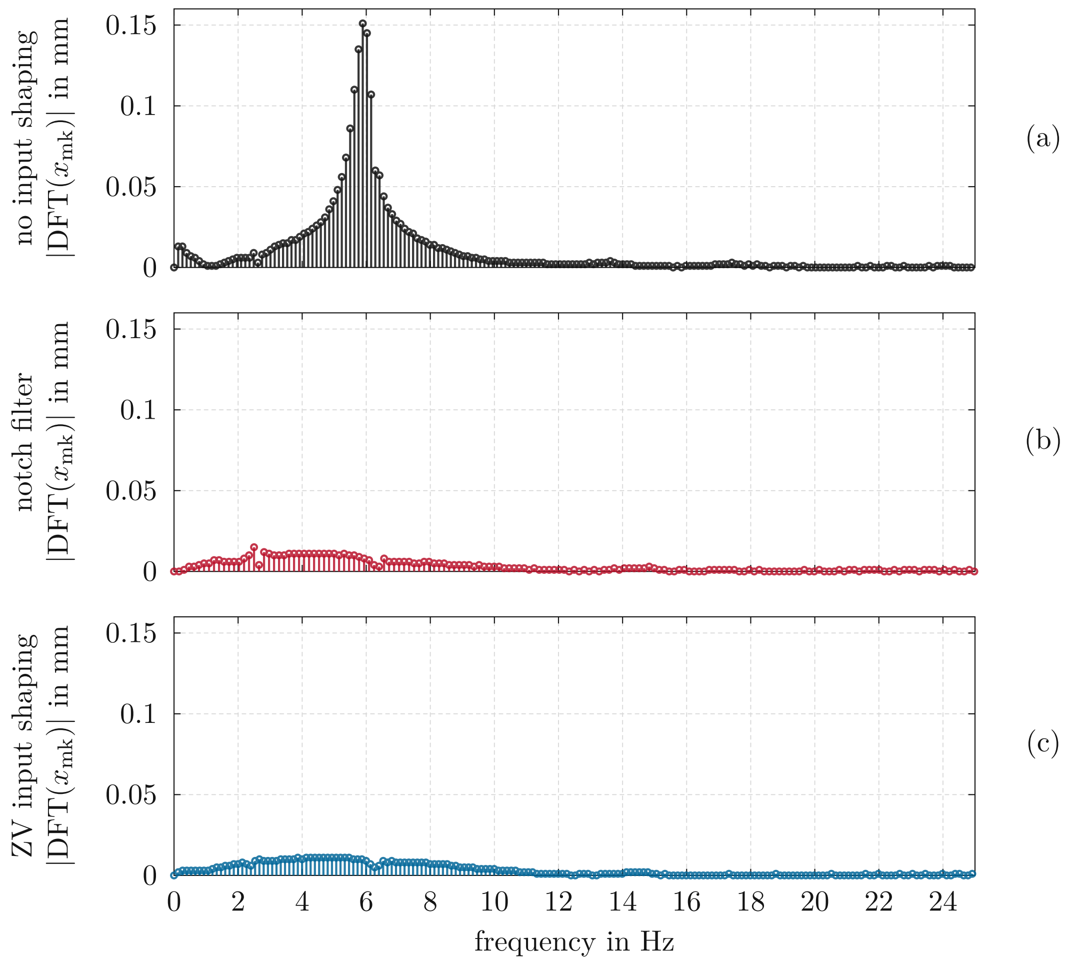


Figure 7.10: Effect of input shaping methods in frequency domain.
(a) No input shaping. (b) Notch filter. (c) ZV input shaping.

placement x_{mk} from Figure 7.9 is shown. Figure 7.10(a) shows the frequency spectrum of the vibration, where a sharp peak at the resonant frequency of $\approx 6\,\mathrm{Hz}$ is visible. Both the notch filter and the ZV input shaping method eliminate this peak in the spectrum as shown in Figures 7.10(b) and 7.10(c), respectively.

7.4 Dart catching experiment

To investigate the behavior of the flight prediction stage from Chapter 6 and demonstrate the effectiveness of the overall system, a dart catching experiment is conducted. The dart with reflective tape shown in Figure 7.2(a) is thrown by an amateur player from the official regulation distance used in dart sports tournaments. A summary of the model parameters used for the dart is provided in Table A.1. Other parameters used in the dart catching experiment can be found in Table A.3 for reference. The triple 20 segment, which is the most valuable scoring segment on the dartboard in the game of darts, is chosen as the target segment. The center of the triple 20 segment is parameterized using the values

$$r_\mathrm{t} = 0.103\,\mathrm{m}\ , \qquad \varphi_\mathrm{t} = 0\,\mathrm{deg}\ . \tag{7.8}$$

First, the behavior of the observer described in Section 6.4 is examined by comparing the state estimate $\hat{\mathbf{z}}$ to the measurement data. For this purpose, a reference estimate for the true dart state is computed by finding the dart position and orientation

$$\mathbf{z}_\mathrm{ref} = \begin{bmatrix} x_\mathrm{ref} & y_\mathrm{ref} & z_\mathrm{ref} & \vartheta_\mathrm{ref} & \psi_\mathrm{ref} \end{bmatrix}^\mathrm{T}\ , \tag{7.9}$$

which minimizes the Euclidean distance between the modelled marker positions $\mathbf{h}(\mathbf{z}_\mathrm{ref})$ and the measured marker positions $\mathbf{r}_F$, $\mathbf{r}_M$ and $\mathbf{r}_F$. To calculate the reference state $\mathbf{z}_\mathrm{ref}[j]$ for each measurement $\mathbf{y}[j]$, the unconstrained nonlinear least-squares problem

$$\mathbf{z}_\mathrm{ref}[j] = \arg\min_{\mathbf{z}} \quad \|\mathbf{h}(\mathbf{z}) - \mathbf{y}[j]\|_2 \tag{7.10}$$

is solved offline numerically for the captured sequence $\mathbf{y}[k]$. Here, the MATLAB command `lsqnonlin` is used which implements the Levenberg-Marquardt method. In case marker signals are missing, the last valid reference state is used for $\mathbf{z}_\mathrm{ref}$. To find a reference estimate for the dart velocities, the backwards difference quotient is used to approximate the first derivative of the dart position and orientation with respect to time. Hence, the reference dart velocities are given by

$$\dot{\mathbf{z}}_\mathrm{ref} = \frac{\mathbf{z}_\mathrm{ref}[j] - \mathbf{z}_\mathrm{ref}[j-1]}{t_j - t_{j-1}}\ . \tag{7.11}$$

Using the relation from Eq. (6.7), the angular velocities in the dart reference frame $\omega_{1,\mathrm{ref}}[j]$ and $\omega_{2,\mathrm{ref}}[j]$ can be computed.

Figure 7.11 visualizes the dart tracking behavior of the state observer by comparing the estimated position $\hat{x}_D$, $\hat{y}_D$, $\hat{z}_D$ and orientation $\hat{\vartheta}$, $\hat{\psi}$ to the reference states $\mathbf{z}_\mathrm{ref}$. The trowing process and flight of the dart can be divided into 3 sections. In Section ①, the dart is held by the human player and the player prepares to throw the dart. The dart leaves the hand at the time $t = 0\,\mathrm{s}$ and is airborne during Section ②. This phase can be divided into two parts (2a) and (2b). At the beginning of the flight in (2a), the state machine coordinating the flight prediction stage has not yet detected that the dart is airborne and thus the switching variable has the value $u = 0$. When the estimated distance $\hat{z}_D$ reaches the threshold $\hat{z}_D < 1.8\,\mathrm{m}$ in Section (2b), the state machine detects that the dart is in flight and sets the switching variable to $u = 1$. Thus, the duration of phase (2a) of approximately 80 ms can be interpreted as the time necessary to detect that the dart is airborne. Finally, in Section ③ the dart hits the dartboard. Due to the limited visibility of the markers in this stage as well as vibrations of the dart shortly after the impact, the IR tracking cameras provide only sporadic measurements of the marker locations in phase ③.

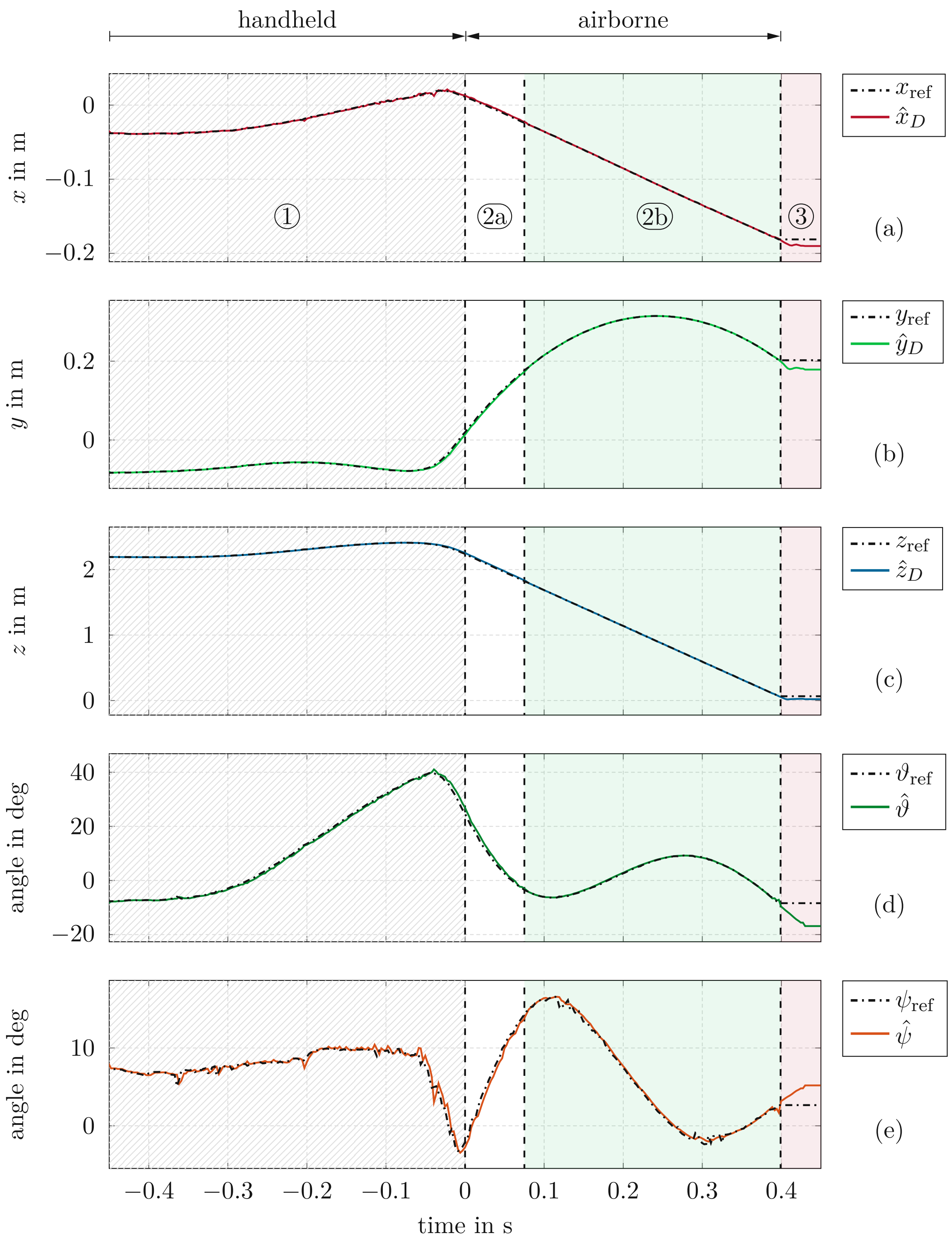


Figure 7.11: Experimental results: (a) Position tracking x. (b) Position tracking y. (c) Position tracking z. (d) Angle tracking ϑ. (e) Angle tracking ψ.

Figure 7.11 shows, that the observer closely tracks the motion of the dart. During Section $\textcircled{2a}$ when the dart is airborne but the observer uses the handheld model from Eq. (6.11) because $u = 0$, the model mismatch is large. Due to this model mismatch, the state estimates ($\hat{x}_D$, $\hat{y}_D$, $\hat{z}_D$, $\hat{\vartheta}$, $\hat{\psi}$) lag behind the reference states $\mathbf{z}_{\text{ref}}$. As soon as the model switches to the airborne stage by setting $u = 1$ in $\textcircled{2b}$, the observer uses the dart model from Eq. (6.21). In addition, a large uncertainty is assigned to the previous estimates by setting the estimation error covariance matrix $\mathbf{P}$ to a large value as stated in Eq. (6.39). As a result, the estimated states very closely track the reference states during phase $\textcircled{2b}$.

The behavior of the velocity estimates $\hat{\dot{x}}_D$, $\hat{\dot{y}}_D$, $\hat{\dot{z}}_D$ and the angular velocity estimates $\hat{\omega}_1$, $\hat{\omega}_2$ is illustrated in Figure 7.12. In Section $\textcircled{1}$ when the dart is handheld, the estimated velocities only poorly approximate the reference velocities. This can be explained by the unmodelled accelerations provided by the human hand holding the dart. At the end of phase $\textcircled{1}$, the human player rapidly accelerates the dart during the throwing process. Consequently, the velocity prediction error becomes particularly large. Due to the model mismatch in Section $\textcircled{2a}$, the estimated velocities $\hat{\dot{x}}_D$, $\hat{\dot{y}}_D$, $\hat{\dot{z}}_D$ and the angular velocity estimates $\hat{\omega_1}$, $\hat{\omega_2}$ show large deviations from the corresponding reference values. As soon as the model is switched to the airborne dart model from Eq. (6.21), i. e. the transition between $\textcircled{2a}$ and $\textcircled{2b}$, the estimates rapidly change and closely track the reference velocities.

Comparing the reference velocity $\dot{x}_{\text{ref}}$ shown in Figure 7.12(a) to the velocities $\dot{y}_{\text{ref}}$ from Figure 7.12(b) and $\dot{z}_{\text{ref}}$ from Figure 7.12(c) shows a larger uncertainty, i. e. more noise in the marker data in x-direction than in y- and z-direction. This asymmetry in the measurement error can be explained by the shallow viewing angle of the IR cameras. Due to the chosen camera positions depicted in Figure 7.1, any change in the x-coordinate of a marker results in a small change in the viewing angle of the cameras. Thus, the measurement accurracy with respect to motion in x-direction is comparably low.

Figure 7.13 visualizes the timing of the dart catching process and illustrates the behavior of the impact prediction step and the rotation planning step outlined in Sections 6.5 and 6.6, respectively. In Figure 7.13(a), the impact prediction $\hat{x}_{\text{p}}$, $\hat{y}_{\text{p}}$ is compared to the position of the dart's tip $x_{T,\text{ref}}$, $y_{T,\text{ref}}$. The tip position is obtained from the reference state $\mathbf{z}_{\text{ref}}$ using the relation from Eq. (6.40). In addition, the motion of the target field x_{t}, y_{t} is shown for comparison in Figure 7.13(a). The motion of the EE is shown in Figure 7.13(b)-(d). Thus, the behavior of the rotation planning algorithm is shown. The impact prediction shown in Figure 7.13(a) is considered valid when the dart has travelled 0.1 m and the threshold $\hat{z}_D < 1.7\,\text{m}$ is reached. This results in a delay of approximately 15 ms between the beginning of $\textcircled{2b}$ when the dart is detected as airborne and the first valid impact position estimate $\hat{x}_{\text{p}}, \hat{y}_{\text{p}}$. The delay is incorporated to avoid transient spikes in the impact prediction $\hat{x}_{\text{p}}$, $\hat{y}_{\text{p}}$ when the dart model is switched to the airborne stage. After the impact prediction reaches the first valid value, the prediction remains close to constant for the duration of the dart flight. This indicates that the flight trajectory predicted by the aerodynamic dart model from Section 6.2 is in very good agreement with the actual flight trajectory. The rotation planning algorithm computes a suitable set-point pose $\mathbf{x}_{\text{set}}$ consisting of rotational motion and translational motion. The performed translational motion x_C, y_C of the EE center-point C is shown in Figure 7.13(b) and (c) while the

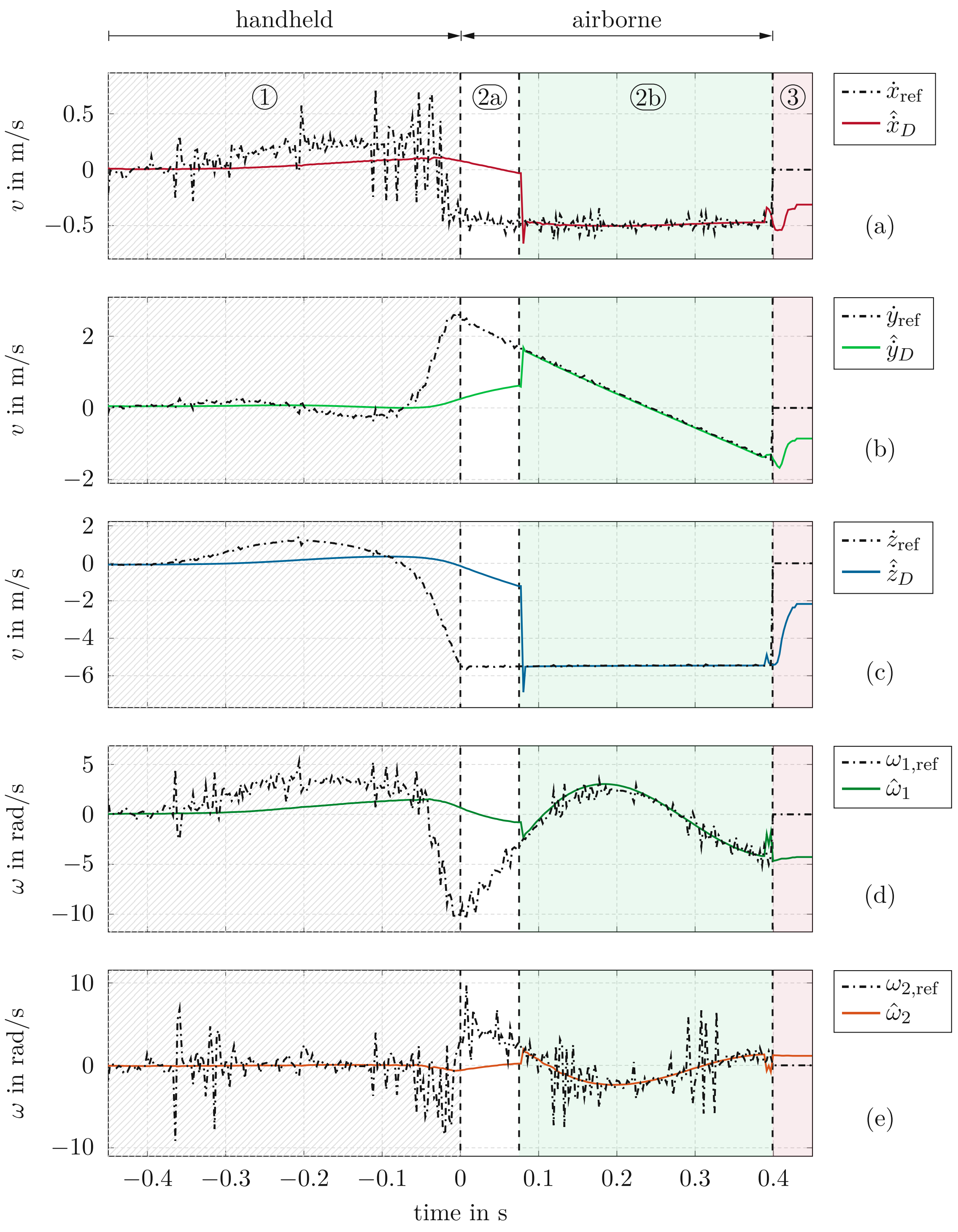


Figure 7.12: Experimental results: (a) Velocity estimation $\dot{x}_D$. (b) Velocity estimation $\dot{y}_D$. (c) Velocity estimation $\dot{z}_D$. (d) Angular velocity ω_1. (e) Angular velocity ω_2.

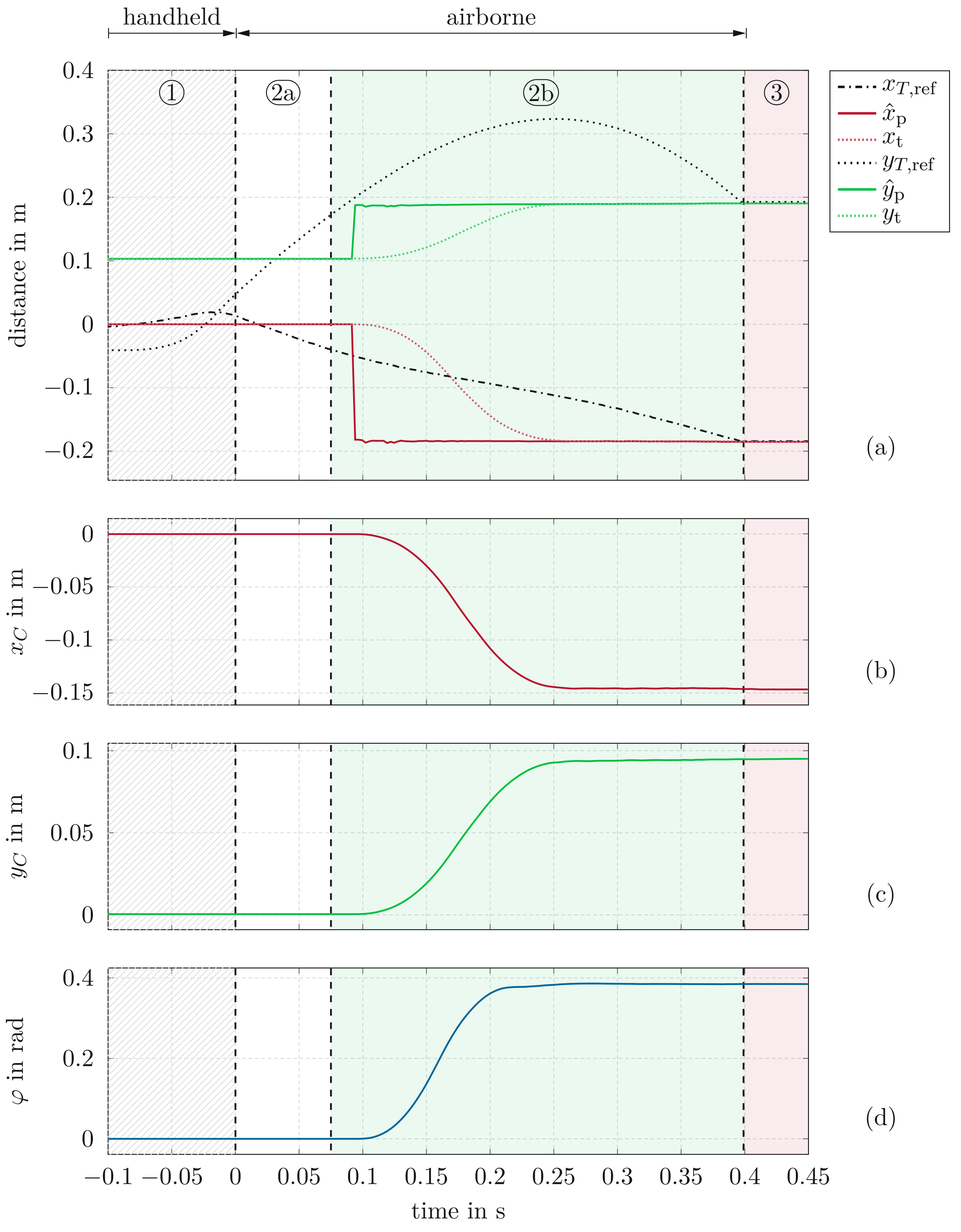
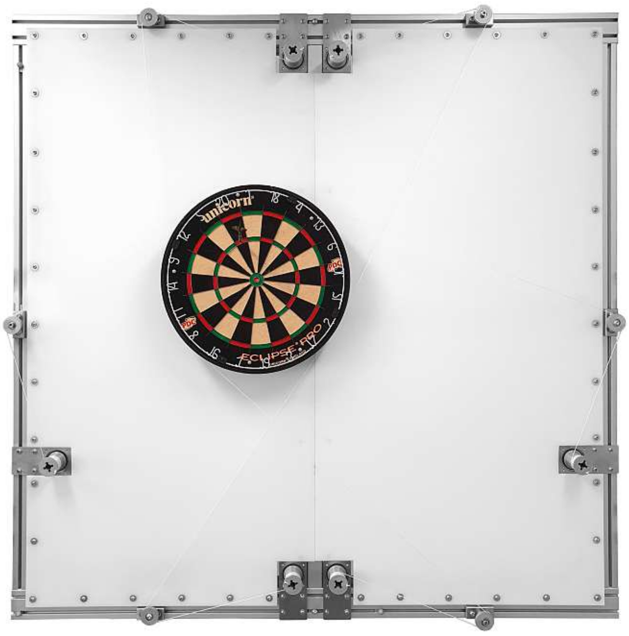


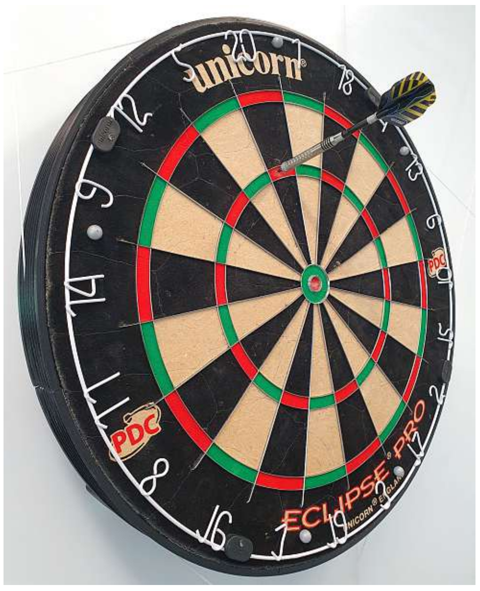
Figure 7.13: Experimental results: (a) Dart impact prediction. (b) Dartboard motion x_C. (c) Dartboard motion y_C. (d) Dartboard rotation φ.

rotational motion φ is visualized in Figure 7.13(d). Here, the translational and rotational motion complete approximately at the same time around 130 ms before the dart impact occurs. The combination of rotation and translation moves the target segment x_t, y_t to the predicted impact location as visible in Figure 7.13(a). After a total flight duration of approximately $t \approx 400\,\mathrm{ms}$, the tip of the dart reaches the dartboard very close to the predicted impact position as shown in Figure 7.13(a). It can be seen that the final prediction error is very small and the target field (triple 20) is positioned at the impact point in time.

Figure 7.14 shows the final impact position of the dart on the dartboard. In Figure 7.14(a), the final configuration of the dartboard and the chosen combination of rotation and translation can be observed. Figure 7.14(b) shows the impact location inside the triple 20 segment in more detail.



(a) Front view.



(b) Detail view.

Figure 7.14: Impact position of the dart.

7.5 Dart catching reliability

To demonstrate the reliability, repeatability and robustness of the developed dart catching robot, a sequence of 15 consecutive throws by three different amateur players is conducted. The robot is able to catch all 15 darts, such that they hit the target segment which was chosen as the triple 20 segment. A video of this experiment and an additional demonstration of the robot can be found at:

https://youtu.be/_WwZZbF93H4

For all experiments shown in the video, the dart with spherical markers depicted in Figure 7.2(b) was used. The experiments conducted in the course of this work suggest that both dart version from Figure 7.2(a) and Figure 7.2(b) perform similarly. Further investigation is necessary to conclusively determine which marker design performs better and to optimize the dart design.

8 Conclusion and outlook

In this work, the benchmark problem of catching a flying dart thrown by a human is tackled and solved. The flight trajectory of the dart is tracked via infrared (IR) cameras and the impact location is predicted. A cable-driven parallel robot (CDPR) is used to move a tournament dartboard accordingly, such that any desired field can be hit reliably. Thus, a technology demonstrator was created which showcases the performance of the CDPR and the algorithms involved. A control system architecture was developed by dividing the task of catching a dart into smaller sub-problems.

A mathematical model of the CDPR was presented in Chapter 2, which was used to design a trajectory tracking controller for the kinematically redundantly restrained robot in Chapter 3. The proposed control structure uses a force distribution algorithm to resolve the redundancies, which was studied in detail using simulations. The trajectory tracking behavior of the CDPR was evaluated and validated using experiments.

A real-time capable online trajectory generator (OTG) was developed for the application of planning and dynamically adapting a trajectory for the motion of the dartboard in Chapter 4. The behavior of the OTG algorithm and the duration and path of the generated trajectories were analyzed in detail to ensure that the trajectories generated by the algorithm are time-efficient and smooth.

To reduce structural vibrations excited by the rapid acceleration of the relatively heavy tournament dartboard, so-called input-shaping techniques were applied to the trajectories of the dartboard. Different input shaping methods were discussed and compared to each other in Chapter 5. The resulting vibration reduction was validated experimentally via measurements.

The flight of a steel dart compliant with dart sport tournament regulations was analyzed and a dynamical model was developed. Parameters for the model were identified from measurement data. Based on this model, an algorithm for tracking the flight trajectory and predicting the impact location on the dartboard was developed in Chapter 6.

In Chapter 7, various experiments were presented to validate the theoretical considerations. All components and algorithms were tested and examined individually. In addition, experiments were conducted to demonstrate and validate the effectiveness and reliability of the overall dart catching robot. These experiments showed that the developed system is capable of reliably performing the challenging task of catching a flying dart.

While the robot presented in this master's thesis performs well, future work could be devoted to improve various aspects of the dart catching robot and extend its capabilities. In darts, a regular turn of each player consists of 3 consecutive throws. The present work only considers a single dart throw. The employed algorithms could be adapted and extended to consider a whole turn with 3 darts. This opens the possibility of many extensions and improvements. The tracking algorithm could be extended to reliably track the motion of 3 darts and a strategy for optimally catching these 3 darts such that already caught darts do not interfere with the catching process could be developed.

Furthermore, future work could improve the motion of the robot to improve the time-efficiency or reduce vibrations and noise. In addition, possible extensions could aim at increasing the precision and reliability of the robot. This can be achieved by either increasing the positioning accuracy of the dartboard or the prediction accuracy of the flight trajectory of the dart. The positioning accuracy could be potentially increased by improving the model of the CDPR and incorporating nonlinear effects such as friction, elastic deformations and nonlinear geometric effects. Alternatively, the position of the dartboard could be measured via the optical tracking system used for tracking the dart and this data could be incorporated into the control system. On the other hand, the prediction accuracy for the dart impact could be improved by improving the dart model or improving the IR camera system by adding more cameras or improving the usage of the existing cameras. Alternatively, a totally different measurement system for tracking the dart could be employed. An optical tracking system could be developed which does not rely on IR markers and potentially offers higher reliability and performance for tracking the dart.

A Appendix parameter values

Parameter	Symbol	Value	Unit
Tip length	l_T	30	mm
Shaft length	l_S	29.9	mm
Barrel length	l_B	55.6	mm
Center of gravity distance	l_D	27	mm
Aerodynamic lift constant	C_f	65	mm^{-1}
Aerodynamic damping constant	C_d	0.008	1
Aerodynamic length constant	C_l	115	mm^{-1}

Table A.1: Dart model parameters.

Parameter	Symbol	Value	Unit
Disturbance covariance matrix	$\mathbf{Q}_1$	$\mathrm{diag}([1, 1, 1, 10, 10, 10, 10, 10, 100, 100])$	$\mathrm{m}^2, \mathrm{m}^2, \mathrm{m}^2, \mathrm{m}^2/\mathrm{s}^2, \mathrm{m}^2/\mathrm{s}^2, \mathrm{m}^2/\mathrm{s}^2$ $1, 1, \mathrm{s}^{-2}, \mathrm{s}^{-2}$
Disturbance covariance matrix	$\mathbf{Q}_2$	$\mathrm{diag}([0.01, 0.01, 0.01, 0.1, 0.1, 0.1, 0.3, 0.3, 0.3, 0.3])$	$\mathrm{m}^2, \mathrm{m}^2, \mathrm{m}^2, \mathrm{m}^2/\mathrm{s}^2, \mathrm{m}^2/\mathrm{s}^2, \mathrm{m}^2/\mathrm{s}^2$ $1, 1, \mathrm{s}^{-2}, \mathrm{s}^{-2}$
Noise parameter	R_{mk}	20	m

Table A.2: Observer parameters.

Parameter	Symbol	Value	Unit
Number of steps	N_{p}	80	1
Rotation parameter	K_φ	0.5	1
Velocity limit	$v_{\max}$	2.5	m/s
Acceleration limit	$a_{\max}$	30	m/s^2
Input shaping method	–	none	–

Table A.3: Dart catching experiment parameters.

Bibliography

[1] W. Decker and A. Guttmann, *Sports and Games of Ancient Egypt*. Yale University Press, 1992.

[2] S. Mori, K. Tanaka, S. Nishikawa, R. Niiyama, and Y. Kuniyoshi, "High-speed and lightweight humanoid robot arm for a skillful badminton robot," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1727–1734, 2018.

[3] M. Hattori *et al.*, "Fast tennis swing motion by ball trajectory prediction and joint trajectory modification in standalone humanoid robot real-time system," *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3612–3619, 2020.

[4] A. R. M. Khairudin *et al.*, "Design and control of an articulated robotic arm for archery," *Proceedings of the 2022 IEEE 5th International Symposium in Robotics and Manufacturing Automation*, pp. 1–5, 2022.

[5] J. Tian, H. Liu, S.-L. Dai, and C. Yang, "A real-time football goalkeeper robot system based on fuzzy logic control," *Proceedings of the 2021 China Automation Congress*, pp. 3258–3263, 2021.

[6] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters, "Learning to play table tennis from scratch using muscular robots," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3850–3860, 2022.

[7] W. Gao *et al.*, "Robotic table tennis with model-free reinforcement learning," *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5556–5563, 2020.

[8] P. Yang, Z. Zhang, H. Wang, and D. Xu, "Design and motion control of a ping pong robot," *Proceedings of the 2010 8th World Congress on Intelligent Control and Automation*, pp. 102–107, 2010.

[9] A. Kyohei, N. Masamune, and Y. Satoshi, "The ping pong robot to return a ball precisely," *Omron TECHNICS*, vol. 51, pp. 1–6, 2020.

[10] G. Feiler, "Design, modeling and implementation of a cable driven parallel robot," Diploma Thesis, TU Wien, 2023.

[11] B.-S. Kim and J.-B. Song, "Hybrid dual actuator unit: A design of a variable stiffness actuator based on an adjustable moment arm mechanism," *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pp. 1655–1660, 2010.

[12] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[13] C. Obayashi, T. Tamei, and T. Shibata, "Assist-as-needed robotic trainer based on reinforcement learning and its application to dart-throwing," *Neural Networks*, vol. 53, pp. 52–60, 2014.

[14] M. Linderoth, A. Robertsson, R. Johansson, *et al.*, "Vision based tracker for dart catching robot," *IFAC Proceedings Volumes*, vol. 42, no. 16, pp. 717–722, 2009.

[15] TU Munich, ITQ GmbH. "Mi5-dartboard." (2006), [Online]. Available: `https://www.itq.de/innovationen/demonstratoren/#hightech__mi5-showcase` (visited on 12/11/2022).

[16] F. Wolfslehner, "Entwicklung und Aufbau eines mechatronischen Hochgeschwindigkeitspositionierungssystems mit hydraulischem Antrieb," Diploma Thesis, JKU Linz, 2022.

[17] JKU Linz, INRAS GmbH, Linz Center of Mechatronics. "Magic darts." (2020), [Online]. Available: `https://ars.electronica.art/keplersgardens/en/magic-darts/` (visited on 01/09/2023).

[18] M. Rober. "Automatic bullseye dartboard." (2017), [Online]. Available: `https://www.markroberbuildinstructions.com/auto-bullseye` (visited on 12/10/2022).

[19] A. Pott, *Cable-Driven Parallel Robots*. Stuttgart: Springer Tracts in Advanced Robotics, 2018.

[20] K. Lynch and F. Park, *Modern Robotics: Mechanics, Planning and Control*. Cambridge: Cambridge University Press, 2017.

[21] L.-W. Tsai, *Robot analysis: the mechanics of serial and parallel manipulators*. New York: Wiley, 1999.

[22] M. Yuan and F. Freudenstein, "Kinematic analysis of spatial mechanisms by means of screw coordinates. part 1—screw coordinates," *Transactions of the ASME Journal of Engineering for Industry*, vol. 93, no. 1, pp. 61–66, 1971.

[23] R. Verhoeven and M. Hiller, "Tension distribution in tendon-based stewart platforms," *Advances in Robot Kinematics: Theory and Applications*, pp. 117–124, 2002.

[24] R. Verhoeven, "Analysis of the workspace of tendon based stewart platforms," Ph.D. dissertation, Duisburg, Essen, 2006.

[25] C. Gosselin and M. Grenier, "On the determination of the force distribution in overconstrained cable-driven parallel mechanisms," *Meccanica*, vol. 46, pp. 3–15, 2011.

[26] A. Pott, T. Bruckmann, and L. Mikelsons, "Closed-form force distribution for parallel wire robots," *Proceedings of the 5th International Workshop on Computational Kinematics*, pp. 25–34, 2009.

[27] A. Pott, "An improved force distribution algorithm for over-constrained cable-driven parallel robots," *Proceedings of the 6th International Workshop on Computational Kinematics*, pp. 139–146, 2014.

[28] T. F. Coleman and Y. Li, "A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables," *SIAM Journal on Optimization*, vol. 6, no. 4, pp. 1040–1058, 1996.

[29] D. Koditschek, “Natural motion for robot arms,” *Proceedings of the 23rd IEEE Conference on Decision and Control*, pp. 733–735, 1984.

[30] B. Paden and R. Panja, “Globally asymptotically stable ’PD+’ controller for robot manipulators,” *International Journal of Control*, vol. 47, no. 6, pp. 1697–1712, 1988.

[31] V. Santibañez and R. Kelly, “Global asymptotic stability of the PD control with computed feedforward in closed loop with robot manipulators,” *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 683–688, 1999.

[32] S. M. LaValle, *Planning algorithms*. Cambridge: Cambridge University Press, 2006.

[33] T. Kröger, *On-Line Trajectory Generation in Robotic Systems*. Berlin, Heidelberg: Springer Tracts in Advanced Robotics, 2010.

[34] R. Katzschmann, T. Kröger, T. Asfour, and O. Khatib, “Towards online trajectory generation considering robot dynamics and torque limits,” *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5644–5651, 2013.

[35] Q. Zhang, S. Li, J.-X. Guo, and X.-S. Gao, “Time-optimal path tracking for robots under dynamics constraints based on convex optimization,” *Robotica*, vol. 34, no. 9, pp. 2116–2139, 2016.

[36] G. Huber and D. Wollherr, “An online trajectory generator on SE(3) for human–robot collaboration,” *Robotica*, vol. 38, no. 10, pp. 1756–1777, 2020.

[37] R. L. Andersson, “Aggressive trajectory generator for a robot ping-pong player,” *IEEE Control Systems Magazine*, vol. 9, no. 2, pp. 15–21, 1989.

[38] Z. Lin, V. Zeman, and R. V. Patel, “On-line robot trajectory planning for catching a moving object,” *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pp. 1726–1727, 1989.

[39] L. Berscheid and T. Kröger, “Jerk-limited real-time trajectory generation with arbitrary target states,” *Proceedings of Robotics: Science and Systems XVII*, 2021.

[40] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, “Automated tracking and grasping of a moving object with a robotic hand-eye system,” *IEEE Transactions on Robotics and Automation*, vol. 9, no. 2, pp. 152–165, 1993.

[41] L. Biagiotti and C. Melchiorri, “Trajectory generation via FIR filters: A procedure for time-optimization under kinematic and frequency constraints,” *Control Engineering Practice*, vol. 87, pp. 43–58, 2019.

[42] P. Besset, R. Béarée, and O. Gibaru, “FIR filter-based online jerk-controlled trajectory generation,” *Proceedings of the 2016 IEEE International Conference on Industrial Technology*, pp. 84–89, 2016.

[43] R. Zanasi, C. G. L. Bianco, and A. Tonielli, “Nonlinear filters for the generation of smooth trajectories,” *Automatica*, vol. 36, no. 3, pp. 439–448, 2000.

[44] J. E. Lloyd, “Trajectory generation implemented as a non-linear filter,” Department of Computer Science, University of British Columbia, Vancouver, Canada, Tech. Rep., 1998.

[45] M. Bonfè and C. Secchi, “Online smooth trajectory planning for mobile robots by means of nonlinear filters,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 4299–4304.

[46] M. M. G. Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, “Real-time trajectory generation using model predictive control,” *Proceedings of the 2015 IEEE International Conference on Automation Science and Engineering*, pp. 942–948, 2015.

[47] L. Akulenko and A. Koshelev, “Time-optimal steering of a point mass to a specified position with the required velocity,” *Journal of applied mathematics and mechanics*, vol. 71, no. 2, pp. 200–207, 2007.

[48] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, “Time-optimal control of robotic manipulators along specified paths,” *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

[49] R. Alkhatib and M. Golnaraghi, “Active structural vibration control: A review,” *Shock and Vibration Digest*, vol. 35, no. 5, p. 367, 2003.

[50] C. Conker, H. Yavuz, and H. H. Bilgic, “A review of command shaping techniques for elimination of residual vibrations in flexible-joint manipulators,” *Journal of Vibroengineering*, vol. 18, no. 5, pp. 2947–2958, 2016.

[51] J. Vaughan, A. Yano, and W. Singhose, “Performance comparison of robust negative input shapers,” *Proceedings of the 2008 American control conference*, pp. 3257–3262, 2008.

[52] S. Orfanidis, *Introduction to Signal Processing.* Upper Saddle River: Prentice Hall, 1996.

[53] N. Singer and W. Seering, “Preshaping command inputs to reduce system vibration,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 112, no. 1, pp. 76–82, 1990.

[54] W. E. Singhose, W. P. Seering, and N. C. Singer, “Shaping inputs to reduce vibration: A vector diagram approach,” *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 922–927, 1990.

[55] A. A. Pawar, K. S. Ranjan, A. Roy, and S. Saha, “Investigation of flowfield over a dart using smoke flow visualization,” *Proceedings of the 48th National Conference on Fluid Mechanics and Fluid Power*, pp. 99–103, 2023.

[56] D. James and J. Potts, “Experimental validation of dynamic stability analysis applied to dart flight,” *Sports Engineering*, vol. 21, pp. 347–358, 2018.

[57] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches.* Cleveland: John Wiley & Sons, 2006.

[58] K. Atkinson, W. Han, and D. E. Stewart, *Numerical solution of ordinary differential equations.* Iowa City: John Wiley & Sons, 2011.