PhD-FSTC-2018-50
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defence held on 10/07/2018 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN SCIENCES DE L'INGÉNIEUR

by

Jan DENTLER
Born on 24 December 1987 in Riedlingen (Germany)

REAL-TIME MODEL PREDICTIVE CONTROL OF COOPERATIVE AERIAL MANIPULATION

Dissertation defence committee

University of Luxembourg

Doctoral Thesis

Real-Time Model Predictive Control of Cooperative Aerial Manipulation

Author:
Jan Eric Dentler

Supervisors:
Prof. Dr.-Ing. Holger Voos
Dr. Somasundar Kannan
Prof. Dr. Gianluca Antonelli

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering

in the

Automation and Robotics Research Group
SnT - Interdisciplinary Centre for Security, Reliability and Trust

Granted by the Luxembourg National Research Fund

July 2018

Declaration of Authorship

I, Jan Eric Dentler, declare that this thesis titled, 'Real-Time Model Predictive Control of Cooperative Aerial Manipulation' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Jan Dentler

[illegible]

17 October 2018, Luxembourg

Abstract

The rapid development in the field of Unmanned Aerial Vehicles (*UAV*s) is driven by new applications in agriculture, logistics, inspection and smart manufacturing. The future keys in these domains are the abilities to autonomously interact with the environment and with other robotic systems. This thesis is providing control engineering solutions to contribute to these key capabilities.

The first step of this thesis is to develop an understanding of the dynamic behavior of *UAV*s. For this purpose, dynamic and kinematic models are presented to describe a *UAV*'s motion. This includes a kinematic model which is suitable for off-the-shelf *UAV*s and combines full 360° heading operation with a low computational complexity. The presented models are subsequently used to develop a nonlinear model predictive control *NMPC* strategy. In this context, the performance of several *NMPC* solvers and inequality constraint handling techniques is evaluated. The real-time capability and *NMPC* performance are validated with real *AR.Drone 2.0* and *DJI M100* quadrotors. This includes collision avoidance and advanced tracking scenarios. The design work-flow for the related control objectives and constraints is presented accordingly. As a next step, this *UAV NMPC* strategy is extended for a *UAV* with attached robotic arm. For this purpose, the forward kinematics of the robotic arm are developed and combined with the kinematic model of the *UAV*. The resulting *NMPC* strategy is validated in a grasping scenario with a real aerial manipulator. The final step of this thesis is the *NMPC* of cooperating *UAV*s. The computational complexity of such scenarios conflicts directly with the fast *UAV* dynamics. In addition, control objectives and system topologies can dynamically change. To address these challenges, this thesis presents the *DENMPC* software framework. *DENMPC* provides a computationally efficient central *NMPC* strategy that allows changing the control scenario at runtime. This is finally stated in the control of a real cooperative aerial manipulation scenario.

Keywords:
Nonlinear Model Predictive Control, **Aerial Manipulation**, **Cooperative Control**, **Task-based Control**, **Distributed Systems**, **Unmanned Aerial Vehicles**

Acknowledgments

First, I would like to thank my supervisor Professor Holger Voos who made this work possible. The liberty and support he gave me, was the foundation on which this work has been built. Further, I owe my deepest gratitude to my supervisor Somasundar Kannan for sharing his extensive literature knowledge and passion for control engineering with me. I would also like to thank Miguel Angel Olivares-Mendez for the support in the laboratory and hope all the nerve-wrecking initial controller tests did not leave permanent traces. A special thanks also to Seyed Amin Sajadi Alamdari whose door was always open for scientific discussions about optimization and who hooked me with CGMRES. Also a big thanks to Arun Annaiyan with whom I shed sweat, blood and tears in the trenches of sciences. In the context of this thesis, I want to express my profound gratitude to Christian Maurer for his time, effort and amazing feedback. I wish also to especially thank Professor Gianluca Antonelli, Professor Toshiyuki Ohtsuka and Professor Moritz Diehl for their help and remote support. Without their willingness to openly share and discuss research, this work would not have been possible. Thanks also to Dr. Martin Rosalie and Dr. Gregoire Danoy for their ideas, time and data throughout our cooperation which made it an excellent experience for me. Furthermore, I want to say thanks to all my colleagues, the fast and creative technical support as well as the amazing SnT administration staff members.

This work was supported by FNR "Fonds national de la Recherche" (Luxembourg) through AFR "Aides à la Formation-Recherche" Ph.D. grant scheme No. 9312118, Project "FLYMAN".

To my family Léa, Ernst, Mathilde and Anja.
To Klara, Dr. Josef Rapp,
Hermann Heinzelmann, Prof. Dr. Max Riederle and Prof. Dr.-Ing. Knut Graichen
who set my path.

"Mathematics is a kind of toy which nature threw to us
for comfort and for entertainment in the darkness."
- Jean Baptiste le Rond d'Alembert

Contents

Declaration of Authorship v

Abstract vii

Acknowledgments viii

List of Tables xiv

List of Figures xvi

List of Abbreviations xxi

Glossary xxiv

1 Introduction **1**

1.1 Problem statement 2

1.2 Thesis objectives 4

1.3 Solution approach 4

1.4 Contribution 5

1.5 Outline 6

1.6 Publications 6

2 Conventions **9**

2.1 Coordinate convention & transformation 10

2.2 Euler convention 12

2.3 Quaternion convention 14

2.4 Denavit-Hartenberg convention 15

3 Quadrotor models **17**

3.1 Related work 18

3.2 Physical model with Euler angle description 19

3.3 Physical model with quaternion description 23

3.4 Hover model with yaw description 24

3.5 Hover model with direction vector attitude description 27

3.6 Discretization of grey box models 33

3.7 Identification . . . 34
3.7.1 AR.Drone 2.0 identification . . . 34
3.7.2 DJI M100 identification . . . 42
3.8 Conclusion . . . 46

4 Nonlinear model predictive control 49
4.1 State of the art in model predictive control . . . 52
4.2 Optimality . . . 54
4.2.1 Static optimization problems . . . 54
4.2.2 Optimality of dynamic optimal control problems . . . 55
4.3 Solvers . . . 59
4.3.1 Gauß-Newton sequential quadratic programming . . . 59
4.3.2 GRAMPC . . . 61
4.3.3 C/GMRES . . . 62
4.3.4 CMSCGMRES . . . 64
4.4 NMPC Benchmark . . . 69
4.5 Constraint handling . . . 75
4.5.1 Primal barrier method . . . 75
4.5.2 Auxiliary variable method . . . 76
4.5.3 Saturation function approach . . . 77
4.6 Stability discussion . . . 79
4.7 Conclusion . . . 80

5 UAV control 83
5.1 Related work . . . 83
5.2 Controllability . . . 86
5.3 NMPC pose tracking . . . 88
5.3.1 AR.Drone 2.0 NMPC pose tracking parametrization . . . 90
5.3.2 DJI M100 NMPC pose tracking . . . 100
5.4 Offset-free MPC . . . 104
5.5 Model predictive collision avoidance . . . 114
5.6 Task-based control using potential functions . . . 119
5.6.1 Formulating task with inequality constraints . . . 120
5.6.2 Inequality transformation to potential function . . . 121
5.6.3 Adding cost gradient . . . 121
5.6.4 Addressing differentiability of the potential function . . . 123
5.6.5 Coordinate relation . . . 123

5.6.6 Safety constraints 126
5.6.7 Experimental validation of task-based MPC 130
5.7 Conclusion 138

6 Aerial manipulation **141**
6.1 Related work in aerial manipulation control 143
6.2 Aerial manipulator kinematics 146
6.3 Aerial manipulator control 151
6.4 Experimental validation 155
6.4.1 End-effector pose tracking scenario 155
6.4.2 Bottle grasping scenario 159
6.5 Conclusion 162

7 Cooperative control **163**
7.1 Related work in cooperative control 164
7.2 OCP of distributed systems 167
7.3 Modularization 169
7.4 DENMPC framework implementation 173
7.4.1 Code generation 177
7.5 Validation of DENMPC 177
7.5.1 Modularization benchmark 178
7.5.2 Unidirectional sensor-constrained tracking 182
7.5.3 Online adaptability 188
7.5.4 Cooperative aerial manipulation 194
7.6 Conclusion 200

8 Conclusion and future work **203**
8.1 Future work 206

A Annex **207**
A.1 Hardware & software 207
A.2 Global OCP for distributed systems and optimality conditions for different constraint handling methods 208

Bibliography **213**

List of Tables

2.1 Mathematical convention 9

3.1 Identification parameters to match *AR.Drone 2.0* velocity channels . 36
3.2 Empirically chosen parameters to match *AR.Drone 2.0* position channels 39
3.3 Empirically chosen parameters to match *DJI* position channels 43

4.1 MPC: advantages and disadvantages 51
4.2 Newton-type algorithms with different approximations of the Hessian 60
4.3 *NMPC* solver parametrization for benchmark scenario, given in the solver inherent nomenclature 70

5.1 Task-based *UAV* control: Peak distance values for the set of 10 conducted experiments 138

6.1 Physical parameters of the *MUAV* setup in m 151
6.2 End-effector pose tracking scenario: control parametrization 155
6.3 *MUAV* grasping scenario: control parameters 160

7.1 Atomic functions to realize agents and couplings 172
7.2 Functionals and functions required for the code generation 177

List of Figures

1.1 Cooperative aerial manipulation scenario . . . 3

2.1 Quadrotor coordinate conventions . . . 12
2.2 Euler coordinate transformation chain . . . 13
2.3 Denavit-Hartenberg convention [BH09, p.159] (θ sign corrected) . . . 16

3.1 Quadrotor rotor convention . . . 22
3.2 First order plant step response . . . 25
3.3 Angle discontinuity visualization with constant velocity . . . 28
3.4 Angle discontinuity problem regarding the control trajectory of an *AR.Drone 2.0* . . . 29
3.5 Unit circle with direction vector projections . . . 30
3.6 Experimental validation of direction vector attitude description: trajectory of a real *AR.Drone 2.0* . . . 32
3.7 AR.Drone 2.0 photography . . . 35
3.8 Automized *AR.Drone 2.0* identification . . . 37
3.9 *AR.Drone 2.0* identification of forward channel . . . 40
3.10 *AR.Drone 2.0* identification of sideward channel . . . 40
3.11 *AR.Drone 2.0* identification of upward channel . . . 41
3.12 *AR.Drone 2.0* identification of heading channel . . . 41
3.13 DJI M100 photography . . . 42
3.14 DJI M100 identification of forward channel . . . 44
3.15 DJI M100 identification of sideward channel . . . 44
3.16 DJI M100 identification of upward channel . . . 45
3.17 DJI M100 identification of heading channel . . . 45

4.1 *MPC* receding horizon scheme . . . 50
4.2 *NMPC* pose tracking benchmark: *UAV* pose . . . 72
4.3 *NMPC* pose tracking benchmark: velocities . . . 73

4.4 *NMPC* pose tracking benchmark: controls, computation time, costs . 74
4.5 Sigmoid approximation of unit step 78

5.1 *AR.Drone 2.0 NMPC* pose tracking parametrization: 3D trajectories 92
5.2 *AR.Drone 2.0 NMPC* pose tracking parametrization: state trajectories 94
5.3 *AR.Drone 2.0 NMPC* pose tracking parametrization: tracking error, controls and computation time 95
5.4 *AR.Drone 2.0 NMPC* pose tracking: 3D trajectories 97
5.5 *AR.Drone 2.0 NMPC* pose tracking: channel trajectories 98
5.6 *AR.Drone 2.0 NMPC* pose tracking: tracking error, controls and computation time 99
5.7 *DJI M100 NMPC* pose tracking: 3D trajectory 101
5.8 *DJI M100 NMPC* pose tracking channel trajectories 102
5.9 *DJI M100 NMPC* pose tracking channel trajectories 103
5.10 Simulation of *NMPC* tracking of constantly moving target 104
5.11 Simulation of *NMPC* tracking of constantly moving target 106
5.12 Analytically determined system response for $t_s = 2\,\mathrm{s}$, $v_0 = 0.2\,\mathrm{m\,s^{-1}}$. 107
5.13 *TPC* control structure 107
5.14 *TPC* idea 108
5.15 *TPC* system responses for different parametrization for $t_s = 2\,\mathrm{s}$ and $v_0 = 0.2\,\mathrm{m\,s^{-1}}$ 109
5.16 Simulation of tracking a constantly moving target with *TPC* 110
5.17 Visualization of the experimental data of tracking of a target, moving in a square with a constant velocity 110
5.18 Real *AR.Drone 2.0* square tracking without *TPC* 112
5.19 Real *AR.Drone 2.0* square tracking with *TPC* 113
5.20 Sigmoid distance cost function L_{CA} with $\kappa_H = 1$ 114
5.21 *AR.Drone 2.0 CA*: the trajectory of the quadrotor is deviated by an obstacle, depicted as stand in the center point of the circle. As desired, the drone is avoiding the circular keep out area with radius $r = 1\,\mathrm{m}$. 115
5.22 *AR.Drone 2.0 CA*: pose trajectory 117
5.23 *AR.Drone 2.0 CA*: trajectories 118
5.24 Use case scenario: tracking with cone-shaped sensor perception of a quadrotor 119
5.25 Unit step penalty function values of cone constraint 122
5.26 Unit step penalty function of cone constraint in the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane . . . 122

5.27 Extended soft cone constraint transformed with sigmoid 123
5.28 Tracking of global position with soft cone constraint 126
5.29 Potential function for collision avoidance 128
5.30 Potential function for cohesion 129
5.31 Task-based *UAV* control: target tracking while obstacle is avoided . . 131
5.32 Task-based *UAV* control: *UAV* ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-trajectory with pose markers every $\Delta t = 4.2\,\mathrm{s}$ showing circular evasion maneuver pattern 133
5.33 Task-based *UAV* control: *UAV* pose trajectory with steps in the ${}^{\mathcal{V}}\Psi$-trajectory due to the limited yaw angle interval 134
5.34 Task-based *UAV* control: manually introduced obstacle position is showing circular pattern in order to follow *UAV* on its evasion trajectory. The steps are caused by the obstacle entering the field of perception of the motion caption system. 134
5.35 Task-based *UAV* control: *UAV* distance to obstacle d_o stays above the defined minimum distance of $d_{c,min} = 1.0\,\mathrm{m}$ 136
5.36 Task-based *UAV* control: *UAV* distance to target d stays within the defined minimum distance of $d_{min} = 0.7\,\mathrm{m}$ and maximum distance of $d_{max} = 2.0\,\mathrm{m}$ 136
5.37 Task-based *UAV* control: *UAV* absolute tracking angle α_t stays within the defined maximum of $\alpha_{tmax} = 0.5\,\mathrm{rad}$ of the sensor cone constraint 137
5.38 Task-based *UAV* control: *UAV* actuation 137

6.1 Aerial Manipulator consisting of *DJI M100* and two-joint robotic arm (the stereo camera on top is not part of the experimental setup) . . . 142
6.2 Manipulator dimension and coordinate frames for front tracking . . . 147
6.3 *MUAV* with grasped object (orange cylinder): coordinate frames . . . 147
6.4 *MUAV* coordinate frame transformation chain 148
6.5 End-effector pose tracking scenario: 3D-path with target (red circles), end-effector (blue), *UAV* center (orange) 156
6.6 End-effector pose tracking scenario: states with target (red), end-effector (blue), *UAV* center (orange) 157
6.7 End-effector pose tracking scenario: control influence 158
6.8 *MUAV* grasping scenario: step chain with transition conditions . . . 159
6.9 *MUAV* grasping scenario: scenario sequence using *DJI M100* with two-joint robotic arm 160
6.10 *MUAV* grasping scenario: trajectories from controller perspective . . 161

7.1 Scheme of *UAV* cooperation 167
7.2 Schematic DENMPC class structure 173
7.3 Schematic DENMPC memory and concatenation structure 174
7.4 Comparison with (left) and without (right) modularization: UAV_1 and UAV_2 keep distance $d_{des} = 1$ while UAV_1 is tracking target 180
7.5 Comparison with and without modularization UAV_1 and UAV_2 keep distance $d_{des} = 1$ and UAV_1 is tracking target 181
7.6 Cooperative sensor-constrained tracking scenario: Visual tracking of quadrotor from camera equipped quadrotor 182
7.7 Cooperative sensor-constrained tracking scenario: setup 185
7.8 Cooperative sensor-constrained tracking scenario: position 186
7.9 Cooperative sensor-constrained tracking scenario: data 187
7.10 Runtime formation change scenario: Video footage: t_1: 2 agents (left) vs. t_2:3 agents (right) 188
7.11 Runtime formation change scenario: 3D position plot with formation visualization 191
7.12 Runtime formation change scenario: *UAV* trajectories 192
7.13 Runtime formation change scenario: distance, tracking and computation time 193
7.14 Cooperative aerial manipulation video footage 194
7.15 Trigonometric scheme for tracking function 195
7.16 Cooperative aerial manipulation: *MUAV* end effector pose tracking . 196
7.17 Cooperative aerial manipulation: *DJI M100* trajectories 197
7.18 Sensing *UAV* tracking 3d plot showing the virtual *AR.Drone 2.0* reference ($p_{ee,des}$ and its trajectory p_{ee}) 198
7.19 Cooperative aerial manipulation: *AR.Drone 2.0* trajectories and controller computation time 199

8.1 Aerial manipulation scenario: Autonomous grasping and pouring of a bottle using a task step chain in combination with *DENMPC* 206

List of Abbreviations

Symbol	**Description**
SS	State-Space
MPC	Model Predictive Control
KKT	Karush-Kuhn-Tucker Optimality conditions
SQP	Sequential Quadratic Programming
QP	Quadratic Programming
OCP	Optimal Control Problem
NMPC	Receeding horizon Nonlinear Model Predictive Control
ROS	Robot Operating System
PID	Controller penalizing the proportional integral and derivative of the control error
PI	Controller penalizing the proportional integral of the control error
UAV	Unmanned Aerial Vehicle
MUAV	Manipulating Unmanned Aerial Vehicle
ISS	Input-to-State-Stabiliy
GRAMPC	Graichen MPC is a indirect gradient model predictive control algorithm
ACADO	Automatic Control And Dynamic Optimization toolkit
GMRES	Generalized Minimal RESidual method
FDGMRES	Finite Difference Generalized Minimal RESidual method
C/GMRES	Continuation Generalized Minimal RESidual method
MSC/GMRES	Multiple Shooting Continuation Generalized Minimal RESidual method
CMSC/GMRES	Condensed Multiple Shooting Continuation Generalized Minimal RESidual method
C	Programming language "C"
C++	Programming language "C++"
V-REP	Virtual Robot Experimentation Platform
LQR	Linear Quadratic Regulator

Symbol	Description
CA	Collision Avoidance
HEUN	Heun integrator
rhs	Right hand side
lhs	Left hand side
PT1	First order plant
PT2	Second order plant
ODE	Ordinary Differential Equation
AR.Drone 2.0	Augmented Reality Drone 2.0 by Parrot
DJI M100	DJI Matrice 100
PMP	Pontryagin's Maximum Principle
DJI	D-Jing Innovations Science and Technology Co., Ltd
DOF	Degree Of Freedom
MATLAB	MATrix LABoratory
Mathematica	Wolfram—Alpha Mathematica
DH	Denavit-Hartenberg
TPC	Target Position Control
SSREGEST	Estimate state-space model by reduction of regularized ARX model
SSEST	Estimate state-space model
N4SID	State-space estimation using subspace method
TC	Test case
ARCAS	Aerial Robotics Cooperative Assembly System
FK	Forward Kinematics
IK	Inverse Kinematics
DENMPC	Distributed System Event-based Nonlinear Model Predictive Control Framework
p.s.d.	Positive semi-definite
OPTITRACK	Optitrack motion capture system
BFGS	Broyden-Fletcher-Goldfarb-Shannon approximation
HMDV	Hover Model with Direction Vector attitude description

Symbol	**Description**
$RHMDV$	Reduced Hover Model with Direction Vector attitude description
$HMDV$	Hover Model with Yaw attitude description
$RHMY$	Reduced Hover Model with Yaw attitude description
$NLopt$	Open Source Library for NonLinear optimization
$ERR20$	European Roadmap for Robotics in 2020
$s.\ t.$	subject to

Glossary

Symbol	**Description**
ϖ	Concatenated state and adjoint state variables
φ	Variated of concatenated state and adjoint state variables
σ	Vector
v	Velocity
ν	Slack variable
p	Point
x	Coordinate x
y	Coordinate y
z	Coordinate z
w	Coordinate w
Φ	Roll
Θ	Pitch
Ψ	Yaw
o_x	Direction vector projection on x
o_y	Direction vector projection on y
o	Direction vector
ϱ	Angular velocity around x-axis
ϑ	Angular velocity around y-axis
ω	Angular velocity around z-axis
t_s	Time constant
k_p	Proportional gain
k_i	Integral gain
k_d	Derivative gain
ζ	Integration error
e	Error
n	Dimension/amount
$\varkappa$	Parameter
y	Output
x	State

Symbol	**Description**
ς	Disturbance
λ	Adjoint state
λ_{in}	Inequality constraint Lagrange multiplier
λ_{eq}	Equality constraint Lagrange multiplier
$\boldsymbol{w}$	Concatenated optimization variables
x_{des}	State
u	Control
u^*	Optimal control
u_{des}	Desired control
L	Integral/stage costs
V	Terminal/final costs
J	Cost function
$\mathcal{L}$	Lagrangian
$\mathcal{H}$	Hamiltonian
f	System function
c	Constraint function
c_{in}	Inequality constraint function
c_{eq}	Equality constraint function
τ	Horizon time variable
t	Global time variable
s	Laplace operator
k	Iterator/discrete time index
j	Solver iteration index
l	Horizon iteration index
T	Horizon length
N	Horizon samples
Δt	Time step
$\Delta\tau$	Horizon time step
α	Step width
υ	Horizon expansion factor

Symbol	**Description**
ϵ	Tolerance
h	Forward differentiation step
i_{max}	Maximum number of iterations
ξ	Convergence factor
χ	Concatenation matrix of Li-Brackets
$\mathfrak{f}$	Concatenated function
$\mathfrak{g}$	Concatenated function
g	Gravity constant
m	Mass
ι	Inertia
d_{rot}	Rotor distance
d_s	Sensor distance
c_Ψ	Yaw constant
c_Γ	Thrust constant
q	State penalty
r	Control penalty
d	Distance
R	Rotation matrix
T	Transformation matrix
ρ	Quaternion
Γ	Force
Λ	Torque
ω_{rot}	Rotor speed
$\mathcal{A}$	Frame $\mathcal{A}$
$\mathcal{B}$	Frame $\mathcal{B}$
$\mathcal{V}$	Vehicle frame
$\mathcal{V}1$	Vehicle 1 frame
$\mathcal{V}2$	Vehicle 2 frame
$\mathcal{G}$	Global frame
$\mathcal{B}$	Body frame

Symbol	**Description**
$\mathcal{M}$	Base frame
$\mathcal{J}$	Joint frame
$\mathcal{J}1$	Joint 1 frame
$\mathcal{J}2$	Joint 2 frame
$\mathcal{E}_F$	End effector front frame
$\tilde{\mathcal{E}}_F$	Auxiliary end effector front frame
$\mathcal{E}$	End effector frame
$\mathcal{E}_E$	End effector front bottom frame
$\mathcal{P}$	End effector center auxiliary frame
$\mathcal{E}_C$	End effector center frame
$\mathcal{A}$	Set of agents
$\mathcal{N}$	Set of neighbors
t_c	Computation time
$\mathfrak{L}$	Lie bracket
$\mathcal{S}$	Sensor frame
α_{FoV}	Sensor angle of view
β_{FoV}	Sensor inclination angle
α_t	Sensor tracking angle
θ	Joint angle
a	Joint length
d	Joint height
α	Joint twist
ee	End effector
ϵ	Unit step function
κ_G	Gradient parameter
κ_A	Approximation gradient parameter
κ_H	Cost parameter
sig	Sigmoid
$\mathcal{V}$	Set of agents/vertices
$\mathcal{E}$	Set of couplings/edges

Symbol	**Description**
ν	Agent/vertex
ϵ	Coupling/edge
i	Agent iterator
j	Neighbor iterator
n_ν	Number of agents
n_η	Number of neighbors
n_ϵ	Number of edges
l	Constraint iterator
$\mathbb{R}$	Real numbers
$\mathcal{C}^2$	Class of twice differentiable functions/functionals

Chapter 1

Introduction

The significance of Unmanned Aerial Vehicles (*UAV*s) has been increasing over the last decades. According to [Meo17], 2.2 million consumer drones have been sold for $4.5 billion in 2016. Already until 2021, the global annual revenue in drones is estimated to grow to a total of $12 billion. Originally, *UAV* applications have been limited to governmental use such as military purposes, fire monitoring as well as search and rescue missions. However, the advances in energy storage and computation technology have led to a downsize of *UAV*s and has made them attractive for the consumer market. Their main civil selling points today are their sensing capability and extensive operational space. Today, the related use reaches from leisure entertainment to surveillance and videography purposes. In the agricultural sector, the reduction of chemical plant treatment is demanding alternative solutions for large-scale fostering and monitoring of plants [Pat16, Mos15]. In parallel, the industry is developing modular smart factories which require novel solutions in logistics, inspection and manufacturing [Jos17]. These demands open new fields of applications for *UAV*s regarding transportation and manipulation purposes.

To address these demands, the "European Roadmap for Robotics in 2020" (*ERR20*) [SPA16] is stating the required key abilities of aerial robots in future and their technological readiness (in brackets):

- Configurability: adapting to different environmental conditions and state (3)
- Interaction ability: Interaction cognitively and physically with other robots (5)
- Motion ability: Ability to move indoor and outdoor avoiding obstacles and exerting forces (5)
- Manipulation ability: Ability to perform aerial manipulation tasks (2-3)
- Decisional autonomy: On-board reactivity and planning including multiple-robot systems autonomy (8)

Due to the low technological readiness of aerial manipulation and robot interaction technology, the applications of small *UAV*s are currently still limited to sensing and transportation scenarios.

The development of this technology is therefore a key priority towards industrial applications. The governmental support of this conclusion is stated by dedicated research projects. In this context, a significant project is the European Aerial Robotics Cooperative Assembly System project (*ARCAS*). Examples of the fast-growing field of aerial manipulation applications are contact inspection of bridges [JCBHO15], canopy sampling of the environment [KSX15], opening of valves [KOO14] and positioning of assembly parts in factories [MLS15]. To fulfill such tasks, a manipulator is attached to the *UAV* which results in a Manipulating Unmanned Aerial Vehicle (*MUAV*). In contrast to traditional fix wing *UAV*s, small multi-rotor *UAV*s are capable of hovering at one position. This makes them particularly suitable for deployment in cluttered environments and for manipulation tasks.

In order to execute complex tasks (e.g. in industrial environments) efficiently and flexibly, it is advantageous to deploy multiple specialized robots with complimentary abilities [ATDG10]. Hence, a key technology target of the *ERR20* [SPA16] is the "distributed architectures for multiple aerial robots and heterogeneous multi-robot systems with dependability and reconfiguration properties". Such systems allow the execution of versatile tasks in a highly efficient way and can furthermore provide safety by redundancy. The versatility thereby allows the adaptation of the system to different tasks, utilized robots and environments. The challenging nature of such dynamically changing cooperative control tasks in combination with the fast dynamics of *UAV*s and complexity of *MUAV* systems has motivated this work.

In order to contribute to the technological development of aerial robotics, this thesis is dedicated to advanced control design for commercially available small multi-rotor *UAV*s. In particular, this work is focusing on model predictive control strategies for cooperative aerial manipulation.

1.1 Problem statement

The applications of cooperative aerial manipulation and corresponding control approaches are manifold. The necessary elements to control such scenarios can be explained using the example scenario given in Figure 1.1. In this scenario, two different types of robots are deployed. The *UAV* shown on the left is equipped with computer vision and a localization system. The *MUAV* on the right is only equipped with a robotic arm to optimize payload and energy consumption. In order to manipulate its environment (e.g. grasping the bottle), the $MUAV$ has to be precisely localized in its environment. For this purpose, the $MUAV$ can cooperate with the UAV by requesting localization and optical sensor information. As the perception space of the optical sensor is limited, the UAV has to be controlled in relation with the $MUAV$ or/and vice versa to continuously provide sensor information.

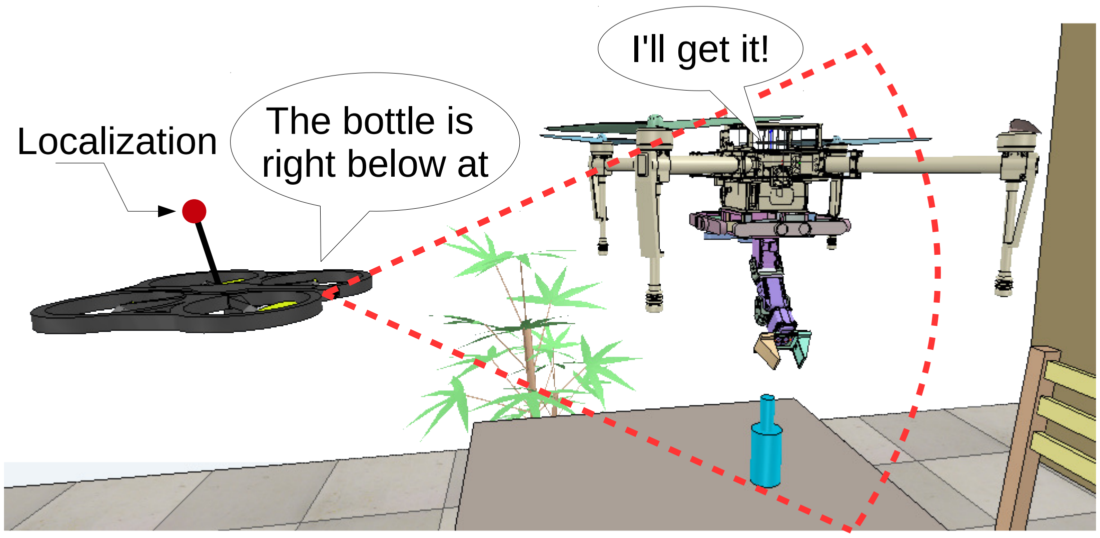


Figure 1.1: Cooperative aerial manipulation scenario

From a control engineering perspective, there are three major challenges of the illustrated scenario. The first challenge is the complex fast nonlinear dynamics of $MUAV$s and UAVs. This complexity and speed conflicts with their limited computational power onboard. Hence, computationally efficient algorithms are required to achieve real-time control. This is particularly challenging for commercial UAVs, as the information about internal control and physical UAV properties is generally very limited.

The second challenge is to provide a generic controller which is adaptable to different control tasks (e.g. searching for the bottle vs. tracking the $MUAV$), robots (e.g. UAV, $MUAV$, different sensor setups) and environmental constraints (e.g. obstacles, such as furniture).

The third challenge is to control the cooperation of the UAVs within the scenario. For safety and efficiency, it is advantageous to be able to dynamically exchange robots of different types and the overall scenario objectives at runtime.

The objective of this work is to provide applied control solutions for these challenges, evaluated in real-world experiments. These have been exclusively conducted with commercially available *UAV*s, while omitting proprietary solutions. The usage of commercial *UAV*s facilitates the reproducibility of results and promotes science as well as application development by reducing the development effort and costs.

This thesis focuses on control engineering. To limit the scope of this work localization, communication, respectively perception and estimation are not subject of this thesis. However, all commercial *UAV*s do have internal attitude controllers. A communication failure would thus only lead to missing localization data, but not to a complete system failure.

1.2 Thesis objectives

According to the problem statement in §1.1, the objective of this thesis is the development of a control strategy for mobile robots. This control shall allow

- exploiting the robot dynamics for time and energy efficient operation
- the consideration of environmental constraints
- runtime reconfiguration and adaptation for different situations and tasks
- the robot to interact with its environment e.g. by means of a manipulator
- the coordinated real-time control of heterogeneous teams of robots

These objectives are coherent with the key technology targets given in *ERR20* for aerial robots in order to increase the capability of robots towards industrial applications. For this reason, a further objective of this thesis is to validate the applicability of the developed control strategy in real aerial manipulation and cooperative control scenarios under the use of commercially available *UAV*s.

1.3 Solution approach

In this thesis Model Predictive Control (*MPC*) is used to handle complex control scenarios. *MPC* allows defining control objectives by means of an optimization problem. This so-called Optimal Control Problem (*OCP*) is minimizing a given objective function subject to constraints. However, the real-time application of *MPC* for fast systems is challenging, due to the computational complexity of solving the underlying *OCP*. This is further exacerbated if additional algorithmic mechanisms are required, for example to allow a runtime modification of the *MPC*. Hence, the efficiency of the applied solver is limiting the velocity and complexity of the controlled scenarios. The advantage of a single central *MPC* unit is hereby a simple implementation of safety features. Furthermore, the global *OCP* solution can be computed without considering additional consensus techniques. For these reasons, this thesis is focusing on the development of a centralized *MPC* solution. However, the presented approaches can be adapted in a distributed manner for large-scale applications.

1.4 Contribution

The first contribution of this work is a model which describes the motion of *UAV*s with internal attitude controller. The model is designed to be computationally efficient and to avoid singularities in the orientation description. The model is suited to approximate and predict the *UAV*'s pose and velocity. The corresponding model parameters are identified for real *AR.Drone 2.0* and *DJI M100* quadrotors.

The second contribution is a nonlinear *MPC* (*NMPC*) for the pose of these *UAV*s and the corresponding control parameterization. In addition, a workflow is contributed that allows to consider constraints e.g. position and sensor perception space constraints within the real-time *MPC*.

The third contribution of this work is the development of a controller for end effector pose tracking of a commercial *MUAV*. This novel approach is based on a separate tracking of the end effector position and orientation. As a consequence, the compact direction vector *UAV* model can be used which results in low computation times. The effectiveness of the proposed approach is demonstrated in a real-world aerial manipulation scenario.

The final contribution of this work is the implementation of the Distributed System Event-Based Nonlinear Model Predictive Control (*DENMPC*) framework. This is an object-oriented real-time nonlinear *MPC* framework for small-scale multi-robot systems. *DENMPC* features a computationally efficient *MPC* modularization which allows the runtime modification of robots, control objectives and scenario constraints. To facilitate implementation and prototyping, *DENMPC* is using the communication infrastructure of the Robot Operating System (*ROS* [QCG$^+$09]). Subsequently, the experimental validation of the developed framework is conducted in real multi-*UAV* scenarios.

1.5 Outline

This thesis is structured according to the thesis objectives given in §1.2. Each chapter is thereby addressing one of the presented objectives. As the related literature for the individual chapters is distinct, the related work is given separately at the beginning of each particular chapter. The results for each topic are summarized in an individual conclusion section at the end of the related chapter. The thesis is beginning with a brief introduction in pose description techniques, coordinate conventions, coordinate transformation and the related nomenclature in §2. In §3, different approaches to model the *UAV* behavior are given. In this context, a compact direction vector model is introduced and identified for real *UAV* systems. Fundamental *MPC* knowledge, as well as an implementational description of the utilized *MPC* methods are provided in §4. This includes different solution strategies for the underlying *OCP* and various constraint handling techniques. §5 is demonstrating the *MPC* of real *UAV* systems, using the previously presented *UAV* models. In addition, a workflow is presented to facilitate the formulation of complex control objectives. The *MUAV* kinematic modeling and control is discussed in §6. Finally, the control of multiple *UAV*/*MUAV* systems is discussed in §7. This chapter presents the development of the *DENMPC* control framework and its application in cooperative control scenarios. To conclude this thesis, §8 is summarizing the presented work, discussing the results and providing a future perspective.

1.6 Publications

In the context of this thesis, the open source (GPL3 licensed) software

- Jan Dentler. ***DENMPC*****: An event-based real-time nonlinear model predictive control framework**. *Github, 2017* [Den16].

has been developed (see §7). This thesis contains material which has been also published in journal and conference proceedings and is available under the following sources:

- Jan Dentler, Somasundar Kannan, Souad Bezzoucha, Miguel Angel Olivares-Mendez and Holger Voos. **Model predictive cooperative localization control of multiple UAVs using potential function sensor constraints**. *Springer Autonomous Robots Journal 2018 [DKB+18]* (see §3,§5,§7).

- Jan Dentler, Somasundar Kannan, Miguel Angel Olivares-Mendez and Holger Voos. **Implementation and Validation of an Event-Based Real-Time Nonlinear Model Predictive Control Framework with ROS Interface for Single and Multi-robot Systems**. *Proceedings of the IEEE Conference on Control Technology and Applications 2017* [DKMV17] (see §3,§5,§7).

- Jan Dentler, Somasundar Kannan, Miguel Angel Olivares-Mendez and Holger Voos. **A modularization approach for nonlinear model predictive control of distributed fast systems**. *Proceedings of the IEEE 24th Mediterranean Conference on Control and Automation 2016* [DKMV16a] (see §7).

- Jan Dentler, Somasundar Kannan, Miguel Angel Olivares-Mendez and Holger Voos. **A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors**. *Proceedings of the IEEE Multiconference on Systems and Control 2016* [DKMV16b] (see §3,§5).

- Jan Dentler, Somasundar Kannan, Miguel Angel Olivares-Mendez and Holger Voos. **A tracking error control approach for model predictive position control of a quadrotor with time varying reference**. *Proceedings of the IEEE International Conference on Robotics and Biomimetics 2016* [DKMV16c] (see §5).

Within the context of this thesis, following work of students has been conducted:

- Patrick Kremer, Jan Dentler, Somasundar Kannan and Holger Voos. **Cooperative Localization of Unmanned Aerial Vehicles in ROS - The Atlas Node**. *Proceedings of the IEEE Conference on Industrial Informatics 2017* [PDKV17].

- Patrick Hoffmann. **Development of a controller for a lightweight manipulator**. *University of Luxembourg, Bachelor thesis, 2016* [Hof16].

In cooperation with the Parallel Computing and Optimization Group (PCOG) at the University of Luxembourg, a study on *UAV* swarm trajectory planning using chaotic dynamics and their control with *MPC* has been conducted. This study is not within the scope of this thesis, but has been published as follows:

- Martin Rosalie, Jan Dentler, Gregoire Danoy, Pascal Bouvry, Somasundar Kannan, Miguel Angel Olivares-Mendez and Holger Voos. **Collision avoidance effects on the mobility of a UAV swarm using Chaotic Ant Colony**

with Model Predictive Control. *Journal of Intelligent & Robotic Systems 2018* [DMD+18].

- Martin Rosalie, Jan Dentler, Gregoire Danoy, Pascal Bouvry, Somasundar Kannan, Miguel Angel Olivares-Mendez and Holger Voos. **Area exploration with a swarm of UAVs combining deterministic Chaotic Ant Colony Mobility with position MPC**. *Proceedings of the IEEE Conference on Unmanned Aircraft Systems 2017* [MDD+17].

Chapter 2

Conventions

This section is discussing the nomenclature and conventions used in this thesis. The applied mathematical nomenclature is given in Table 2.1.

Expression	Notion	e.g.
Scalars:	small letters and capital greek letters	p, Γ
Functionals:	capital latin letters	P
Vectors & vector-valued functions	bold small letters and bold capital greek letters	$\boldsymbol{p}, \boldsymbol{\Gamma}$
3D-Vectors defined in the Cartesian space	indicated by overlying arrow	$\overrightarrow{\boldsymbol{p}}$
Matrices	capital bold latin letters	$\mathbf{P}$
Coordinate systems & sets	calligraphic capital latin letters	$\mathcal{P}$
MPC related variables which are determined over the prediction horizon	underline	$\underline{\boldsymbol{p}}$
Transformation matrix to describe $\mathcal{A}$ in reference to $\mathcal{B}$		${}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{A}}$

Table 2.1: Mathematical convention

In the context of this thesis, the standard state-space (*SS*) control terminology is used. The plant state $\boldsymbol{x} \in \mathbb{R}^{n_{\boldsymbol{x}}}$ is therefore influenced by its input. Without disturbance this input is equal to the control output $\boldsymbol{u} \in \mathbb{R}^{n_{\boldsymbol{u}}}$. The idea of control is to use $\boldsymbol{u}$ to steer the plant towards a desired state $\boldsymbol{x}_{des} \in \mathbb{R}^{n_{\boldsymbol{x}}}$. A common way to describe the dynamics of the plant is thereby a first-order Ordinary Differential

Equation (ODE) which is called system function

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \quad \boldsymbol{f}: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}^{n_x} \tag{2.1}$$

2.1 Coordinate convention & transformation

A common tool to describe the position and orientation in $3D$ space are Cartesian coordinate systems. The pose of an object in reference to a global coordinate frame $\mathcal{G}$ is defined by its position and orientation. The position is thereby given as translation of the attached body frame's $\mathcal{B}$ origin in respect to the global coordinate systems $\mathcal{G}$

$$^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{B}} = [^{\mathcal{G}}x_{\mathcal{B}}, {}^{\mathcal{G}}y_{\mathcal{B}}, {}^{\mathcal{G}}z_{\mathcal{B}}]^{\mathrm{T}} \in \mathbb{R}^3. \tag{2.2}$$

The lower right index indicates the original coordinate frame and the higher left index the reference coordinate frame in which the original coordinates shall be expressed. If no indices or contextual information are explicitely given, the translation/rotation between body frame and global frame is described

$$^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{B}} \equiv \overrightarrow{\boldsymbol{p}}. \tag{2.3}$$

The orientation can be described as rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$, Euler angles or quaternions ρ. The transformation of coordinates $^{\mathcal{A}}\overrightarrow{\boldsymbol{p}}$ which are defined in $\mathcal{A}$, into $\mathcal{B}$ ($^{\mathcal{B}}\overrightarrow{\boldsymbol{p}}$) can be described with the help of homogeneous coordinates. For $3D$-space the homogeneous coordinates are formed by extending the point vectors $\overrightarrow{\boldsymbol{p}} \in \mathbb{R}^3$ by one dimension. This extension allows the usage of homogeneous transformation matrices $^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B}} \in \mathbb{R}^{4\times4}$ ([WS16])

$$\begin{bmatrix} ^{\mathcal{A}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} = {}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B}} \begin{bmatrix} ^{\mathcal{B}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} = \begin{bmatrix} ^{\mathcal{A}}\boldsymbol{R}_{\mathcal{B}} & ^{\mathcal{A}}\overrightarrow{\boldsymbol{p}}_{\mathcal{B}} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} ^{\mathcal{B}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} \tag{2.4}$$

Due to the orthogonality of $\boldsymbol{R}$, $(\boldsymbol{R})^{-1} = \boldsymbol{R}^{\mathrm{T}}$ holds and the inversion of $\boldsymbol{T}$ yields ([WS16])

$$\begin{bmatrix} ^{\mathcal{B}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} = {}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B}}{}^{-1} \begin{bmatrix} ^{\mathcal{A}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} = \begin{bmatrix} ^{\mathcal{A}}\boldsymbol{R}_{\mathcal{B}}^{\mathrm{T}} & -{}^{\mathcal{A}}\boldsymbol{R}_{\mathcal{B}}^{\mathrm{T}}\,{}^{\mathcal{A}}\overrightarrow{\boldsymbol{p}}_{\mathcal{B}} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} ^{\mathcal{B}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} \tag{2.5}$$

The advantage of such homogeneous transformation matrices is, that a chain of multiple translations and rotations can be expressed in a single transformation matrix

[Bea08]. The representative transformation matrix results from multiplying basic homogeneous transformation matrices. Examples of such basic transformations include the translation from $\mathcal{A}$ to $\mathcal{B}$ (2.6) by ${}^{\mathcal{A}}\vec{\varkappa}$, counterclockwise rotation around axis $z_{\mathcal{A}}$ (2.7) by Ψ, ${}^{\mathcal{A}}y$ (2.8) by Θ and ${}^{\mathcal{A}}x$ (2.9) by Φ:

$$
{}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},tl}\left({}^{\mathcal{A}}\vec{\boldsymbol{p}}\right) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & {}^{\mathcal{A}}x \\ 0 & 1 & 0 & {}^{\mathcal{A}}y \\ 0 & 0 & 1 & {}^{\mathcal{A}}z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.6}
$$

$$
{}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},rotz}\left({}^{\mathcal{A}}\Psi\right) = \left[\begin{array}{ccc|c} \cos({}^{\mathcal{A}}\Psi) & -\sin({}^{\mathcal{A}}\Psi) & 0 & 0 \\ \sin({}^{\mathcal{A}}\Psi) & \cos({}^{\mathcal{A}}\Psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} & & & 0 \\ & {}^{\mathcal{A}}\boldsymbol{R}_{\mathcal{B}rotz}\left({}^{\mathcal{A}}\Psi\right) & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.7}
$$

$$
{}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},roty}\left({}^{\mathcal{A}}\Theta\right) = \left[\begin{array}{ccc|c} \cos({}^{\mathcal{A}}\Theta) & 0 & \sin({}^{\mathcal{A}}\Theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin({}^{\mathcal{A}}\Theta) & 0 & \cos({}^{\mathcal{A}}\Theta) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} & & & 0 \\ & {}^{\mathcal{A}}\boldsymbol{R}_{\mathcal{B}roty}\left({}^{\mathcal{A}}\Theta\right) & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.8}
$$

$$
{}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},rotx}\left({}^{\mathcal{A}}\Phi\right) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos({}^{\mathcal{A}}\Phi) & -\sin({}^{\mathcal{A}}\Phi) & 0 \\ 0 & \sin({}^{\mathcal{A}}\Phi) & \cos({}^{\mathcal{A}}\Phi) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} & & & 0 \\ & {}^{\mathcal{A}}\boldsymbol{R}_{\mathcal{B}rotx}\left({}^{\mathcal{A}}\Phi\right) & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.9}
$$

To indicate the sign of the variables $\vec{\varkappa}$, Ψ, Θ, Φ, the index gives the base coordinate system in which the translation vector and angles are defined. The inversion of these elementary matrices is straightforward:

$$
{}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{A},tl}\left({}^{\mathcal{A}}\vec{\boldsymbol{p}}\right) = \left({}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},tl}\left({}^{\mathcal{A}}\vec{\boldsymbol{p}}\right)\right)^{-1} = {}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},tl}\left(-{}^{\mathcal{A}}\vec{\boldsymbol{p}}\right) \tag{2.10}
$$

$$
{}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{A},rotz}\left({}^{\mathcal{A}}\Psi\right) = \left({}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},rotz}\left({}^{\mathcal{A}}\Psi\right)\right)^{-1} = \left({}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},rotz}\left({}^{\mathcal{A}}\Psi\right)\right)^{\mathrm{T}} = {}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},rotz}\left(-{}^{\mathcal{A}}\Psi\right) \tag{2.11}
$$

$$
{}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{A},roty}\left({}^{\mathcal{A}}\Theta\right) = \left({}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},roty}\left({}^{\mathcal{A}}\Theta\right)\right)^{-1} = \left({}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},roty}\left({}^{\mathcal{A}}\Theta\right)\right)^{\mathrm{T}} = \boldsymbol{T}^{\mathcal{B}}_{\mathcal{A},roty}\left(-{}^{\mathcal{A}}\Theta\right) \tag{2.12}
$$

$$
{}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{A},rotx}\left({}^{\mathcal{A}}\Phi\right) = \left({}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},rotx}\left({}^{\mathcal{A}}\Phi\right)\right)^{-1} = \left({}^{\mathcal{A}}\boldsymbol{T}_{\mathcal{B},rotx}\left({}^{\mathcal{A}}\Phi\right)\right)^{\mathrm{T}} = \boldsymbol{T}^{\mathcal{B}}_{\mathcal{A},rotx}\left(-{}^{\mathcal{A}}\Phi\right) \tag{2.13}
$$

For the rotation matrices, the inversion is equal to clockwise rotations around the corresponding axis. The orthogonality of the rotation matrices allows thereby the inversion by a simple transposition. These basic transformation matrices can be combined to describe an object orientation according to the Euler convention.

2.2 Euler convention

Euler angles are a traditional approach to describe an orientation in $3D$-space (the $\mathcal{SO}(3)$ rotation group). In the context of this work, the Euler angles follow the $ZY'X''$-convention with yaw Ψ, pitch Θ, roll Φ angle. Figure 2.1 illustrates this description with the example of the UAV coordinate convention used within this thesis.

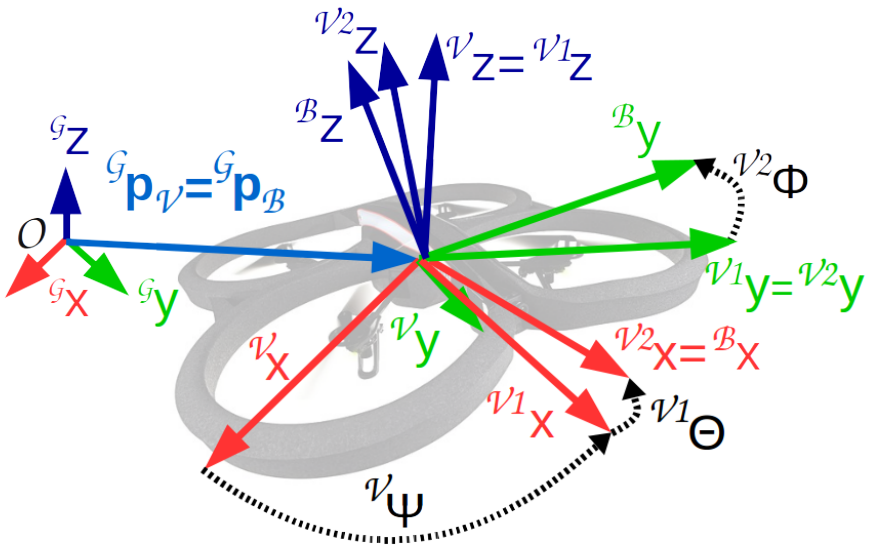


Figure 2.1: Quadrotor coordinate conventions

Figure 2.1 shows $\mathcal{B}$ coordinate axes originating from the UAV center pointing to its front, left and top. To describe the UAV orientation in reference to the global coordinate frame $\mathcal{G}$, first, the vehicle frame $\mathcal{V}$ is defined. $\mathcal{V}$ has the same orientation as $\mathcal{G}$, but is translated to the center of the UAV. From there on, the idea of Euler angles is to consecutively rotate around the coordinate system axes in order to align the resulting coordinate system with the body coordinates $\mathcal{B}$. The first rotation is rotating the vehicle frame $\mathcal{V}$ by the yaw angle Ψ around the $^{\mathcal{V}}z$-axis. The result is the new “vehicle 1 frame” $\mathcal{V}1$. Within $\mathcal{V}1$ $^{\mathcal{V}1}x$ and $^{\mathcal{V}1}y$ are lying in the $^{\mathcal{G}}x^{\mathcal{G}}y$-plane. If the quadrotor is not tilted, $^{\mathcal{V}1}x$ is accordingly pointing in forward and $^{\mathcal{V}1}y$ in left direction. $^{\mathcal{V}1}z$ and $^{\mathcal{G}}z$ are identically pointing upwards. This frame definition allows the description of the UAV heading in its static equilibrium and is therefore extensively used within this work. The vehicle 2 frame $\mathcal{V}2$ is defined by the rotation of $\mathcal{V}1$ around $^{\mathcal{V}1}y$ by a pitch angle Θ. Finally, the body frame $\mathcal{B}$ is reached by rotating $\mathcal{V}2$ around $^{\mathcal{V}2}x$ by a roll angle Φ. The body frame axes $^{\mathcal{B}}x$, $^{\mathcal{B}}y$, $^{\mathcal{B}}z$ are accordingly aligned with the UAVs forward, sideward and upward movement. This definition of the Euler angles is used within this thesis. In order to improve the readability, the related coordinate frame indeces can be ommitted in the following. The whole coordinate transformation chain with the transformation matrices (2.7)-(2.13) is shown in Figure 2.2.

$$\begin{tikzcd}[column sep=huge]
\mathcal{B} \arrow[r, bend left, "{}^{\mathcal{V}2}\boldsymbol{T}_{\mathcal{B},rotx}\left({}^{\mathcal{V}2}\Phi\right)"] \arrow[r, phantom, "\text{roll}"] & \mathcal{V}2 \arrow[l, bend left, "{}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{V}2,rotx}\left({}^{\mathcal{V}2}\Phi\right)"] \arrow[r, bend left, "{}^{\mathcal{V}1}\boldsymbol{T}_{\mathcal{V}2,roty}\left({}^{\mathcal{V}1}\Theta\right)"] \arrow[r, phantom, "\text{pitch}"] & \mathcal{V}1 \arrow[l, bend left, "{}^{\mathcal{V}2}\boldsymbol{T}_{\mathcal{V}1,roty}\left({}^{\mathcal{V}1}\Theta\right)"] \arrow[r, bend left, "{}^{\mathcal{V}}\boldsymbol{T}_{\mathcal{V}1,rotz}\left({}^{\mathcal{V}}\Psi\right)"] \arrow[r, phantom, "\text{yaw}"] & \mathcal{V} \arrow[l, bend left, "{}^{\mathcal{V}1}\boldsymbol{T}_{\mathcal{V},rotz}\left({}^{\mathcal{V}}\Psi\right)"] \arrow[r, bend left, "{}^{\mathcal{G}}\boldsymbol{T}_{\mathcal{V},tl}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)"] \arrow[r, phantom, "\text{shift}"] & \mathcal{G} \arrow[l, bend left, "{}^{\mathcal{V}}\boldsymbol{T}_{\mathcal{G},tl}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)"]
\end{tikzcd}$$

Figure 2.2: Euler coordinate transformation chain

According to the transformation chain in Fig.2.2, the transformation matrix from body $\mathcal{B}$ to vehicle frame $\mathcal{V}$ yields

$$\begin{aligned}
{}^{\mathcal{V}}\boldsymbol{T}_{\mathcal{B}} &= {}^{\mathcal{V}}\boldsymbol{T}_{\mathcal{V}1,rotz}{}^{\mathcal{V}1}\boldsymbol{T}_{\mathcal{V}2,roty}{}^{\mathcal{V}2}\boldsymbol{T}_{\mathcal{B},rotx} \\
&= \begin{bmatrix} \cos\Psi & -\sin\Psi & 0 & 0 \\ \sin\Psi & \cos\Psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & 0 & \sin\Theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Phi & -\sin\Phi & 0 \\ 0 & \sin\Phi & \cos\Phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \left[\begin{array}{cccc} \cos\Theta\cos\Psi & \cos\Psi\sin\Theta\sin\Phi - \cos\Phi\sin\Psi & \cos\Phi\cos\Psi\sin\Theta + \sin\Phi\sin\Psi & 0 \\ \cos\Theta\sin\Psi & \cos\Phi\cos\Psi + \sin\Theta\sin\Phi\sin\Psi & -\cos\Psi\sin\Phi + \cos\Phi\sin\Theta\sin\Psi & 0 \\ -\sin\Theta & \cos\Theta\sin\Phi & \cos\Theta\cos\Phi & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
\end{aligned} \tag{2.14}$$

end its inverse

$$\begin{aligned}
{}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{V}} &= \left({}^{\mathcal{V}}\boldsymbol{T}_{\mathcal{B}}\right)^{-1} \\
&= \begin{bmatrix} \cos(\Theta)\cos(\Psi) & \cos(\Theta)\sin(\Psi) & -\sin(\Theta) & 0 \\ \cos(\Psi)\sin(\Theta)\sin(\Phi) - \cos(\Phi)\sin(\Psi) & \cos(\Phi)\cos(\Psi) + \sin(\Theta)\sin(\Phi)\sin(\Psi) & \cos(\Theta)\sin(\Phi) & 0 \\ \cos(\Phi)\cos(\Psi)\sin(\Theta) + \sin(\Phi)\sin(\Psi) & \cos(\Phi)\sin(\Theta)\sin(\Psi) - \cos(\Psi)\sin(\Phi) & \cos(\Theta)\cos(\Phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.15}$$

Under the assumption that "Gimbal-Lock" singularities are avoided, the reverse transformation from $R_{\mathcal{B}}^{\mathcal{V}} \in \mathcal{R}^{3\times 3}$ to Euler angles is given by [WS16, p.12]

$$\Theta = \operatorname{arctan2}\left(-{}^{\mathcal{B}}\boldsymbol{R}_{\mathcal{V},31}, \sqrt{\left({}^{\mathcal{B}}\boldsymbol{R}_{\mathcal{V},11}{}^{2} + {}^{\mathcal{B}}\boldsymbol{R}_{\mathcal{V},21}{}^{2}\right)}\right) \tag{2.16}$$

$$\Phi = \operatorname{arctan2}\left(\frac{{}^{\mathcal{B}}\boldsymbol{R}_{\mathcal{V},32}}{\Theta}, \frac{{}^{\mathcal{B}}\boldsymbol{R}_{\mathcal{V},33}}{\Theta}\right) \tag{2.17}$$

$$\Psi = \operatorname{arctan2}\left(\frac{{}^{\mathcal{B}}\boldsymbol{R}_{\mathcal{V},21}}{\Theta}, \frac{{}^{\mathcal{B}}\boldsymbol{R}_{\mathcal{V},11}}{\Theta}\right). \tag{2.18}$$

The advantage of Euler angles is an intuitive understanding of the *UAV* pose. However, a major disadvantage is its discontinuous angle definition. The angles are defined on the intervals $-\pi < \Psi \leq \pi$, $-\pi/2 < \Theta \leq \pi/2$, $-\pi < \Phi \leq \pi$, thus a full rotation leads to a step (e.g. for Ψ between $\Psi = \pi$ and $\Psi = -\pi$). This would imply a velocity Dirac-impulse, which is not shown by the real system. One way to avoid this problem is the use of quaternions ρ.

2.3 Quaternion convention

Quaternions ρ are a representation of the 3D rotation group $\mathcal{SO}(3)$ which contains all rotations around the origin of $\mathbb{R}^3$. In contrast to Euler angles or rotation matrices (§2.2), the quaternion is composed of one real component ρ_w and three imaginary components ρ_x, ρ_y, ρ_z

$$\boldsymbol{\rho} = \rho_w + \rho_x\mathrm{i} + \rho_y\mathrm{j} + \rho_z\mathrm{k}. \tag{2.19}$$

In the context of this thesis, only unit quaternions ρ are considered. These fulfill the property

$$1 = \rho_w^2 + \rho_x^2 + \rho_y^2 + \rho_w^2. \tag{2.20}$$

A quaternion is representing a rotation around vector $\overrightarrow{\boldsymbol{p}} = [\rho_x, \rho_y, \rho_z]^\mathrm{T}$ by an angle $\alpha = \arccos(\frac{1}{2}\rho_w)$. This can be equally described by the rotation matrix

$${}^{\mathcal{V}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{V}}\boldsymbol{\rho}\right) := \left(\begin{bmatrix} 1-2\left(\rho_y^2+\rho_z^2\right) & 2\rho_x\rho_y - 2\rho_w\rho_z & 2(\rho_w\rho_y + \rho_x\rho_z) \\ 2(\rho_w\rho_z + \rho_x\rho_y) & 1-2\left(\rho_x^2+\rho_z^2\right) & 2\rho_y\rho_z - 2\rho_w\rho_x \\ 2\rho_x\rho_z - 2\rho_w\rho_y & 2(\rho_w\rho_x + \rho_y\rho_z) & 1-2\left(\rho_x^2+\rho_y^2\right) \end{bmatrix}\right)^{-1}. \tag{2.21}$$

Using computer algebraic methods, the direct inversion results in a large expression which is omitted here. However, considering unit quaternions, the resulting rotation matrices are orthogonal wherefore

$$\boldsymbol{R}^{-1} = \boldsymbol{R}^\mathrm{T} \tag{2.22}$$

holds. Under the assumption of unit quaternions, (2.21) therefore results to

$${}^{\mathcal{V}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{V}}\boldsymbol{\rho}\right) := \begin{bmatrix} 1-2\left(\rho_y^2+\rho_z^2\right) & 2\rho_x\rho_y - 2\rho_w\rho_z & 2\rho_x\rho_z - 2\rho_w\rho_y \\ 2(\rho_w\rho_z + \rho_x\rho_y) & 1-2\left(\rho_x^2+\rho_z^2\right) & 2(\rho_w\rho_x + \rho_y\rho_z) \\ 2(\rho_w\rho_y + \rho_x\rho_z) & 2\rho_y\rho_z - 2\rho_w\rho_x & 1-2\left(\rho_x^2+\rho_y^2\right) \end{bmatrix} \tag{2.23}$$

The Euler angles (Ψ, Θ, Φ) can be mapped to quaternions by [Hen77]

$$\begin{pmatrix} \rho_w \\ \rho_x \\ \rho_y \\ \rho_z \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\Phi}{2}\right)\cos\left(\frac{\Theta}{2}\right)\cos\left(\frac{\Psi}{2}\right) + \sin\left(\frac{\Phi}{2}\right)\sin\left(\frac{\Theta}{2}\right)\sin\left(\frac{\Psi}{2}\right) \\ \sin\left(\frac{\Phi}{2}\right)\cos\left(\frac{\Theta}{2}\right)\cos\left(\frac{\Psi}{2}\right) - \cos\left(\frac{\Phi}{2}\right)\sin\left(\frac{\Theta}{2}\right)\sin\left(\frac{\Psi}{2}\right) \\ \cos\left(\frac{\Phi}{2}\right)\sin\left(\frac{\Theta}{2}\right)\cos\left(\frac{\Psi}{2}\right) + \sin\left(\frac{\Phi}{2}\right)\cos\left(\frac{\Theta}{2}\right)\sin\left(\frac{\Psi}{2}\right) \\ \cos\left(\frac{\Phi}{2}\right)\cos\left(\frac{\Theta}{2}\right)\sin\left(\frac{\Psi}{2}\right) - \sin\left(\frac{\Phi}{2}\right)\sin\left(\frac{\Theta}{2}\right)\cos\left(\frac{\Psi}{2}\right) \end{pmatrix}. \tag{2.24}$$

and in reverse by [Hen77]

$$\begin{bmatrix} \Phi \\ \Theta \\ \Psi \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{2(\rho_w\rho_x - \rho_y\rho_z)}{1-2\left(\rho_x^2+\rho_y^2\right)}\right) \\ \arcsin\left(2\left(\rho_w\rho_y - \rho_z\rho_x\right)\right) \\ \arctan\left(\frac{2(\rho_w\rho_z - \rho_x\rho_y)}{1-2\left(\rho_y^2+\rho_z^2\right)}\right) \end{bmatrix} = \begin{bmatrix} \arctan\left(2\left(\rho_w\rho_x - \rho_y\rho_z\right), 1 - 2\left(\rho_x^2 + \rho_y^2\right)\right) \\ \arcsin\left(2\left(\rho_w\rho_y - \rho_z\rho_x\right)\right) \\ \arctan\left(2\left(\rho_w\rho_z - \rho_x\rho_y\right), 1 - 2\left(\rho_y^2 + \rho_z^2\right)\right) \end{bmatrix} \tag{2.25}$$

2.4 Denavit-Hartenberg convention

An alternative description of coordinate transformations which is used in the context of this work, is the Denavit-Hartenberg (*DH*) transformation for robotic arms. As the most common actuators for robotic arms are revolute and prismatic joints, the idea of *DH* is to describe this transformation based on the corresponding movement constraints. This principle is illustrated in Figure 2.3 showing the coordinate transformation from coordinate frame $\mathcal{J}_i$ in joint i to coordinate frame $\mathcal{J}_{i+1}$ in joint $i+1$. For this purpose, *DH* is aligning $\mathcal{J}_i$ with $\mathcal{J}_{i+1}$ by following translation and rotations [BH09, p.161]:

- 1: d_i: Translation of $\mathcal{J}_i$ to $\mathcal{J}_{iH1}$ by d_i along $z_{\mathcal{J}i}$-axis
- 2: θ_i : Rotation of $\mathcal{J}_{iH1}$ to $\mathcal{J}_{iH2}$ by θ_i around $z_{\mathcal{J}i}$-axis
- 3: a_i: Translation of $\mathcal{J}_{iH2}$ to $\mathcal{J}_{iH3}$ by a_i along $x_{\mathcal{J}i+1}$-axis
- 4: α_i: Rotation of $\mathcal{J}_{iH3}$ to $\mathcal{J}_{i+1}$ by α_i around $x_{\mathcal{J}i+1}$-axis.

The auxiliary coordinate frames $\mathcal{J}_{iH1}$-$\mathcal{J}_{iH3}$ are defined according to the translation/rotation steps. The four parameters d, θ, a, α are sufficient to describe the coordinate transformation between to two linked joints.

Considering a coordinate frame in each joint, the end effector pose can be determined by a sequence of coordinate transformations forming a kinematic chain. Starting from the base $i = 0$, each individual joint is consecutively transforming the coordinate system of the next joint until reaching the end effector coordinate frame. The elementary matrices to describe the transformations in step 1-4 have been introduced in (2.6)-(2.13) [WS16]. As a result, a point ${}^{\mathcal{J}_{i+1}}\vec{\boldsymbol{p}}$ defined in $\mathcal{J}_{i+1}$ is transformed into $\mathcal{J}_i$ coordinates using

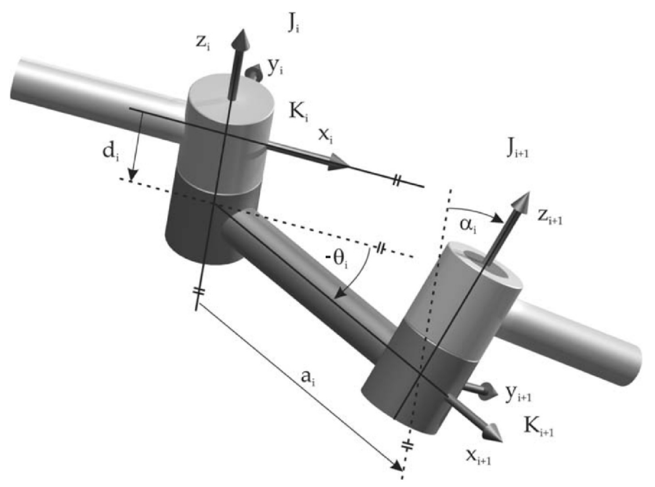


Figure 2.3: Denavit-Hartenberg convention [BH09, p.159] (θ sign corrected)

$$\begin{aligned}
\begin{bmatrix} {}^{\mathcal{J}_i}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} &= {}^{\mathcal{J}_i}\boldsymbol{T}_{\mathcal{J}_{i+1},DH}\left(\theta_i, d_i, a_i, \alpha_i\right) \begin{bmatrix} {}^{\mathcal{J}_{i+1}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} \qquad (2.26) \\
&= {}^{\mathcal{J}_{iH1}}\boldsymbol{T}_{\mathcal{J}_i,tl}\left(\begin{bmatrix} 0 \\ 0 \\ d_i \end{bmatrix}\right) {}^{\mathcal{J}_{iH2}}\boldsymbol{T}_{\mathcal{J}_{iH1},rotz}\left(\theta_i\right) {}^{\mathcal{J}_{iH3}}\boldsymbol{T}_{\mathcal{J}_{iH2},tl}\left(\begin{bmatrix} a_i \\ 0 \\ 0 \end{bmatrix}\right) {}^{\mathcal{J}_{i+1}}\boldsymbol{T}_{\mathcal{J}_{iH3},rotx}\left(\alpha_i\right) \begin{bmatrix} {}^{\mathcal{J}_{i+1}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{\mathcal{J}_{i+1}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix}
\end{aligned}$$

This specific transformation matrix is in the following indicated with the index DH. In reverse conclusion, the coordinates defined in $\mathcal{J}_i$ can be transformed into the $\mathcal{J}_{i+1}$ frame with the inverse

$$\begin{aligned}
{}^{\mathcal{J}_{i+1}}\boldsymbol{T}_{\mathcal{J}_i,DH}\left(\theta_i, d_i, a_i, \alpha_i\right) &= \left({}^{\mathcal{J}_i}\boldsymbol{T}_{\mathcal{J}_{i+1},DH}\left(\theta_i, d_i, a_i, \alpha_i\right)\right)^{-1} \qquad (2.27) \\
&= \begin{bmatrix} \cos(\theta_i) & \sin(\theta_i) & 0 & -a_i \\ -\cos(\alpha_i)\sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & \sin(\alpha_i) & -d_i\sin(\alpha_i) \\ \sin(\alpha_i)\sin(\theta_i) & -\cos(\theta_i)\sin(\alpha_i) & \cos(\alpha_i) & -d_i\cos(\alpha_i) \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned}$$

Chapter 3

Quadrotor models

A mathematical description of a *UAV*'s flight behavior is required to simulate or predict its motion. This chapter is dedicated to the modeling of *UAV*s. It starts with an overview of related work regarding different modeling approaches and sources in §3.1. To provide the physical background of *UAV* motion, a comprehensive physical model for quadrotor *UAV*s is derived in §3.2. This model is based on an Euler angle description which shows an undesired angle singularity. A physical model with quaternion description is developed in §3.3 to address this issue. For most commercial *UAV* systems such a physical model has to be adapted, as they are steered with velocity reference commands instead of force and torque inputs. This is challenging, as most manufacturers do not provide information about internal control structure and physical properties of the *UAV*. In addition, fast control update intervals and thus low computation times are required to cope with the fast *UAV* dynamics. Hence, the amount of mathematical operations and states has to be limited to minimize computation time and memory consumption. This is particularly crucial if the model is used in *MPC* on embedded systems. To satisfy these requirements, a Hover Model with single Yaw angle description (*HMDV*) is developed in §3.4. In order to avoid the singularity of the Ψ angle attitude description, a Hover Model with Direction Vector (*HMDV*) attitude description is derived in §3.5. The utilized discretization for the implementation of the quadrotor models on a computer platform is given in §3.6. Finally, the parameters for the derived models are identified for a real *AR.Drone 2.0* in experiments as well as a *DJI M100* quadrotor from simulation. The corresponding identification process and experimental setup are described in §3.7. A summary and discussion of the quadrotor modeling is given in §3.8.

3.1 Related work

This thesis is focusing on the deployment of quadrotor UAV systems. Regarding the $6DOF$ of a rigid body, quadrotors are under-actuated systems with the four rotor thrusts and the weight acting upon. A common way to describe UAV dynamics is to derive the rigid-body dynamics according to the Newton-Euler method. In addition, the aerodynamic effects, the electric drive, etc. are modeled to determine the force and torque acting upon the UAV. Most scientific literature assumes symmetry to simplify the resulting UAV model. The compact example for such a model is given in [HMLO02], providing a $6DOF$ physical model with rotor drag dynamics. A very comprehensive theoretical work on coordinate conventions, modeling of quadrotor kinematics and dynamics has been published in [Bea08]. This work develops a $6DOF$ physical model with rotor speed inputs, considering the UAV pose as well as linear and angular velocities. In this context, model simplification strategies are discussed and models for sensor behavior such as accelerometers, gyros and cameras are given. A similarly comprehensive work focusing on design and application of quadrotors is published in [Pou07]. This includes a study on design of a quadrotor and aerodynamic effects such as rotor dynamics, flapping and inflow distortion. These effects combined with electric drive dynamics are then combined in a mathematical model. The introduced theoretical principles are used to develop the “X-4” quadrotor with a detailed description of the utilized hardware. Similarly, the development of the “STARMAC II” and the modeling of its physical properties are presented in [HHWT07]. Also here, a full proprietary quadrotor solution is developed, starting from the electric drive properties, the aerodynamic effects to the rigid body UAV dynamics. In the “OS4” project, a compact summary of models for influences acting as forces and moments upon the UAV is given in [BS07b]. The extended version of the related quadrotor design is presented in [BS07a]. Using the Lagrangian is an alternative modeling approach to derive a rigid-body quadrotor model, as shown in [KKP09]. In [HD11], this is used to develop a model for a quadrotor with attached inverse pendulum. Both modeling approaches for UAV models are also discussed in the book [GDLP13].

Despite its extensive appearance in literature the presented physical modeling approaches have three limitations considering their application for this thesis. First, the presented sources use Euler angles to model the UAV attitude. However,the surjection of the Euler-angle attitude description poses a problem when used as a prediction model within a MPC. To address this issue, the attitude can be described with quaternions, as shown in [ACLV16]. The second disadvantage of such physical models is their computational complexity. This is for example addressed by linearizing the model around its stationary position. Examples for linearized state space models are presented in [Bou12], [ANT10]. Third, they cannot be easily used to model commercial UAV systems. As most commercial UAVs are controlled via lateral speed commands, the applied force and torque is a result of an internal controller and not directly measurable. This can be addressed by identifying all *UAV* subsystems and the controller, as shown in [Li14] for a *AR.Drone 2.0* quadrotor. However, such an extensive identification is challenging as the information and accessibility to physical parameters and internal software structure is limited. Within this thesis, this is approached by directly approximating and identifying the input-output behavior. The relation between *UAV* input and velocity response is thereby modeled with first order plants (*PT1*) and the velocities are mapped to the *UAV* pose with a nonlinear map. While this has been published in [DKMV16b], later on this approach has also been presented for an *AR.Drone 2.0* by [SF16]. A congruent model for the *AR.Drone 2.0* has been published in [SBSF16]. In [FBC13], the relation between input pitch and roll angle and the corresponding velocity response of an *AR.Drone 2.0* is identified similarly.

3.2 Physical model with Euler angle description

In order to relate the *UAV* position and orientation to the imposed force and torque, two integrations steps are required from acceleration over velocity. For a *UAV* pose with $6-DOF$, this can be described by a system of 12 *ODE*s, leading to a 12-state model. Based on the coordinate conventions of §2.2, the corresponding states of a model states are

$$\boldsymbol{x} = \left({}^{\mathcal{G}}x, {}^{\mathcal{G}}y, {}^{\mathcal{G}}z, {}^{\mathcal{G}}\dot{x}, {}^{\mathcal{G}}\dot{y}, {}^{\mathcal{G}}\dot{z}, {}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi, {}^{\mathcal{B}}\varrho, {}^{\mathcal{B}}\vartheta, {}^{\mathcal{B}}\omega\right)^{\mathrm{T}}. \tag{3.1}$$

${}^{\mathcal{G}}x$, ${}^{\mathcal{G}}y$, ${}^{\mathcal{G}}z$ are representing the *UAV* position coordinates in the global coordinate frame $\mathcal{G}$. Correspondingly ${}^{\mathcal{G}}\dot{x}$, ${}^{\mathcal{G}}\dot{z}$, ${}^{\mathcal{G}}\dot{z}$ are the *UAV* velocities in $\mathcal{G}$. The orientation of the *UAV* is described by the Euler angles ${}^{\mathcal{V}2}\Phi$, ${}^{\mathcal{V}1}\Theta$, ${}^{\mathcal{V}}\Psi$ which are defined in their corresponding frames ($\mathcal{V}2$, $\mathcal{V}1$, $\mathcal{V}$, see §2.2). The angular velocities around the body frame coordinate axes are given by ${}^{\mathcal{B}}\varrho$, ${}^{\mathcal{B}}\vartheta$, ${}^{\mathcal{B}}\omega$.

Similar to the model discussed in [Bea08], the relation between applied force $\overrightarrow{\boldsymbol{\Gamma}}$ and the change of body momentum is given by the second Euler-equation. Assuming a constant mass m, this results to

$${}^{\mathcal{B}}\overrightarrow{\boldsymbol{\Gamma}} = \begin{bmatrix} {}^{\mathcal{B}}\Gamma_x \\ {}^{\mathcal{B}}\Gamma_y \\ {}^{\mathcal{B}}\Gamma_z \end{bmatrix} = m \cdot \begin{bmatrix} {}^{\mathcal{B}}\ddot{x} \\ {}^{\mathcal{B}}\ddot{y} \\ {}^{\mathcal{B}}\ddot{z} \end{bmatrix}. \tag{3.2}$$

Accordingly, Euler's rotation equations are relating torque $\overrightarrow{\boldsymbol{\Lambda}}$ with angular velocity. With a constant inertia matrix $\boldsymbol{I_\iota} \in \mathbb{R}^{3\times3}$, this yields

$${}^{\mathcal{B}}\overrightarrow{\boldsymbol{\Lambda}} = \begin{bmatrix} {}^{\mathcal{B}}\Lambda_x \\ {}^{\mathcal{B}}\Lambda_y \\ {}^{\mathcal{B}}\Lambda_z \end{bmatrix} = \boldsymbol{I_\iota} \cdot \begin{bmatrix} {}^{\mathcal{B}}\dot{\varrho} \\ {}^{\mathcal{B}}\dot{\vartheta} \\ {}^{\mathcal{B}}\dot{\omega} \end{bmatrix} + \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \times \left(\boldsymbol{I_\iota} \cdot \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \right) \tag{3.3}$$

which can be reformulated to determine the angular accelerations

$$\begin{bmatrix} {}^{\mathcal{B}}\dot{\varrho} \\ {}^{\mathcal{B}}\dot{\vartheta} \\ {}^{\mathcal{B}}\dot{\omega} \end{bmatrix} = \boldsymbol{I_\iota}^{-1} \left(\begin{bmatrix} {}^{\mathcal{B}}\Lambda_x \\ {}^{\mathcal{B}}\Lambda_y \\ {}^{\mathcal{B}}\Lambda_z \end{bmatrix} - \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \times \left(\boldsymbol{I_\iota} \cdot \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \right) \right). \tag{3.4}$$

In order to determine the *UAV* dynamics in the global world frame $\mathcal{G}$, (3.2) and (3.4) have to be mapped from body coordinates $\mathcal{B}$ to world coordinates $\mathcal{G}$. For this mapping, the rotation matrix ${}^{\mathcal{V}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi\right) \equiv {}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi\right)$, introduced as (2.14) can be used:

$$\begin{bmatrix} {}^{\mathcal{G}}\ddot{x} \\ {}^{\mathcal{G}}\ddot{y} \\ {}^{\mathcal{G}}\ddot{z} \end{bmatrix} = \frac{1}{m} {}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi\right) \begin{bmatrix} {}^{\mathcal{B}}\Gamma_x \\ {}^{\mathcal{B}}\Gamma_y \\ {}^{\mathcal{B}}\Gamma_z \end{bmatrix} \tag{3.5}$$

The angular velocities ${}^{\mathcal{B}}\varrho$, ${}^{\mathcal{B}}\vartheta$, ${}^{\mathcal{B}}\omega$ are defined in the body frame. In order to describe the influence of the orientational dynamics in terms of Euler angles, these body frame velocities have to be mapped to the different coordinate frames ($\mathcal{V}2$, $\mathcal{V}1$, $\mathcal{V}$) in which the Euler angles are defined. For this purpose, the relation of the Euler angle derivatives are mapped to the angular body velocities via [Bea08, p.13]

$$\begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} = {}^{\mathcal{B}}\boldsymbol{R}_{\mathcal{V}2}\left({}^{\mathcal{V}2}\Phi\right) \left(\begin{bmatrix} {}^{\mathcal{V}2}\dot{\Phi} \\ 0 \\ 0 \end{bmatrix} + {}^{\mathcal{V}2}\boldsymbol{R}_{\mathcal{V}1}\left({}^{\mathcal{V}1}\Theta\right) \left(\begin{bmatrix} 0 \\ {}^{\mathcal{V}1}\dot{\Theta} \\ 0 \end{bmatrix} + {}^{\mathcal{V}1}\boldsymbol{R}_{\mathcal{V}}\left({}^{\mathcal{V}}\Psi\right) \begin{bmatrix} 0 \\ 0 \\ {}^{\mathcal{V}}\dot{\Psi} \end{bmatrix} \right) \right) \tag{3.6}$$

(3.6) can then be analytically solved to isolate the Euler angle derivatives

$$\begin{bmatrix} {}^{\mathcal{V}2}\dot{\Phi} \\ {}^{\mathcal{V}1}\dot{\Theta} \\ {}^{\mathcal{V}}\dot{\Psi} \end{bmatrix} = {}^{Euler}R_{\mathcal{B}}\left({}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi\right) \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \tag{3.7}$$

$$= \begin{bmatrix} 1 & \sin\left({}^{\mathcal{V}2}\Phi\right)\tan\left({}^{\mathcal{V}1}\Theta\right) & \cos\left({}^{\mathcal{V}2}\Phi\right)\tan\left({}^{\mathcal{V}1}\Theta\right) \\ 0 & \cos\left({}^{\mathcal{V}2}\Phi\right) & -\sin\left({}^{\mathcal{V}2}\Phi\right) \\ 0 & \sin\left({}^{\mathcal{V}2}\Phi\right)\sec\left({}^{\mathcal{V}1}\Theta\right) & \cos\left({}^{\mathcal{V}2}\Phi\right)\sec\left({}^{\mathcal{V}1}\Theta\right) \end{bmatrix} \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \tag{3.8}$$

Based on the states (3.1) the Euler equations (3.4), (3.5) and the rotation matrices (3.6), (3.8) the system function $\boldsymbol{f}\left(\boldsymbol{x}, \overrightarrow{\boldsymbol{\Gamma}}, \overrightarrow{\boldsymbol{\Lambda}}\right)$ for the rigid body dynamics can be concatenated

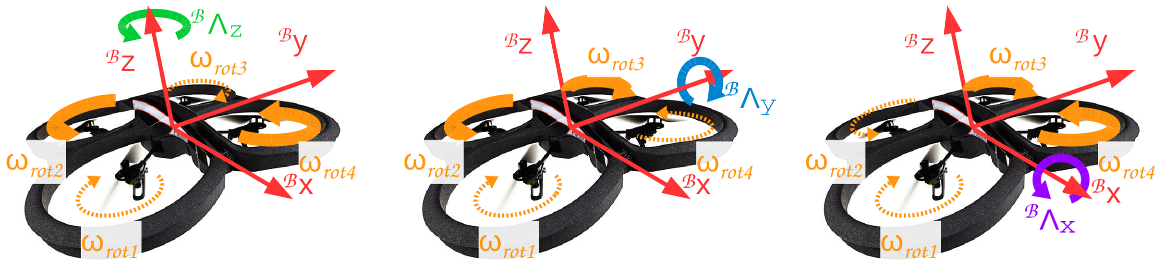
$$\dot{\boldsymbol{x}} = \begin{bmatrix} {}^{\mathcal{G}}\dot{x} \\ {}^{\mathcal{G}}\dot{y} \\ {}^{\mathcal{G}}\dot{z} \\ {}^{\mathcal{G}}\ddot{x} \\ {}^{\mathcal{G}}\ddot{y} \\ {}^{\mathcal{G}}\ddot{z} \\ {}^{\mathcal{V}2}\dot{\Phi} \\ {}^{\mathcal{V}1}\dot{\Theta} \\ {}^{\mathcal{V}}\dot{\Psi} \\ {}^{\mathcal{B}}\dot{\varrho} \\ {}^{\mathcal{B}}\dot{\vartheta} \\ {}^{\mathcal{B}}\dot{\omega} \end{bmatrix} = \boldsymbol{f}\left(\boldsymbol{x}, \overrightarrow{\boldsymbol{\Gamma}}, \overrightarrow{\boldsymbol{\Lambda}}\right) = \begin{bmatrix} {}^{\mathcal{G}}\dot{x} \\ {}^{\mathcal{G}}\dot{y} \\ {}^{\mathcal{G}}\dot{z} \\ \frac{1}{m}{}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi\right)\begin{bmatrix} {}^{\mathcal{B}}\Gamma_x \\ {}^{\mathcal{B}}\Gamma_y \\ {}^{\mathcal{B}}\Gamma_z \end{bmatrix} \\ {}^{Euler}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi\right)\begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \\ \boldsymbol{I}_{\boldsymbol{\iota}}{}^{-1}\left(\begin{bmatrix} {}^{\mathcal{B}}\Lambda_x \\ {}^{\mathcal{B}}\Lambda_y \\ {}^{\mathcal{B}}\Lambda_z \end{bmatrix} - \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \times \left(\boldsymbol{I}_{\boldsymbol{\iota}} \cdot \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix}\right)\right) \end{bmatrix}. \tag{3.9}$$

Assuming the quadrotor is operating on earth, it is exposed to gravity. Further assuming that ${}^{\mathcal{G}}z$ is chosen to point away from the gravity center of the earth, the weight force $\Gamma_g = m \cdot g$ is accelerating downwards in $-{}^{\mathcal{G}}z$ direction. Neglecting any external disturbances, the only force acting on the quadrotor besides weight is the thrust induced by its rotors. If all rotors are aligned in the ${}^{\mathcal{B}}x{}^{\mathcal{B}}y$-plane, the resulting thrust ${}^{\mathcal{B}}\Gamma_z$ is perpendicular to this plane in ${}^{\mathcal{B}}z$ direction. As a result the linear acceleration (3.5) results to

$$\begin{bmatrix} {}^{\mathcal{G}}\ddot{x} \\ {}^{\mathcal{G}}\ddot{y} \\ {}^{\mathcal{G}}\ddot{z} \end{bmatrix} = \frac{1}{m}{}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi\right)\begin{bmatrix} 0 \\ 0 \\ {}^{\mathcal{B}}\Gamma_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{3.10}$$

The torque $\vec{\boldsymbol{\Lambda}}$ is introduced by differing rotor speeds. Defining the system controls $\boldsymbol{u} = \left[{}^{\mathcal{B}}\Gamma, {}^{\mathcal{B}}\Lambda_x, {}^{\mathcal{B}}\Lambda_y, {}^{\mathcal{B}}\Lambda_z\right]^{\mathrm{T}}$, the system function $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ finally results to

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} {}^{\mathcal{G}}\dot{x} \\ {}^{\mathcal{G}}\dot{y} \\ {}^{\mathcal{G}}\dot{z} \\ \frac{1}{m}{}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi\right)\begin{bmatrix} 0 \\ 0 \\ {}^{\mathcal{B}}\Gamma_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \\ {}^{Euler}R_{\mathcal{B}}\left({}^{\mathcal{V}2}\Phi, {}^{\mathcal{V}1}\Theta, {}^{\mathcal{V}}\Psi\right)\begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \\ \boldsymbol{I_\iota}^{-1}\left(\begin{bmatrix} {}^{\mathcal{B}}\Lambda_x \\ {}^{\mathcal{B}}\Lambda_y \\ {}^{\mathcal{B}}\Lambda_z \end{bmatrix} - \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \times \left(\boldsymbol{I_\iota} \cdot \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix}\right)\right) \end{bmatrix}. \tag{3.11}$$



Positive yawing: ${}^{\mathcal{B}}\Lambda_z > 0$ Positive pitching: ${}^{\mathcal{B}}\Lambda_y > 0$ Positive rolling: ${}^{\mathcal{B}}\Lambda_x > 0$

Figure 3.1: Quadrotor rotor convention

Typically, no direct measurement of thrust ${}^{\mathcal{B}}\Gamma_z$ and torque $\vec{\boldsymbol{\Lambda}}$ are available in real quadrotor systems. However, for symmetric quadrotors they can be approximately mapped to the square of the rotor speeds ${\omega_{rot}}^2$ and vice versa. Considering the rotor convention shown in Figure 3.1, the mapping yields [Bea08, 16]

$$\begin{bmatrix} \Gamma \\ {}^{\mathcal{B}}\Lambda_x \\ {}^{\mathcal{B}}\Lambda_y \\ {}^{\mathcal{B}}\Lambda_z \end{bmatrix} = \begin{bmatrix} c_\Gamma & c_\Gamma & c_\Gamma & c_\Gamma \\ -d_{rot}c_\Gamma & -d_{rot}c_\Gamma & d_{rot}c_\Gamma & d_{rot}c_\Gamma \\ -d_{rot}c_\Gamma & d_{rot}c_\Gamma & d_{rot}c_\Gamma & -d_{rot}c_\Gamma \\ -c_\Psi & c_\Psi & c_\Psi & c_\Psi \end{bmatrix} \cdot \begin{bmatrix} {\omega_{rot}}_1^2 \\ {\omega_{rot}}_2^2 \\ {\omega_{rot}}_3^2 \\ {\omega_{rot}}_4^2 \end{bmatrix}. \tag{3.12}$$

The parameters c_Γ and c_Ψ are hereby system inherent parameters that are based on the rotor dimensions and shape with d_{rot} being the distance between drone center and rotor.

In order to facilitate the *UAV* design and control, most quadrotors are built symmetrically. Using this assumption of symmetry the inertia matrix becomes diagonal

$$\boldsymbol{I_\iota} = \begin{bmatrix} \iota\nu_x & 0 & 0 \\ 0 & \iota\nu_y & 0 \\ 0 & 0 & \iota\nu_z \end{bmatrix} \tag{3.13}$$

and the system function (3.11) simplifies to

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} {}^{\mathcal{G}}\dot{x} \\ {}^{\mathcal{G}}\dot{y} \\ {}^{\mathcal{G}}\dot{z} \\ \frac{{}^{\mathcal{B}}\Gamma_z}{m}\left(\cos\left({}^{\mathcal{V}2}\Phi\right)\cos\left({}^{\mathcal{V}}\Psi\right)\sin\left({}^{\mathcal{V}1}\Theta\right) + \sin\left({}^{\mathcal{V}2}\Phi\right)\sin\left({}^{\mathcal{V}}\Psi\right)\right) \\ \frac{{}^{\mathcal{B}}\Gamma_z}{m}\left(-\cos\left({}^{\mathcal{V}}\Psi\right)\sin\left({}^{\mathcal{V}2}\Phi\right) + \cos\left({}^{\mathcal{V}2}\Phi\right)\sin\left({}^{\mathcal{V}1}\Theta\right)\sin\left({}^{\mathcal{V}}\Psi\right)\right) \\ -g + \frac{{}^{\mathcal{B}}\Gamma_z}{m}\cos\left({}^{\mathcal{V}1}\Theta\right)\cos\left({}^{\mathcal{V}2}\Phi\right) \\ {}^{\mathcal{B}}\varrho + {}^{\mathcal{B}}\omega\cos\left({}^{\mathcal{V}2}\Phi\right)\tan{}^{\mathcal{V}1}\Theta + {}^{\mathcal{B}}\vartheta\sin\left({}^{\mathcal{V}2}\Phi\right)\tan{}^{\mathcal{V}1}\Theta \\ {}^{\mathcal{B}}\vartheta\cos\left({}^{\mathcal{V}2}\Phi\right) - {}^{\mathcal{B}}\omega\sin\left({}^{\mathcal{V}2}\Phi\right) \\ \left({}^{\mathcal{B}}\omega\cos\left({}^{\mathcal{V}2}\Phi\right) + {}^{\mathcal{B}}\vartheta\sin\left({}^{\mathcal{V}2}\Phi\right)\right)\sec{}^{\mathcal{V}1}\Theta \\ \frac{1}{\iota\nu_x}\left({}^{\mathcal{B}}\Lambda_x + {}^{\mathcal{B}}\vartheta{}^{\mathcal{B}}\omega(\iota\nu_y - \iota\nu_z)\right) \\ \frac{1}{\iota\nu_y}\left({}^{\mathcal{B}}\Lambda_y + {}^{\mathcal{B}}\varrho{}^{\mathcal{B}}\omega(-\iota\nu_x + \iota\nu_z)\right) \\ \frac{1}{\iota\nu_z}\left({}^{\mathcal{B}}\Lambda_z + {}^{\mathcal{B}}\varrho{}^{\mathcal{B}}\vartheta(\iota\nu_x - \iota\nu_y)\right) \end{bmatrix}. \tag{3.14}$$

3.3 Physical model with quaternion description

As previously discussed in §2.2, the use of Euler angles in (3.2) is disadvantageous due to their inherent singularities. One possibility to overcome this problem is to use quaternions (see convention §2.3) instead. This yields the state vector

$$\boldsymbol{x} = \left({}^{\mathcal{G}}x, {}^{\mathcal{G}}y, {}^{\mathcal{G}}z, {}^{\mathcal{G}}\dot{x}, {}^{\mathcal{G}}\dot{y}, {}^{\mathcal{G}}\dot{z}, {}^{\mathcal{G}}\rho_w, {}^{\mathcal{G}}\rho_x, {}^{\mathcal{G}}\rho_y, {}^{\mathcal{G}}\rho_z, {}^{\mathcal{B}}\varrho, {}^{\mathcal{B}}\vartheta, {}^{\mathcal{B}}\omega\right)^{\mathrm{T}}. \tag{3.15}$$

In order to derive the system function, the derivative of ${}^{\mathcal{G}}\boldsymbol{\rho}$ is determined using the scalar quaternion product $<,>$

$${}^{\mathcal{G}}\dot{\boldsymbol{\rho}} = \frac{1}{2} < {}^{\mathcal{G}}\boldsymbol{\rho}, \begin{bmatrix} 0 \\ {}^{\mathcal{G}}\varrho \\ {}^{\mathcal{G}}\vartheta \\ {}^{\mathcal{G}}\omega \end{bmatrix} > \tag{3.16}$$

which yields [WS16, 13]

$$\begin{bmatrix} {}^{\mathcal{G}}\dot{\rho}_w \\ {}^{\mathcal{G}}\dot{\rho}_x \\ {}^{\mathcal{G}}\dot{\rho}_y \\ {}^{\mathcal{G}}\dot{\rho}_z \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -{}^{\mathcal{G}}\rho_x & -{}^{\mathcal{G}}\rho_y & -{}^{\mathcal{G}}\rho_z \\ {}^{\mathcal{G}}\rho_w & {}^{\mathcal{G}}\rho_z & -{}^{\mathcal{G}}\rho_y \\ -{}^{\mathcal{G}}\rho_z & {}^{\mathcal{G}}\rho_w & {}^{\mathcal{G}}\rho_x \\ {}^{\mathcal{G}}\rho_y & -{}^{\mathcal{G}}\rho_x & {}^{\mathcal{G}}\rho_w \end{bmatrix} \cdot \begin{bmatrix} {}^{\mathcal{G}}\varrho \\ {}^{\mathcal{G}}\vartheta \\ {}^{\mathcal{G}}\omega \end{bmatrix}. \tag{3.17}$$

Under use of rotation matrix ${}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{G}}\boldsymbol{\rho}\right)$ (2.23) the system function results to

$$\boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}\right) = \begin{bmatrix} {}^{\mathcal{G}}\dot{x} \\ {}^{\mathcal{G}}\dot{y} \\ {}^{\mathcal{G}}\dot{z} \\ \frac{1}{m} {}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{G}}\boldsymbol{\rho}\right) \begin{bmatrix} 0 \\ 0 \\ \Gamma \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \\ \frac{1}{2} \begin{bmatrix} -{}^{\mathcal{G}}\rho_x & -{}^{\mathcal{G}}\rho_y & -{}^{\mathcal{G}}\rho_z \\ {}^{\mathcal{G}}\rho_w & {}^{\mathcal{G}}\rho_z & -{}^{\mathcal{G}}\rho_y \\ -{}^{\mathcal{G}}\rho_z & {}^{\mathcal{G}}\rho_w & {}^{\mathcal{G}}\rho_x \\ {}^{\mathcal{G}}\rho_y & -{}^{\mathcal{G}}\rho_x & {}^{\mathcal{G}}\rho_w \end{bmatrix} {}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{B}}\left({}^{\mathcal{G}}\boldsymbol{\rho}\right) \begin{bmatrix} {}^{\mathcal{G}}\varrho \\ {}^{\mathcal{G}}\vartheta \\ {}^{\mathcal{G}}\omega \end{bmatrix} \\ \boldsymbol{I_\iota}^{-1} \left(\begin{bmatrix} {}^{\mathcal{B}}\Lambda_x \\ {}^{\mathcal{B}}\Lambda_y \\ {}^{\mathcal{B}}\Lambda_z \end{bmatrix} - \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \times \left(\boldsymbol{I_\iota} \cdot \begin{bmatrix} {}^{\mathcal{B}}\varrho \\ {}^{\mathcal{B}}\vartheta \\ {}^{\mathcal{B}}\omega \end{bmatrix} \right) \right) \end{bmatrix}. \tag{3.18}$$

For the force/torque mapping to the rotor velocities again (3.12) can be used.

3.4 Hover model with yaw description

The physical thrust and torque based models introduced in §3.2-§3.3 are well suited to describe physical effects and the flight behavior of a *UAV*. However, they are not well suited to predict the behavior of commercial quadrotors for two reasons:

- Commercial *UAV*s have internal controllers that are typically based on velocity references in the $\mathcal{V}1$-frame. In addition, there are typically no inputs or measurements for ${}^{\mathcal{B}}\Gamma_z, {}^{\mathcal{B}}\Lambda_x, {}^{\mathcal{B}}\Lambda_y, {}^{\mathcal{B}}\Lambda_z$ or the rotor speed ω_{rot} and the underlying dynamics available.
- The model complexity is indicated by the multitude of mathematical operations. This leads to high computation times in simulations and as *MPC* model.

For these reasons, a more compact model for commercial quadrotor is advantageous which directly relays on velocity reference inputs. One approach is, to simplify the physical under with the assumption that most mobile robots are designed to move in a planar space. Their position is defined in a ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane and the height ${}^{\mathcal{G}}z$ of this plane. Accordingly, their attitude is defined by the rotation angle Ψ around ${}^{\mathcal{G}}z$. This description does fit multi-rotor *UAV*s, as they are typically operated around their static equilibrium, the hovering position. By neglecting the small angles in pitch Θ and roll Φ, the forward and sideward movement of the quadrotor is considered to be in the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane. Consequently, $\mathcal{B}$ and $\mathcal{V}1$ are identical. The attitude is thus only defined by the heading, given as yaw angle ${}^{\mathcal{V}}\Psi$ rotation around the ${}^{\mathcal{G}}z$ axis. Hence, the state of a multi-rotor *UAV* can be described with the vector

$$\boldsymbol{x}(t) = \left[{}^{\mathcal{G}}x(t), {}^{\mathcal{G}}y(t), {}^{\mathcal{G}}z(t), {}^{\mathcal{V}}\Psi(t), {}^{\mathcal{V}1}\dot{x}(t), {}^{\mathcal{V}1}\dot{y}(t), {}^{\mathcal{V}1}\dot{z}(t), {}^{\mathcal{V}}\dot{\Psi}(t),\right]^{\mathrm{T}}. \tag{3.19}$$

The x, y, z position and Ψ orientation are in the global frame $\mathcal{G}$. $\dot{x}$, $\dot{y}$, $\dot{z}$ and $\dot{\Psi}$ are representing forward, leftward, upward and heading velocity of the *UAV*. The controls

$$\boldsymbol{u} = \left[u_{\mathcal{V}1_x}, u_{\mathcal{V}1_y}, u_{\mathcal{V}1_z}, u_{\mathcal{V}1_\omega}\right]^{\mathrm{T}} \tag{3.20}$$

are given as forward $u_{\mathcal{V}1_x}$, leftward $u_{\mathcal{V}1_y}$, upward $u_{\mathcal{V}1_z}$ and heading velocity $u_{\mathcal{V}1_\omega}$ references in the vehicle 1 frame $\mathcal{V}1$.

A real *UAV* is not instantly reaching these reference velocities. For the appxorimation of this settling behavior, a *PT1* can be used. A *PT1* with state x, input u, settling time constant t_s and proportional gain k_p is defined by *ODE*

$$\dot{x}(t) = -\frac{1}{t_s}x(t) + \frac{k_p}{t_s}u(t) \tag{3.21}$$

The step response of a *PT1* is asymptotically approaching k_p

$$x(t) = k_p\left(1 - e^{-\frac{t}{t_s}}\right), \tag{3.22}$$

as shown in Figure 3.2.

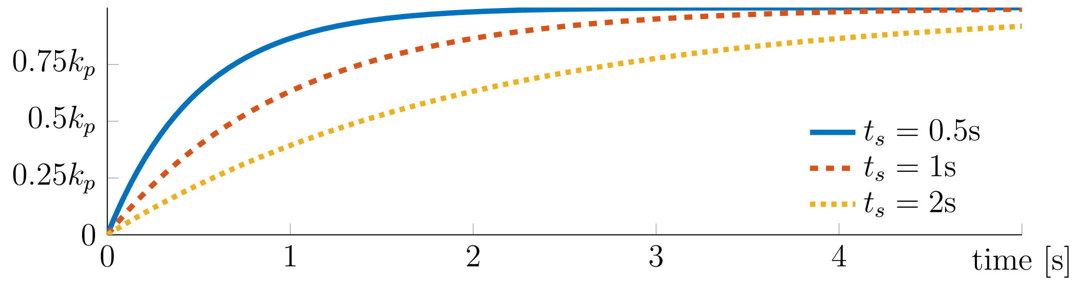


Figure 3.2: First order plant step response

By substituting the factors $a = -\frac{1}{t_s}$ and $b = \frac{k_p}{t_s}$ and applying the rotation matrix ${}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{V}1}\left({}^{\mathcal{V}}\Psi(t)\right) \equiv {}^{\mathcal{V}}\boldsymbol{R}_{\mathcal{V}1}\left({}^{\mathcal{V}}\Psi(t)\right)$ deduced from transformation matrix ${}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{A}rotz}(\Psi)$ (2.7) the system function can be derived. For the Hover Model with Yaw Attitude Description

(*HMDV*) this yields

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} \left. {}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{V}1}\left({}^{\mathcal{V}}\Psi(t)\right) \cdot \begin{bmatrix} {}^{\mathcal{V}1}\dot{x}(t) \\ {}^{\mathcal{V}1}\dot{x}(t) \\ {}^{\mathcal{V}1}\dot{z}(t) \end{bmatrix} \right\} \text{Nonlinear Map: } \mathcal{V}1 \rightarrow \mathcal{G} \\ {}^{\mathcal{V}}\dot{\Psi}(t) \\ \left. \begin{matrix} a_x \cdot {}^{\mathcal{V}1}\dot{x}(t) + b_x \cdot u_{\mathcal{V}1_x}(t) \\ a_y \cdot {}^{\mathcal{V}1}\dot{y}(t) + b_y \cdot u_{\mathcal{V}1_y}(t) \\ a_z \cdot {}^{\mathcal{V}1}\dot{z}(t) + b_z \cdot u_{\mathcal{V}1_z}(t) \\ a_\Psi \cdot {}^{\mathcal{V}1}\dot{\Psi}(t) + b_\Psi \cdot u_{\mathcal{V}1_\omega}(t) \end{matrix} \right\} \textit{PT1} \text{ velocity models.} \end{bmatrix} \tag{3.23}$$

which is equal to

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} {}^{\mathcal{V}1}\dot{x}(t)\cos\left({}^{\mathcal{V}}\Psi(t)\right) - {}^{\mathcal{V}1}\dot{y}(t)\sin\left({}^{\mathcal{V}}\Psi(t)\right) \\ {}^{\mathcal{V}1}\dot{x}(t)\sin\left({}^{\mathcal{V}}\Psi(t)\right) + {}^{\mathcal{V}1}\dot{y}(t)\cos\left({}^{\mathcal{V}}\Psi(t)\right) \\ {}^{\mathcal{G}}\dot{z}(t) \\ {}^{\mathcal{V}}\dot{\Psi}(t) \\ a_x \cdot {}^{\mathcal{V}1}\dot{x}(t) + b_x \cdot u_{\mathcal{V}1_x}(t) \\ a_y \cdot {}^{\mathcal{V}1}\dot{y}(t) + b_y \cdot u_{\mathcal{V}1_y}(t) \\ a_z \cdot {}^{\mathcal{V}1}\dot{z}(t) + b_z \cdot u_{\mathcal{V}1_z}(t) \\ a_\Psi \cdot {}^{\mathcal{V}1}\dot{\Psi}(t) + b_\Psi \cdot u_{\mathcal{V}1_\omega}(t) \end{bmatrix}. \tag{3.24}$$

For ${}^{\mathcal{V}}\Psi(t) = 0$ the rotation matrix equals the identity matrix ${}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{V}1}(0) = \boldsymbol{I}_3$ which makes the double integrator structure of (3.24) visible. The parameters $\varkappa$

$$\varkappa = \left[a_x, b_x, a_y, b_y, a_z, b_z, a_\Psi, b_\Psi\right]^{\mathrm{T}} \tag{3.25}$$

are depending on the *UAV* system and have to be identified individually.

Reduced model for lower memory consumption

The memory consumption of the prediction model used within the *MPC* is crucial. As the model states and inputs are stored for the whole prediction horizon and potentially for some previous horizons, the allocated memory is a multitude of the model states. This is limiting its application on embedded systems. As a result, it can be useful to reduce the state dimension of (3.24) further to a more compact model. Under consideration of a fast internal controller, ${}^{\mathcal{V}1}\dot{z}(t) = b_z \cdot u_{\mathcal{V}1_z}(t)$ and ${}^{\mathcal{V}}\dot{\Psi}(t) = b_\Psi \cdot u_{\mathcal{V}1_\omega}(t)$ can be assumed. Hence, these are eliminated from the state vector

$$\boldsymbol{x}(t) = \left[{}^{\mathcal{G}}x(t), {}^{\mathcal{G}}y(t), {}^{\mathcal{G}}z(t), {}^{\mathcal{V}}\Psi(t), {}^{\mathcal{V}1}\dot{x}(t), {}^{\mathcal{V}1}\dot{y}(t),\right]^{\mathrm{T}}, \tag{3.26}$$

which results in the reduced system function

$$\boldsymbol{f}\left(\boldsymbol{x},\boldsymbol{u}\right)=\begin{bmatrix} {}^{\mathcal{V}1}\dot{x}\left(t\right)\cos\left({}^{\mathcal{V}}\Psi\left(t\right)\right)-{}^{\mathcal{V}1}\dot{y}\left(t\right)\sin\left({}^{\mathcal{V}}\Psi\left(t\right)\right)\\ {}^{\mathcal{V}1}\dot{x}\left(t\right)\sin\left({}^{\mathcal{V}}\Psi\left(t\right)\right)+{}^{\mathcal{V}1}\dot{y}\left(t\right)\cos\left({}^{\mathcal{V}}\Psi\left(t\right)\right)\\ b_z\cdot u_{\mathcal{V}1_z}\left(t\right)\\ b_\Psi\cdot u_{\mathcal{V}1_\omega}\left(t\right)\\ a_x\cdot{}^{\mathcal{V}1}\dot{x}\left(t\right)+b_x\cdot u_{\mathcal{V}1_x}\left(t\right)\\ a_y\cdot{}^{\mathcal{V}1}\dot{y}\left(t\right)+b_y\cdot u_{\mathcal{V}1_y}\left(t\right)\end{bmatrix}. \tag{3.27}$$

The result is a Reduced Hover Model with Yaw Attitude Description (*RHMY*). For surveillance and transportation tasks which are typically executed in the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane, this assumption does not directly reduce the systems performance. However, real *UAV*s do not comply with the assumption of an infinitely fast closed-loop system in ${}^{\mathcal{V}1}\dot{z}\left(t\right)$ and ${}^{\mathcal{V}}\dot{\Psi}\left(t\right)$. The resulting modeling error of the reduced *MPC* prediction model leads to a lower control performance and may cause instability. The usage of this model is therefore limited to systems, where the internal control is sufficiently fast.

3.5 Hover model with direction vector attitude description

The developed system model (3.24) and its reduced version (3.27) can be used to describe mobile robots which are moving in a plane by using a single Ψ-angle attitude description. This section is presenting a direction vector approach in order to address the previously in §2.2 mentioned discontinuous angle definition which is based on the defined yaw angle ${}^{\mathcal{V}}\Psi$ interval

$${}^{\mathcal{V}}\Psi:=\{{}^{\mathcal{V}}\Psi\in\mathbb{R}|-\pi<{}^{\mathcal{V}}\Psi\leq\pi\}. \tag{3.28}$$

For a constantly rotating robot with angular velocity ${}^{\mathcal{V}}\dot{\Psi}=1$, the angle shows the surjection for a limited ${}^{\mathcal{V}}\Psi$ interval as shown in Figure 3.3. This is problematic, if the attitude is controlled by means of velocity or acceleration. To show the problematic behavior, a *MPC* is applied to an *AR.Drone 2.0* quadrotor using the reduced prediction model (3.27) with the parameters chosen to

$$\left[a_x,b_x,a_y,b_y,b_z,b_\Psi\right]^{\mathrm{T}}=\left[-0.5092,1.458,-0.5092,1.458,1,1.6\right]^{\mathrm{T}} \tag{3.29}$$

The resulting trajectories are given in Figure 3.4. To control the quadrotor attitude, traditionally the error $e_{\mathcal{V}\Psi}={}^{\mathcal{V}}\Psi_{des}-{}^{\mathcal{V}}\Psi$ to the desired yaw angle ${}^{\mathcal{V}}\Psi_{des}$ is minimized. The yaw angle plot in Figure 3.4a shows a step at $t\approx 84.5\,\mathrm{s}$ in the desired angle from

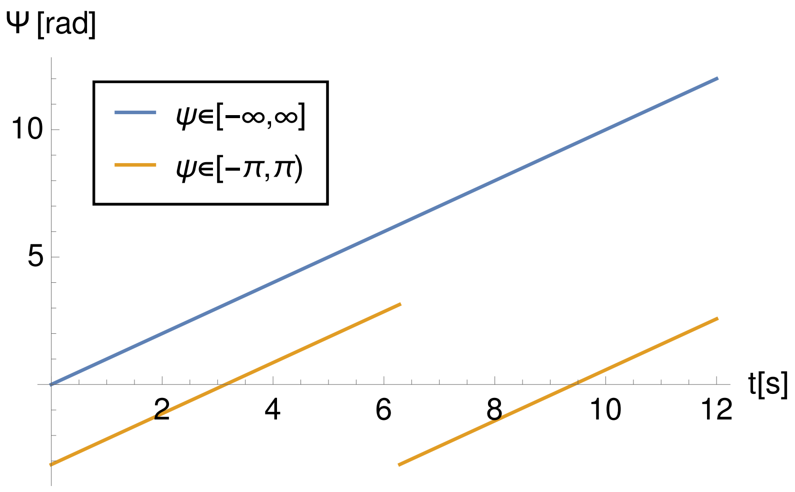


Figure 3.3: Angle discontinuity visualization with constant velocity

${}^{\mathcal{V}}\Psi_{des}\,(0\,\mathrm{s}) = \frac{\pi}{2}$ to ${}^{\mathcal{V}}\Psi_{des}\,(\approx 84.5\,\mathrm{s}) = \pi$. To minimize $e_{\mathcal{V}_\Psi}$, the controller is applying a positive angular velocity to converge towards ${}^{\mathcal{V}}\Psi_{des} = \pi$. As real mobile robotic systems are exposed to disturbance, the robot is likely to overshoot ${}^{\mathcal{V}}\Psi_{des} = \pi$ which leads to a change of sign to ${}^{\mathcal{V}}\Psi = -\pi$. Hence, the quadrotor will again try to reach the desired value from the new angle ${}^{\mathcal{V}}\Psi = -\pi$. As a result, positions close to ${}^{\mathcal{V}}\Psi_{des} = \pm\pi$ cannot be stabilized which leads to a repetitive rotational movement. The resulting high control reaction in Figure 3.4c leads to a disturbance in the position trajectory, as shown in Figure 3.4b.

Consequently, an alternative angle representation in the planar rotation matrix

$$ {}^{\mathcal{V}1}\vec{\boldsymbol{p}} = \begin{bmatrix} \cos({}^{\mathcal{V}}\Psi) & -\sin({}^{\mathcal{V}}\Psi) \\ \sin({}^{\mathcal{V}}\Psi) & \cos({}^{\mathcal{V}}\Psi) \end{bmatrix} {}^{\mathcal{G}}\vec{\boldsymbol{p}} = {}^{\mathcal{V}1}\boldsymbol{R}_{\mathcal{G}}\,{}^{\mathcal{G}}\vec{\boldsymbol{p}} \qquad \text{with } {}^{\mathcal{G}}\vec{\boldsymbol{p}} = \begin{bmatrix} {}^{\mathcal{G}}\vec{\boldsymbol{x}} \\ {}^{\mathcal{G}}\vec{\boldsymbol{y}} \end{bmatrix} \tag{3.30} $$

has to be found.

One way to do this is to use a direction vector. For $3D$-space this is well established in the form of quaternions $\boldsymbol{\rho} \in \mathcal{R}^4$. In state-space models, the four elements of $\boldsymbol{\rho}$ are treated as separate states. This is disadvantageous for real-time MPC, as the related increase in the state dimension is also significantly increasing the memory consumption. The alternative is to describe the $2D$ rotation in the xy-plane with a direction vector. For this purpose, the direction vector is representing a unit vector ${}^{\mathcal{G}}\boldsymbol{o}$ which is defined by its projections onto the coordinate axes ${}^{\mathcal{G}}o_x$ and ${}^{\mathcal{G}}o_y$ as shown in Figure 3.5. The orientation of the drone is therefore unambiguously defined by ${}^{\mathcal{G}}\boldsymbol{o}$.

$$ {}^{\mathcal{G}}\boldsymbol{o} = \begin{bmatrix} {}^{\mathcal{G}}o_x \\ {}^{\mathcal{G}}o_y \end{bmatrix} = \begin{bmatrix} \cos({}^{\mathcal{V}}\Psi) \\ \sin({}^{\mathcal{V}}\Psi) \end{bmatrix} \qquad \begin{matrix} {}^{\mathcal{G}}o_x := \{{}^{\mathcal{G}}o_x \in \mathbb{R} | -1 < {}^{\mathcal{G}}o_x \leq 1\} \\ {}^{\mathcal{G}}o_y := \{{}^{\mathcal{G}}o_y \in \mathbb{R} | -1 < {}^{\mathcal{G}}o_y \leq 1\} \end{matrix}. \tag{3.31} $$

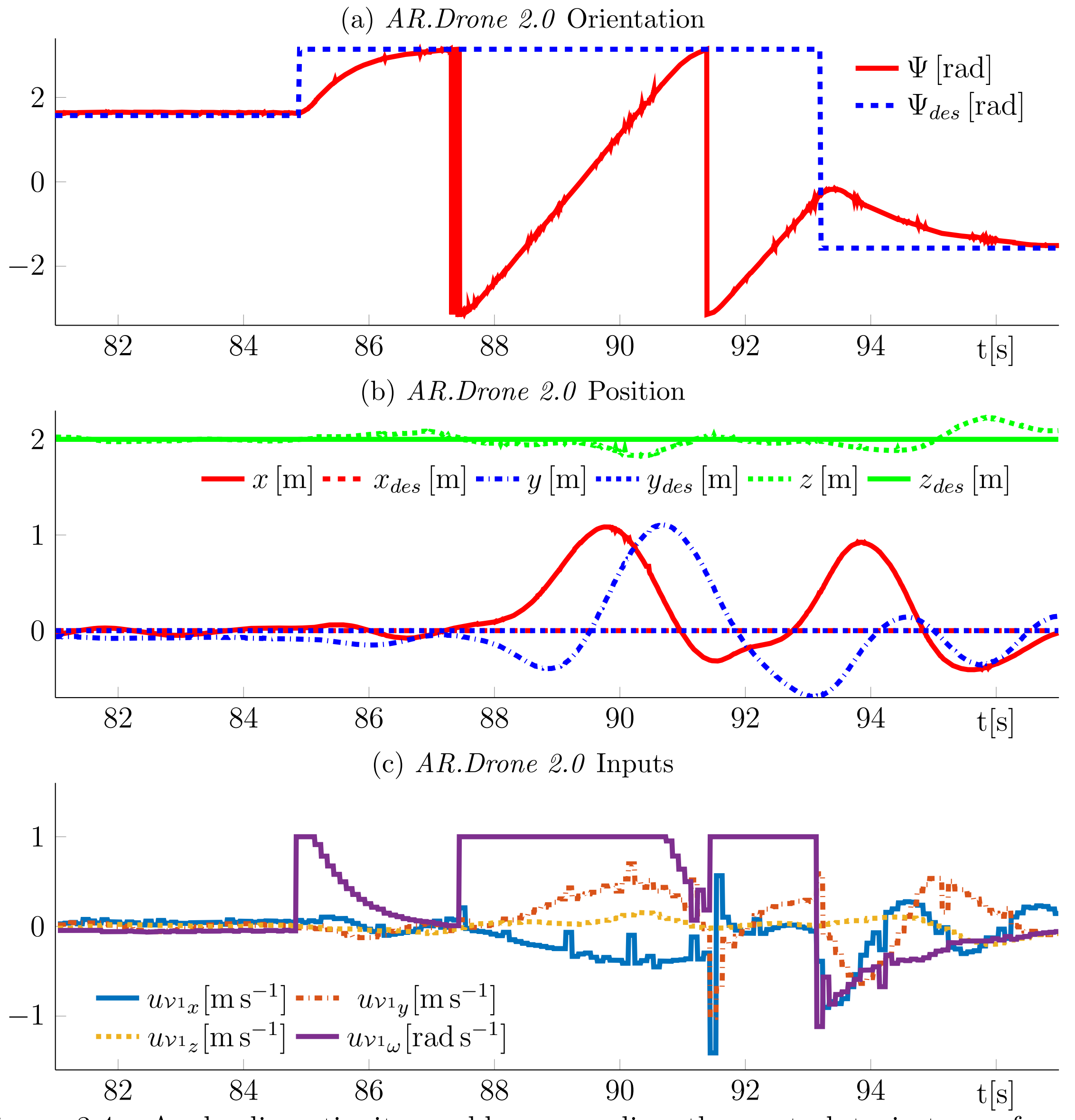


Figure 3.4: Angle discontinuity problem regarding the control trajectory of an *AR.Drone 2.0*

In comparison to a single angle description with yaw ${}^{\mathcal{V}}\Psi$, this transformation is bijective.

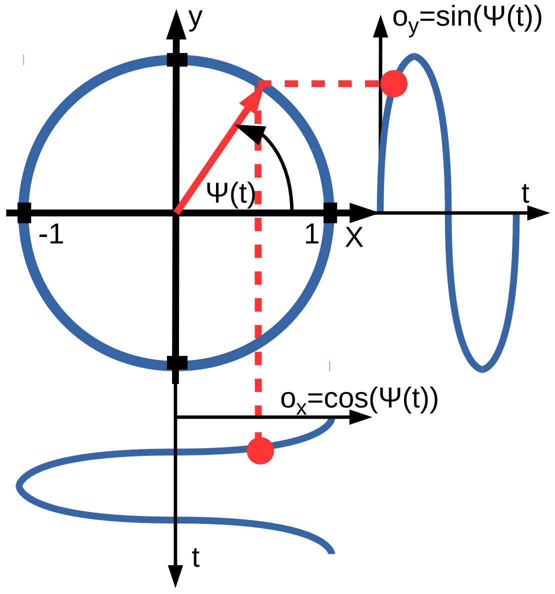


Figure 3.5: Unit circle with direction vector projections

As the direction vector is expressing the projection from the unit circle, ${}^{\mathcal{G}}o_x$ and ${}^{\mathcal{G}}o_y$ are fulfilling the circle constraint

$$1 = {}^{\mathcal{G}}{o_x}^2 + {}^{\mathcal{G}}{o_y}^2. \tag{3.32}$$

Using (3.31) to receive the state-space description, ${}^{\mathcal{G}}\dot{\boldsymbol{o}}$ yields

$${}^{\mathcal{G}}\dot{\boldsymbol{o}} = \begin{bmatrix} {}^{\mathcal{G}}\dot{o_x} \\ {}^{\mathcal{G}}\dot{o_y} \end{bmatrix} = \begin{bmatrix} -\sin({}^{\mathcal{V}}\Psi) \\ \cos({}^{\mathcal{V}}\Psi) \end{bmatrix} {}^{\mathcal{V}}\dot{\Psi} = \begin{bmatrix} -{}^{\mathcal{G}}o_y \\ {}^{\mathcal{G}}o_x \end{bmatrix} {}^{\mathcal{V}}\dot{\Psi} \tag{3.33}$$

This is intuitive, as the derivative ${}^{\mathcal{G}}\dot{\boldsymbol{o}}$ has to be orthogonal to ${}^{\mathcal{G}}\boldsymbol{o}$ to force ${}^{\mathcal{G}}\boldsymbol{o}$ to stay on the unit circle. Transformation matrix (3.30) is transformed with (3.31) to

$${}^{\mathcal{V}1}\boldsymbol{R}_{\mathcal{G}}\left({}^{\mathcal{G}}o_x, {}^{\mathcal{G}}o_y\right) = \begin{bmatrix} {}^{\mathcal{G}}o_x & -{}^{\mathcal{G}}o_y \\ {}^{\mathcal{G}}o_y & {}^{\mathcal{G}}o_x \end{bmatrix} \tag{3.34}$$

Applied to the reduced system model, the angular velocity ${}^{\mathcal{V}}\dot{\Psi}$ in (3.33) is given by the system input $u_{\mathcal{V}_\Psi}$ with the constant factor b_Ψ. Hence, the derivative yields to

$${}^{\mathcal{G}}\dot{\boldsymbol{o}} = \begin{bmatrix} -{}^{\mathcal{G}}o_y \\ {}^{\mathcal{G}}o_x \end{bmatrix} b_{\mathcal{V}_\Psi} \cdot u_{\mathcal{V}_\Psi}(t) \tag{3.35}$$

To transform the system dynamics (3.27) into a direction vector model, ${}^{\mathcal{V}}\Psi$ is substituted by the direction vector in the state vector

$$\boldsymbol{x}(t) = \left[{}^{\mathcal{G}}x(t), {}^{\mathcal{G}}y(t), {}^{\mathcal{G}}z(t), {}^{\mathcal{G}}o_x(t), {}^{\mathcal{G}}o_y(t), {}^{\mathcal{V}1}\dot{x}(t), {}^{\mathcal{V}1}\dot{y}(t)\right]^{\mathrm{T}}. \tag{3.36}$$

Using the direction vector (3.31), the derivative (3.35) with $u_{\mathcal{V}\Psi}$ and the coordinate transformation (3.34), the system dynamics can be derived from (3.27). For the Reduced Hover Model with Direction Vector Attitude Description (*RHMDV*) this finally leads to

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \begin{bmatrix} {}^{\mathcal{V}1}\dot{x}(t)\,{}^{\mathcal{G}}o_x(t) - {}^{\mathcal{V}1}\dot{y}(t)\,{}^{\mathcal{G}}o_y(t) \\ {}^{\mathcal{V}1}\dot{x}(t)\,{}^{\mathcal{G}}o_y(t) + {}^{\mathcal{V}1}\dot{y}(t)\,{}^{\mathcal{G}}o_x(t) \\ a_z \cdot z(t) + b_z \cdot u_{\mathcal{V}1_z}(t) \\ -{}^{\mathcal{G}}o_y(t) \cdot b_{\mathcal{V}\Psi} \cdot u_{\mathcal{V}\Psi}(t) \\ {}^{\mathcal{G}}o_x(t) \cdot b_{\mathcal{V}\Psi} \cdot u_{\mathcal{V}\Psi}(t) \\ a_f \cdot {}^{\mathcal{V}1}\dot{x}(t) + b_f \cdot u_{\mathcal{V}1_x}(t) \\ a_s \cdot {}^{\mathcal{V}1}\dot{y}(t) - b_s \cdot u_{\mathcal{V}1_y}(t) \end{bmatrix} \tag{3.37}$$

To track attitude ${}^{\mathcal{G}}\boldsymbol{o}$, a simple quadratic penalty can be used

$$L_{\boldsymbol{o}} = \left({}^{\mathcal{G}}\boldsymbol{o}_{des} - {}^{\mathcal{G}}\boldsymbol{o}\right)^{\mathrm{T}} \begin{bmatrix} q_{o_x} & 0 \\ 0 & q_{o_y} \end{bmatrix} \left({}^{\mathcal{G}}\boldsymbol{o}_{des} - {}^{\mathcal{G}}\boldsymbol{o}\right) \tag{3.38}$$

where q_{o_x} and q_{o_y} represent orientation penalty factors. This is valid under the condition, that numerical errors of the *MPC* solver avoid the singular problem of opposing attitudes, e.g. ${}^{\mathcal{G}}\boldsymbol{o}_{des} = [-1, 0]^{\mathrm{T}}$ with ${}^{\mathcal{G}}\boldsymbol{o} = [1, 0]^{\mathrm{T}}$, or ${}^{\mathcal{G}}\boldsymbol{o}_{des} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^{\mathrm{T}}$ with ${}^{\mathcal{G}}\boldsymbol{o} = [-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}]^{\mathrm{T}}$.

To validate the direction vector quadrotor model (3.37) experimentally, a *NMPC* pose tracking scenario with a real *AR.Drone 2.0* is conducted (for more details, see §5). In this scenario, the desired drone attitude is rotated anticlockwise in steps of $\Delta^{\mathcal{V}}\Psi = \frac{\pi}{2}$. The ${}^{\mathcal{V}}\Psi$ plot in Figure 3.6 shows the desired and resulting real attitude of the system. In contrast to the previous instability at ${}^{\mathcal{V}}\Psi = \pm\pi$ (Figure 3.4), Figure 3.6 is showing a stable asymptotic convergence towards the desired trajectory. The oscillations in x, y, z around the desired point are caused by disturbance. This includes airflow disturbance, modeling errors, numerical errors and the trade-off between energy optimality and position tracking. Hence, the direction vector approach is resolving the angle discontinuity problem.

Applying the same direction vector description on the model (3.24) results to a Hover Model with Direction Vector Attitude Description (*HMDV*). The state vector yields

$$\boldsymbol{x}(t) = \left[{}^{\mathcal{G}}x(t), {}^{\mathcal{G}}y(t), {}^{\mathcal{G}}z(t), {}^{\mathcal{G}}o_x(t), {}^{\mathcal{G}}o_y(t), {}^{\mathcal{V}1}\dot{x}(t), {}^{\mathcal{V}1}\dot{y}(t), {}^{\mathcal{V}1}\dot{z}(t), {}^{\mathcal{V}}\dot{\Psi}(t),\right]^{\mathrm{T}} \tag{3.39}$$

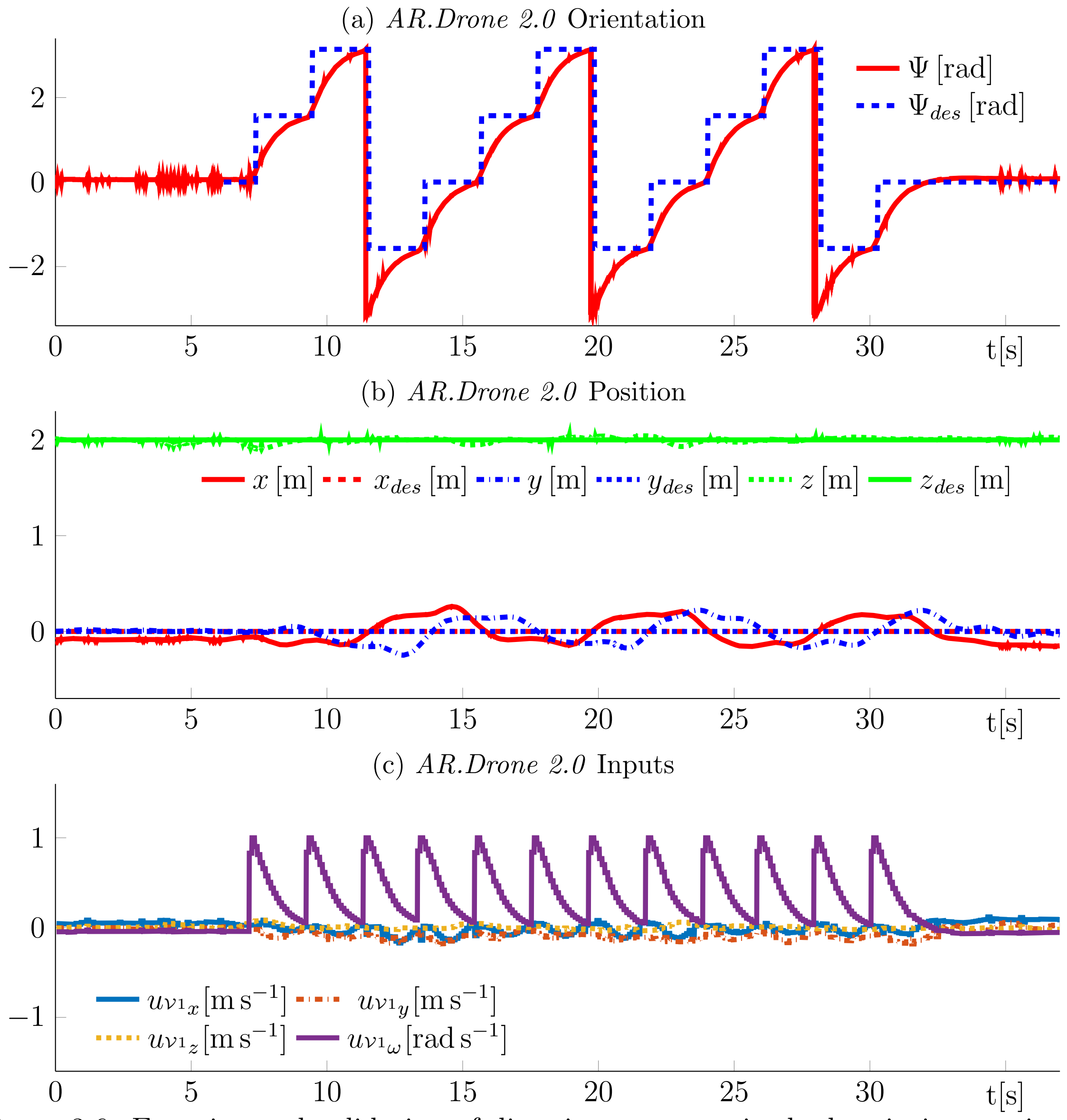


Figure 3.6: Experimental validation of direction vector attitude description: trajectory of a real *AR.Drone 2.0*

and the system function

$$\boldsymbol{f}\left(\boldsymbol{x},\boldsymbol{u}\right)=\begin{bmatrix} {}^{\mathcal{V}1}\dot{x}\left(t\right){}^{\mathcal{G}}o_x\left(t\right)-{}^{\mathcal{V}1}\dot{y}\left(t\right){}^{\mathcal{G}}o_y\left(t\right) \\ {}^{\mathcal{V}1}\dot{x}\left(t\right){}^{\mathcal{G}}o_y\left(t\right)+{}^{\mathcal{V}1}\dot{y}\left(t\right){}^{\mathcal{G}}o_x\left(t\right) \\ {}^{\mathcal{G}}\dot{z}\left(t\right) \\ -{}^{\mathcal{G}}o_y\left(t\right)\cdot{}^{\mathcal{V}}\dot{\Psi}\left(t\right) \\ {}^{\mathcal{G}}o_x\left(t\right)\cdot{}^{\mathcal{V}}\dot{\Psi}\left(t\right) \\ a_x\cdot{}^{\mathcal{V}1}\dot{x}\left(t\right)+b_x\cdot u_{\mathcal{V}1_x}\left(t\right) \\ a_y\cdot{}^{\mathcal{V}1}\dot{y}\left(t\right)+b_y\cdot u_{\mathcal{V}1_y}\left(t\right) \\ a_z\cdot{}^{\mathcal{V}1}\dot{z}\left(t\right)+b_z\cdot u_{\mathcal{V}1_z}\left(t\right) \\ a_\Psi\cdot{}^{\mathcal{V}1}\dot{\Psi}\left(t\right)+b_\Psi\cdot u_{\mathcal{V}1_\omega}\left(t\right) \end{bmatrix}. \tag{3.40}$$

The proposed approach is a trade-off between the continuous attitude description and the computational effort, regarding the quaternion approach (4 states) and the standard angle description (1 state).

3.6 Discretization of grey box models

A discretized model representation is required in order to implement the previously derived system functions in a simulator with an adaptable simulation step, for example *V-REP*. For this purpose, the discrete iterator k is introduced to describe the discrete time $t\left[k\right]=k\cdot\Delta t$. The corresponding states at these discrete time instances can be written accordingly $\boldsymbol{x}\left[k-1\right]\equiv\boldsymbol{x}\left(\left(k-1\right)\Delta t\right)$. A simple way of discretizing the derivative is the backward difference approximation

$$\dot{\boldsymbol{x}}\left(t\right)\approx\frac{\boldsymbol{x}\left[k\right]-\boldsymbol{x}\left[k-1\right]}{\Delta t} \tag{3.41}$$

which can be used to approximate the system function

$$\frac{\boldsymbol{x}\left[k\right]-\boldsymbol{x}\left[k-1\right]}{\Delta t}\approx\boldsymbol{f}\left(\boldsymbol{x}\left[k-1\right],\boldsymbol{u}\left[k-1\right]\right). \tag{3.42}$$

Isolating $\boldsymbol{x}\left[k\right]$ leads to

$$\boldsymbol{x}\left[k\right]=\boldsymbol{x}\left[k-1\right]+\Delta t\cdot\boldsymbol{f}\left(\boldsymbol{x}\left[k-1\right],\boldsymbol{u}\left[k-1\right]\right) \tag{3.43}$$

The discretization of the system function (3.24) can be accordingly used to calculate the next state at $t = \Delta t(k+1)$:

$$\boldsymbol{x}\left[k+1\right] = \begin{bmatrix} {}^{\mathcal{V}1}x\left[k\right] + \Delta t \left(x_{\mathcal{V}1}\left[k\right]\cos\left({}^{\mathcal{V}}\Psi\left[k\right]\right) - {}^{\mathcal{V}1}\dot{y}\left[k\right]\sin\left({}^{\mathcal{V}}\Psi\left[k\right]\right)\right) \\ {}^{\mathcal{V}1}y\left[k\right] + \Delta t \left({}^{\mathcal{V}1}\dot{x}\left[k\right]\sin\left({}^{\mathcal{V}}\Psi\left[k\right]\right) + {}^{\mathcal{V}1}\dot{y}\left[k\right]\cos\left({}^{\mathcal{V}}\Psi\left[k\right]\right)\right) \\ {}^{\mathcal{V}1}y\left[k\right] + \Delta t \cdot {}^{\mathcal{G}}\dot{z}\left[k\right] \\ \Psi_{\mathcal{V}1}\left[k\right] + \Delta t \cdot {}^{\mathcal{V}}\dot{\Psi}\left[k\right] \\ \left(a_x \Delta t + 1\right){}^{\mathcal{V}1}\dot{x}\left[k\right] + \Delta t \cdot b_x \cdot u_{\mathcal{V}1_x}\left[k\right] \\ \left(a_y \Delta t + 1\right){}^{\mathcal{V}1}\dot{y}\left[k\right] + \Delta t \cdot b_y \cdot u_{\mathcal{V}1_y}\left[k\right] \\ \left(a_z \Delta t + 1\right){}^{\mathcal{V}1}\dot{z}\left[k\right] + \Delta t \cdot b_z \cdot u_{\mathcal{V}1_z}\left[k\right] \\ \left(a_\Psi \Delta t + 1\right){}^{\mathcal{V}1}\dot{\Psi}\left[k\right] + \Delta t \cdot b_\Psi \cdot u_{\mathcal{V}1_\omega}\left[k\right] \end{bmatrix}. \tag{3.44}$$

For the direction vector extension, a normalization is required in order to fulfill the circle constraint (3.32). Accordingly the system state is calculated via

$$\boldsymbol{x}\left[k+1\right] = \begin{bmatrix} {}^{\mathcal{V}1}x\left[k\right] + \Delta t \left({}^{\mathcal{V}1}\dot{x}\left[k\right]{}^{\mathcal{G}}o_x\left[k\right] - {}^{\mathcal{V}1}\dot{y}\left[k\right]{}^{\mathcal{G}}o_y\left[k\right]\right) \\ {}^{\mathcal{V}1}y\left[k\right] + \Delta t \left({}^{\mathcal{V}1}\dot{x}\left[k\right]{}^{\mathcal{G}}o_y\left[k\right] + {}^{\mathcal{V}1}\dot{y}\left[k\right]{}^{\mathcal{G}}o_x\left[k\right]\right) \\ {}^{\mathcal{V}1}y\left[k\right] + \Delta t \cdot {}^{\mathcal{G}}\dot{z}\left[k\right] \\ \frac{{}^{\mathcal{G}}o_x\left[k\right] - \Delta t \cdot {}^{\mathcal{G}}o_y\left[k\right]{}^{\mathcal{V}}\dot{\Psi}\left[k\right]}{\sqrt{{}^{\mathcal{G}}o_x\left[k\right]^2 + {}^{\mathcal{G}}o_y\left[k\right]^2}} \\ \frac{{}^{\mathcal{G}}o_y\left[k\right] + \Delta t \cdot {}^{\mathcal{G}}o_x\left[k\right]{}^{\mathcal{V}}\dot{\Psi}\left[k\right]}{\sqrt{{}^{\mathcal{G}}o_x\left[k\right]^2 + {}^{\mathcal{G}}o_y\left[k\right]^2}} \\ \left(a_x \Delta t + 1\right){}^{\mathcal{V}1}\dot{x}\left[k\right] + \Delta t \cdot b_x \cdot u_{\mathcal{V}1_x}\left[k\right] \\ \left(a_y \Delta t + 1\right){}^{\mathcal{V}1}\dot{y}\left[k\right] + \Delta t \cdot b_y \cdot u_{\mathcal{V}1_y}\left[k\right] \\ \left(a_z \Delta t + 1\right){}^{\mathcal{V}1}\dot{z}\left[k\right] + \Delta t \cdot b_z \cdot u_{\mathcal{V}1_z}\left[k\right] \\ \left(a_\Psi \Delta t + 1\right){}^{\mathcal{V}1}\dot{\Psi}\left[k\right] + \Delta t \cdot b_\Psi \cdot u_{\mathcal{V}1_\omega}\left[k\right] \end{bmatrix}. \tag{3.45}$$

3.7 Identification

In order to use the *HMDV* (3.24), the model parameters (3.25) have to be identified. Due to the characteristic of the inherent *PT1*s, these parameters can directly be determined from the step response. For this purpose, the *UAV*'s position and velocity response have to be measured. Due to the separation in forward, leftward, upward and heading channel, the parameters for each subsystem can be identified separately.

In the context of this work, *AR.Drone 2.0* and *DJI M100* quadrotors are used. The corresponding model parameters are identified in the following.

3.7.1 AR.Drone 2.0 identification

This section is dedicated to the identification of the *HMDV* (3.24) parameters for a real *AR.Drone 2.0* quadrotor. The utilized *AR.Drone 2.0* is shown in Figure 3.7. The grey balls are visual markers which are used to triangulate the robots pose with the help of a motion capture system (*OPTITRACK*). The pose measurement is running at $\Delta t = \frac{1}{240\,\mathrm{Hz}}$.

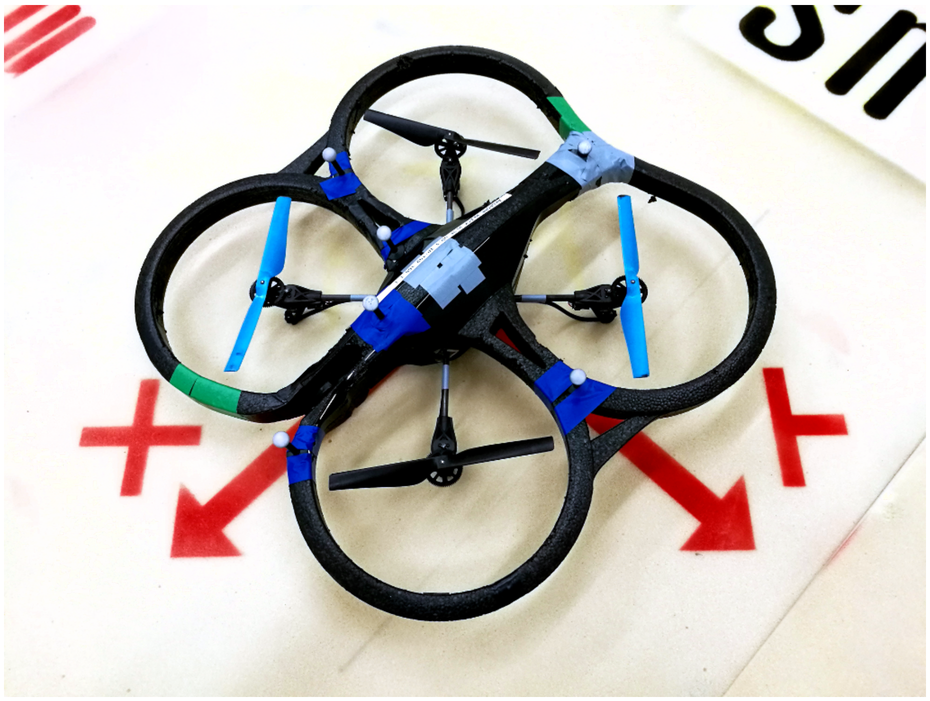


Figure 3.7: AR.Drone 2.0 photography

All *AR.Drone 2.0* inputs $\boldsymbol{u}$ (3.20) are limited to $\left\|u_{\mathcal{V}1_x}\right| \leq 1, \left\|u_{\mathcal{V}1_y}\right| \leq 1, \left\|u_{\mathcal{V}1_z}\right| \leq 1, \left\|u_{\mathcal{V}1_\omega}\right| \leq 1$. For the identification, input steps between $[-1, 0, 1]$ are used to impose the maximal actuation from the stationary state. The *AR.Drone 2.0* velocity response is measured and processed in the following four steps:

1 : The global pose is measured by the *OPTITRACK* system.

2 : The global velocities are approximated via a difference quotient:

$$\begin{bmatrix} {}^{\mathcal{G}}\dot{x}\,(t) \\ {}^{\mathcal{G}}\dot{y}\,(t) \\ {}^{\mathcal{G}}\dot{z}\,(t) \\ {}^{\mathcal{G}}\dot{\Psi}\,(t) \end{bmatrix} \approx \frac{1}{t\,[k] - t\,[k-1]} \begin{bmatrix} {}^{\mathcal{G}}x\,[k] - {}^{\mathcal{G}}x\,[k] \\ {}^{\mathcal{G}}y\,[k] - {}^{\mathcal{G}}y\,[k] \\ {}^{\mathcal{G}}z\,[k] - {}^{\mathcal{G}}z\,[k] \\ {}^{\mathcal{G}}\Psi\,[k] - {}^{\mathcal{G}}\Psi\,[k] \end{bmatrix} \tag{3.46}$$

3 : The determined velocities are filtered with a moving average filter considering the last $N = 10\,samples$.

$$\begin{bmatrix} {}^{\mathcal{G}}\dot{x}(t) \\ {}^{\mathcal{G}}\dot{y}(t) \\ {}^{\mathcal{G}}\dot{z}(t) \\ {}^{\mathcal{G}}\dot{\Psi}(t) \end{bmatrix} \approx \frac{1}{N}\sum_{l=0}^{N-1}\frac{1}{t[k]-t[k-1]}\begin{bmatrix} {}^{\mathcal{G}}x[k-l] - {}^{\mathcal{G}}x[k-l] \\ {}^{\mathcal{G}}y[k-l] - {}^{\mathcal{G}}y[k-l] \\ {}^{\mathcal{G}}z[k-l] - {}^{\mathcal{G}}z[k-l] \\ {}^{\mathcal{G}}\Psi[k-l] - {}^{\mathcal{G}}\Psi[k-l] \end{bmatrix} \tag{3.47}$$

4 : The resulting filtered velocity data is transformed into the vehicle 1 frame $\mathcal{V}1$ using

$$\begin{bmatrix} {}^{\mathcal{V}1}\dot{x}(t) \\ {}^{\mathcal{V}1}\dot{y}(t) \\ {}^{\mathcal{V}1}\dot{z}(t) \\ {}^{\mathcal{V}1}\dot{\Psi}(t) \end{bmatrix} = \left[\begin{array}{ccc|c} & & & 0 \\ & {}^{\mathcal{V}1}\boldsymbol{R}_{\mathcal{G}}\left({}^{\mathcal{V}}\Psi[k]\right) & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{\mathcal{G}}x(t) \\ {}^{\mathcal{G}}y(t) \\ {}^{\mathcal{G}}z(t) \\ {}^{\mathcal{G}}\Psi(t) \end{bmatrix}. \tag{3.48}$$

Due to disturbance, it is advantageous to base the identification of model parameters on a longer interval, including multiple input steps. This is facilitated by identification tools which allow automatic fitting of the model with the real *UAV* behavior. In the context of this thesis, the *MATLAB* algorithms *SSREGEST*, *SSEST* and *N4SID* have been used for this purpose. The best fit trajectories of the first-order approximations are given in Figure 3.8. Based on the automated estimates, an additional manual fitting of the step functions is given under the reference "Manual". The corresponding best fit parameters are given in Table 3.1.

	a_x	b_x	Best Fit ${}^{\mathcal{V}1}\dot{x}$	a_y	b_y	Best Fit ${}^{\mathcal{V}1}\dot{y}$
SSREGEST	-1.225	3.970	43.10%	-1.266	3.839	47.75%
SSEST	-0.526	1.878	46.42%	-1.087	2.603	59.65%
N4SID	-0.523	1.877	46.37%	-2.083	3.757	48.36%
Manual	-0.800	2.560	52.21%	-0.800	2.560	54.90%
	a_z	b_z	Best Fit ${}^{\mathcal{V}1}\dot{z}$	a_Ψ	b_Ψ	Best Fit ${}^{\mathcal{V}1}\dot{\Psi}$
SSREGEST	-1.485	1.796	51.21%	-7.519	15.90	69.01%
SSEST	-1.287	1.644	44.26%	-4.669	7.816	91.99%
N4SID	-1.259	1.633	42.89%	-4.617	7.737	91.95%
Manual	-1.900	2.100	62.37%	-5.000	9.000	89.01%

Table 3.1: Identification parameters to match *AR.Drone 2.0* velocity channels

As the identified *AR.Drone 2.0* parameters have shown to be prone to hull and rotor deformations, the identification has been conducted with a "new" unused *AR.Drone 2.0*. To be able to directly relate velocity and position response for the horizontal and vertical behavior, ${}^{\mathcal{V}1}\Psi$ is manually kept close to zero.

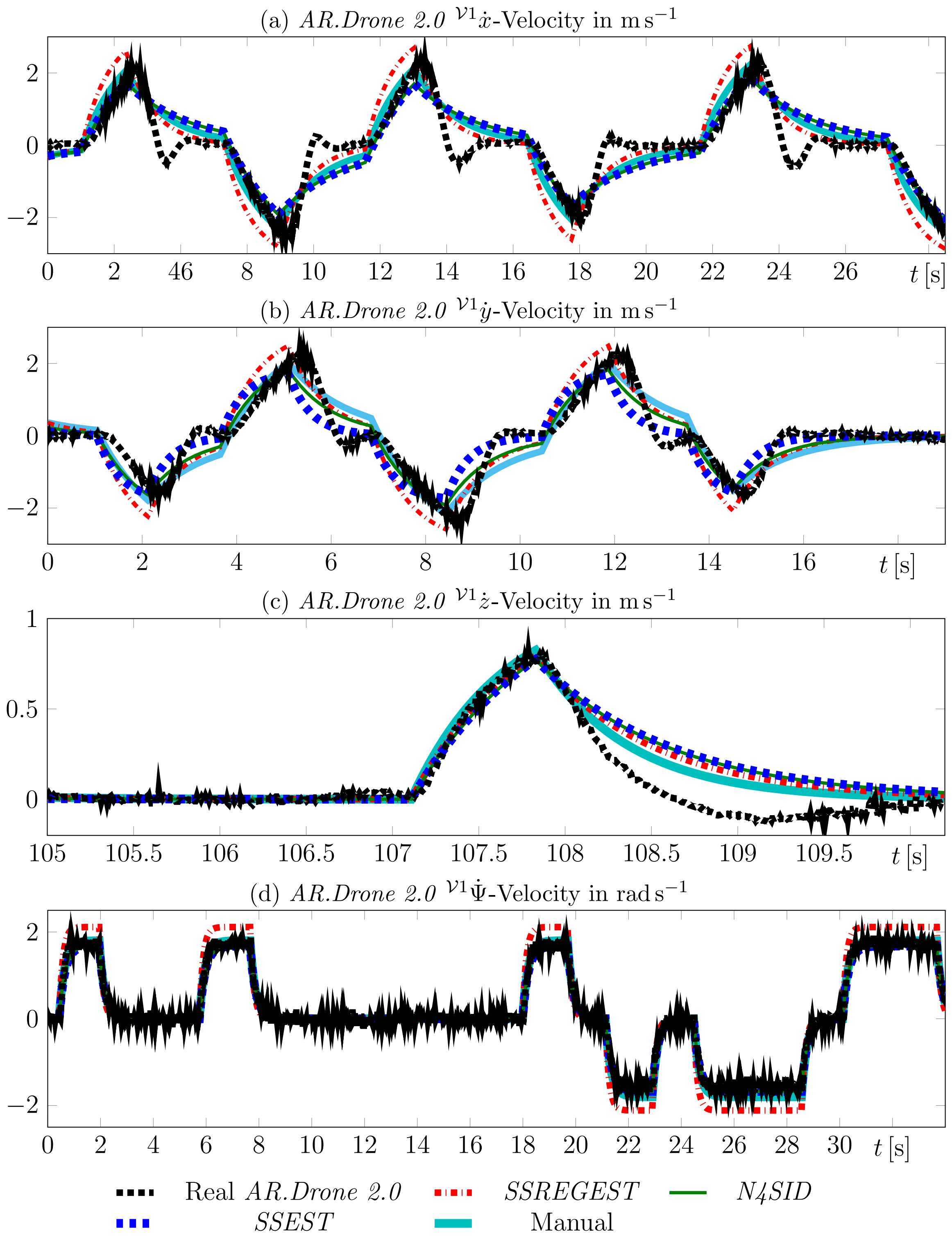


Figure 3.8: Automized *AR.Drone 2.0* identification

The resulting trajectories show, that only the ${}^{\mathcal{V}1}\dot{\Psi}$ behavior in Figure 3.8d appears to comply with a *PT1* approximation. Figure 3.8c indicates a precise fitting of ${}^{\mathcal{V}1}\dot{z}$ on an input step, but a slight overshoot for a step back to zero. This nonlinearity therefore leads to a compulsory *PT1* model error. The same behavior is even more visible in the ${}^{\mathcal{V}1}\dot{x}$ trajectory in Figure 3.8a and the ${}^{\mathcal{V}1}\dot{y}$ plot in Figure 3.8b. The observed nonlinearities are a result of the *AR.Drone 2.0* dynamics with its internal control and are system specific. Their occurence is also directly reflected by the best fit indicators in Table 3.1.

The *AR.Drone 2.0* shows very similar velocity behavior in forward (Figure 3.8a) and leftward direction (Figure 3.8b). Nevertheless, the automatically identified parameters a_x, b_x in Table 3.1 are differing significantly. This is evidence, that the fitting of the nonlinearities is not trivial. The major question is therefore which approximation serves the modeling purpose.

In the context of this thesis, the *UAV* models have been derived to serve as prediction model in *MPC*. The effect of the model error in a prediction model for a *UAV* with stabilizing internal control yields:

- Real system slower than prediction model $\rightarrow$ control response is lower than nominal $\rightarrow$ slow maneuvering
- Real system faster than prediction model $\rightarrow$ control response is higher than nominal $\rightarrow$ aggressive maneuvering which might lead to stability problems

For the first step in Figure 3.8a, it is visible that the model is faster than the system in the upslope. For the downslope the opposite is true. One approach would therefore be to increase the *PT1* settling times to achieve a very slow response and thus accept a low control performance.

To avoid this performance loss, the fitting problem is addressed by empirically adjusting the model parameters in a closed-loop experiment. This work is focusing on model predictive *UAV* position and heading control. Hence, the parameters have been identified in a pose tracking scenario with a real *AR.Drone 2.0* (see §5.3.1). Starting with the initial guess of the manually fitted parameters from Table 3.1 the closed-loop position response showed undesired low damping in ${}^{\mathcal{G}}x$ and ${}^{\mathcal{G}}y$. Consequently, a_x, b_x, a_y and b_y of the forward and sideward channel have been modified systematically in order to achieve a fast, damped position and heading response. As a remark, the damping of the closed-loop system is not only depending on the quality of the *MPC* prediction model, but also on the control parametrization. The control parametrization described in §5.3.1 and the adjustment of the model parameters are therefore an iterative process. The resulting model parameters are given in Table 3.2.

The corresponding real *AR.Drone 2.0* and model (indicated by $\widehat{\ }$) trajectories are given in Figure 3.9-3.12. The resulting ${}^{\mathcal{G}}x$ and ${}^{\mathcal{G}}y$ trajectories show a similar position displace-

	a_x	b_x	a_y	b_y	a_z	b_z	a_Ψ	b_Ψ
Manual	-0.6000	1.200	-0.600	1.200	-1.900	2.100	-5.000	9.000

Table 3.2: Empirically chosen parameters to match *AR.Drone 2.0* position channels

ment after a given input pulse. However, the corresponding velocity (Figure 3.9b, 3.10b) approximation is suboptimal. The velocity and position for upward (Figure 3.11) and heading channel (Figure 3.12) do show the desired fitting. In Figure 3.11a, the slight decrease in the ${}^{\mathcal{G}}z$ position after the initial step is caused by a horizontal movement command, as the vertical and horizontal movement of the *AR.Drone 2.0* are not completely decoupled. The parametrized *AR.Drone 2.0 HMDV* model finally results to

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} {}^{\mathcal{V}1}\dot{x}(t)\,{}^{\mathcal{G}}o_x(t) - {}^{\mathcal{V}1}\dot{y}(t)\,{}^{\mathcal{G}}o_y(t) \\ {}^{\mathcal{V}1}\dot{x}(t)\,{}^{\mathcal{G}}o_y(t) + {}^{\mathcal{V}1}\dot{y}(t)\,{}^{\mathcal{G}}o_x(t) \\ {}^{\mathcal{G}}\dot{z}(t) \\ -{}^{\mathcal{G}}o_y(t)\cdot{}^{\mathcal{V}}\dot{\Psi}(t) \\ {}^{\mathcal{G}}o_x(t)\cdot{}^{\mathcal{V}}\dot{\Psi}(t) \\ -0.600\cdot{}^{\mathcal{V}1}\dot{x}(t) + 1.2\cdot u_{\mathcal{V}1_x}(t) \\ -0.600\cdot{}^{\mathcal{V}1}\dot{y}(t) + 1.2\cdot u_{\mathcal{V}1_y}(t) \\ -1.900\cdot{}^{\mathcal{V}1}\dot{z}(t) + 2.1\cdot u_{\mathcal{V}1_z}(t) \\ -5.000\cdot{}^{\mathcal{V}1}\dot{\Psi}(t) + 9.000\cdot u_{\mathcal{V}1_\omega}(t) \end{bmatrix}. \tag{3.49}$$

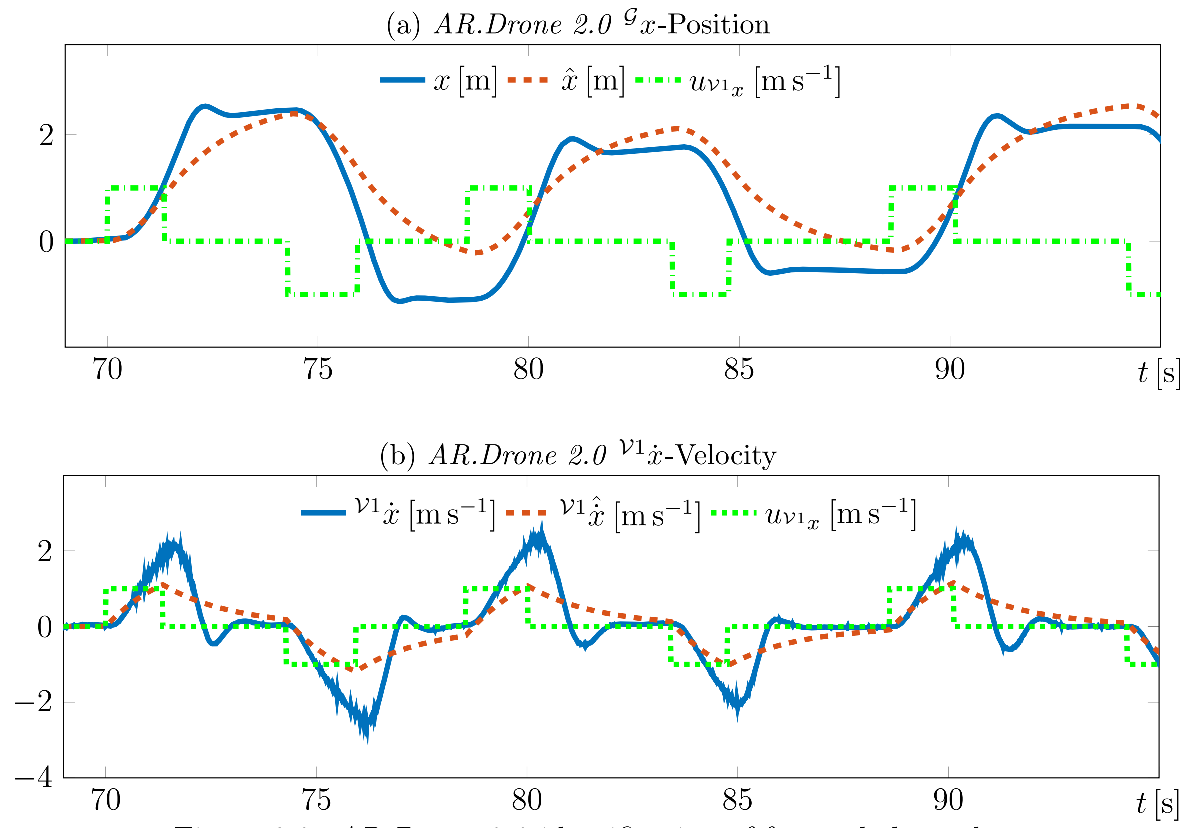


Figure 3.9: *AR.Drone 2.0* identification of forward channel

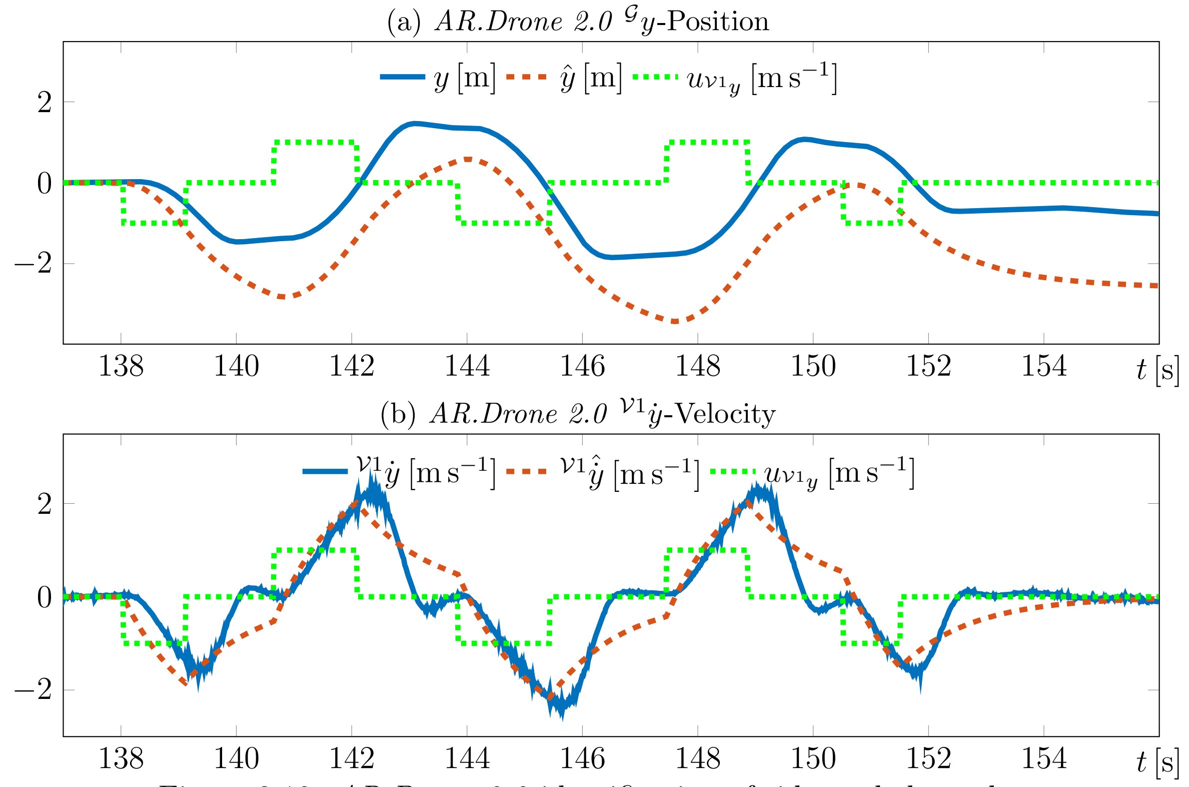


Figure 3.10: *AR.Drone 2.0* identification of sideward channel

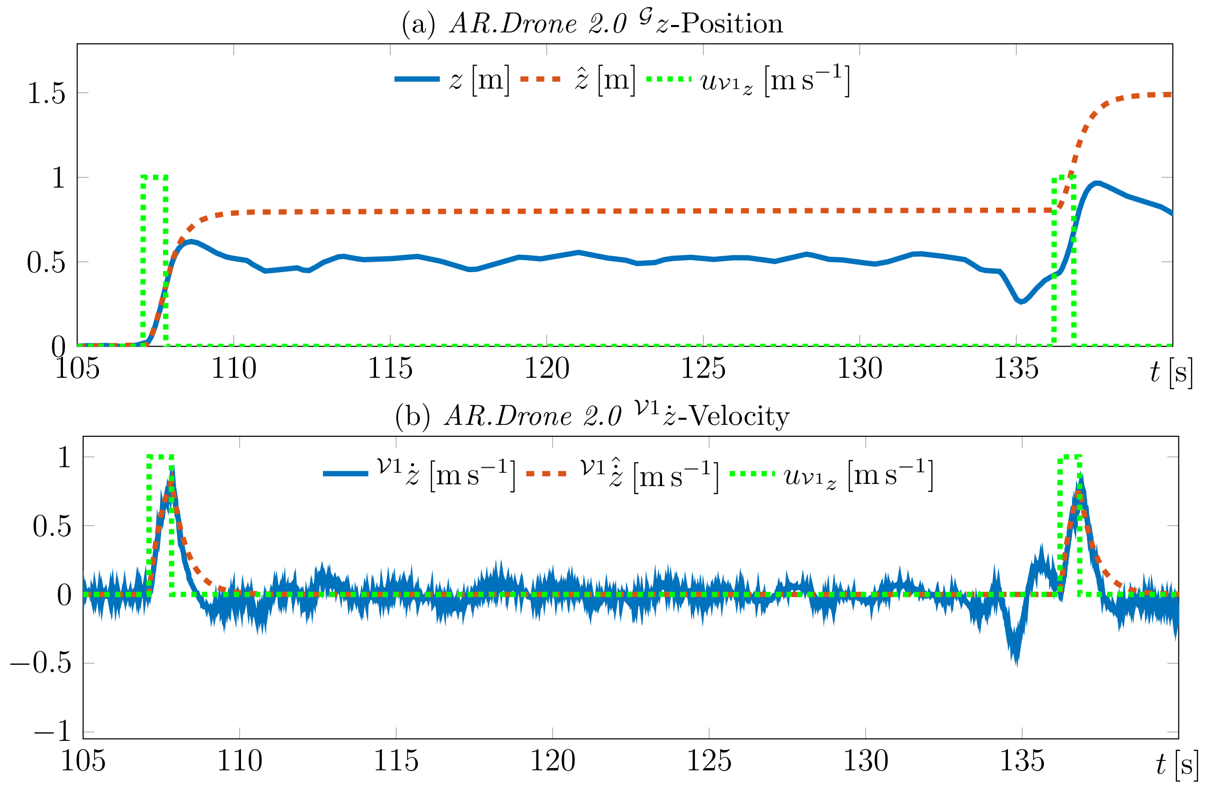


Figure 3.11: *AR.Drone 2.0* identification of upward channel

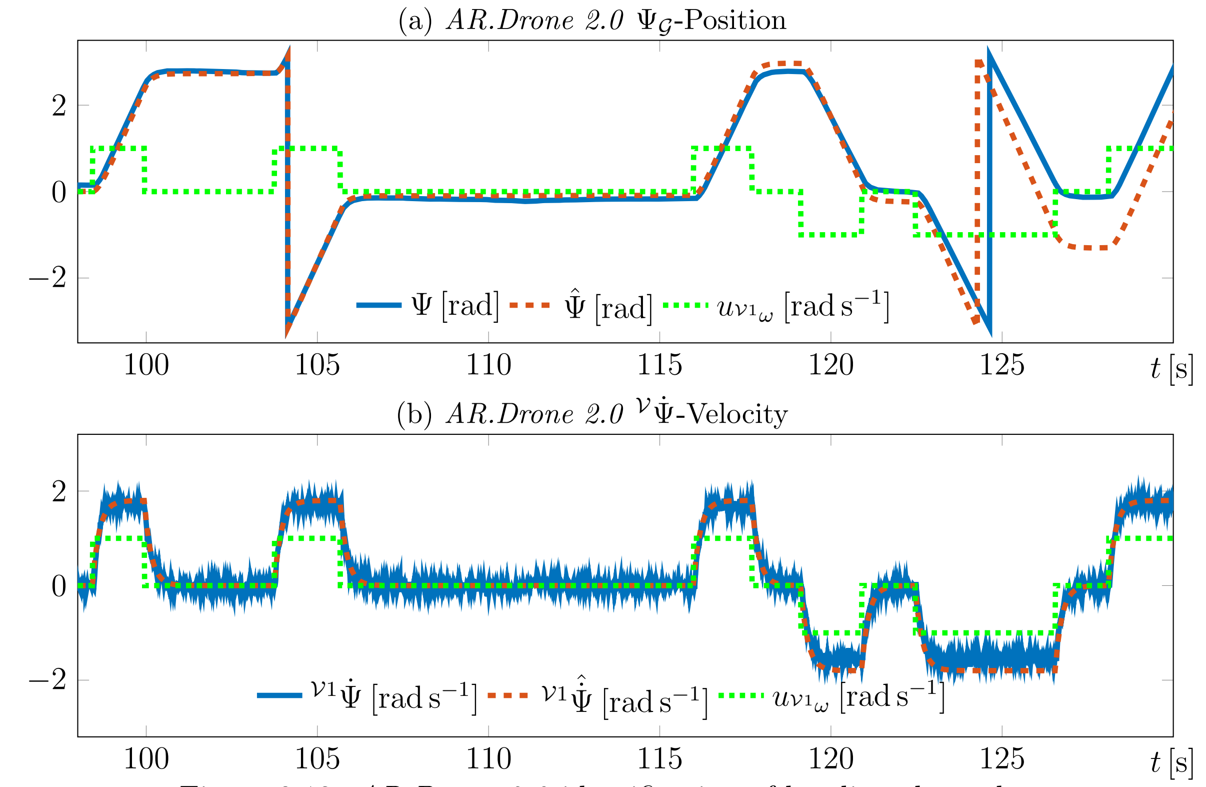


Figure 3.12: *AR.Drone 2.0* identification of heading channel

3.7.2 DJI M100 identification

The second type of *UAV* considered within this thesis is a *DJI M100* quadrotor (shown in Figure 3.13 with attached robotic arm).

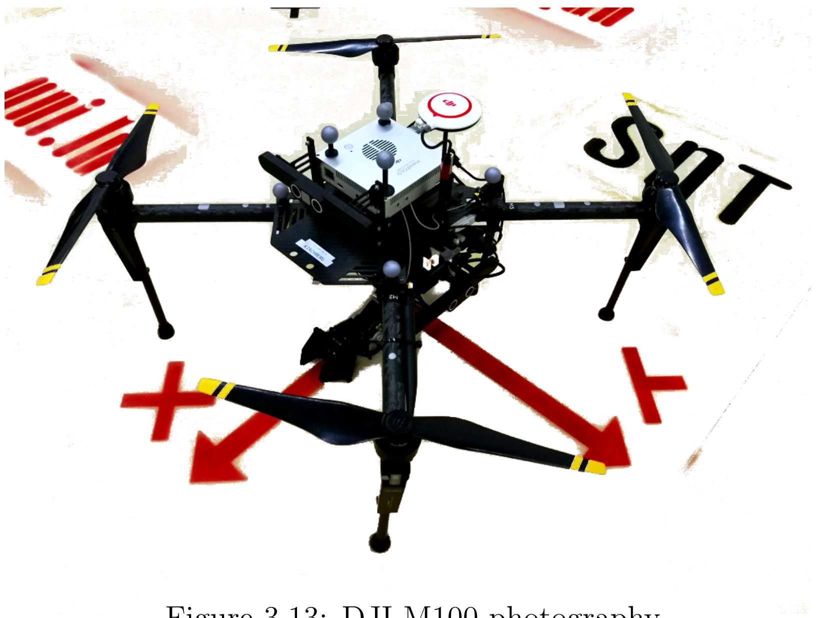


Figure 3.13: DJI M100 photography

Its physical properties are given in [Ltd]. In *ROS* teleoperated mode, the lateral velocities are limited to

$$u_{\nu 1_{x,orig}} := \{u_{\nu 1_{x,orig}} \in \mathbb{R} | -10\,\mathrm{m\,s^{-1}} < u_{\nu 1_{x,orig}} \leq 10\,\mathrm{m\,s^{-1}}\} \tag{3.50}$$
$$u_{\nu 1_{y,orig}} := \{u_{\nu 1_{y,orig}} \in \mathbb{R} | -10\,\mathrm{m\,s^{-1}} < u_{\nu 1_{y,orig}} \leq 10\,\mathrm{m\,s^{-1}}\} \tag{3.51}$$
$$u_{\nu 1_{z,orig}} := \{u_{\nu 1_{z,orig}} \in \mathbb{R} | -4\,\mathrm{m\,s^{-1}} < u_{\nu 1_{z,orig}} \leq 4\,\mathrm{m\,s^{-1}}\} \tag{3.52}$$
$$u_{\nu 1_{\omega,orig}} := \{u_{\nu 1_{\omega,orig}} \in \mathbb{R} | -100\,^{\circ}\,\mathrm{s^{-1}} < u_{\nu 1_{\omega,orig}} \leq 100\,^{\circ}\,\mathrm{s^{-1}}\}. \tag{3.53}$$

For the model and its identification, the inputs are normalized. The original inputs (index: orig) are mapped to the model inputs according to

$$u_{\nu 1_x} = \frac{u_{\nu 1_{x,orig}}}{10} \qquad u_{\nu 1_y} = \frac{u_{\nu 1_{y,orig}}}{10} \qquad u_{\nu 1_z} = \frac{u_{\nu 1_{z,orig}}}{4} \qquad u_{\nu 1_\omega} = \frac{u_{\nu 1_{\omega,orig}}}{100} \tag{3.54}$$

The *DJI M100*'s fast dynamics and in combination with the spatial constraints of the available *OPTITRACK* motion tracking system are problematic. As a consequnce, no experimental identification with full velocity input steps has been possible. However, *DJI* provides a simulator with an accurate data-driven behavior model of the *DJI M100*. This data-driven model of the simulator can be matched with the the velocity reference model (3.40). For this purpose, the same signal processing is used as for the *AR.Drone 2.0* in

§3.7.1. The pose measurement of the motion capture system is thereby substituted by the simulator data. In the context of the identification, the communication latency is neglected. As the *DJI M100*'s internal controller can cope with minor modifications in the *UAV*'s architecture, the resulting velocity reference model is sufficiently precise to serve as *MPC* prediction model. The model parameters have been empirically chosen as shown in Table 3.3.

The corresponding response is given in Figure 3.14-3.17. As in §3.7.1, also here ${}^{\mathcal{V}1}\Psi$ is kept close to zero during the identification of the horizontal and vertical movement. This allows the direct relation between ${}^{\mathcal{G}}x$ (Figure 3.14a) with ${}^{\mathcal{V}1}\dot{x}$ (Figure 3.14b) as well as ${}^{\mathcal{G}}y$ (Figure 3.15a) with ${}^{\mathcal{V}1}\dot{y}$ (Figure 3.15b). The response to the altitude input signal is given in Figure 3.16 and for the heading input in Figure 3.15. The $\widehat{}$ symbol is hereby indicating the model response, while ${}^{\mathcal{G}}x\,(t)$, ${}^{\mathcal{G}}y\,(t)$, ${}^{\mathcal{G}}z\,(t)$ and ${}^{\mathcal{G}}\Psi\,(t)$ are the *DJI M100* simulator response. Particularly the velocity plots show the matching of the developed model with the original simulator model. The position plots show a difference in their amplitude which is caused by the integration of the modeling error of the velocity channels. One particularity is ${}^{\mathcal{V}1}\dot{z}$ in Figure 3.16b which states a generally higher upward than downward velocity. Despite the good fit of model and simulator data, the model quality regarding a real *DJI M100* has to be determined experimentally. For this reason, the developed model is used as prediction model within an *MPC* pose tracking scenario in §5.3.2. The closed-loop behavior is then compared to the same scenario using the developed model to simulate the *DJI M100* behavior. This allows the validatation of the quality of the parametrized *DJI M100 HMDV*.

	a_x	b_x	a_y	b_y	a_z	b_z	a_Ψ	b_Ψ
Model	-1.0	10.0	-1.0	10.0	-2.0	8.0	-5.0	8.0

Table 3.3: Empirically chosen parameters to match *DJI* position channels

According to the determined parameters given in Table 3.3, the *DJI M100 HMDV* model results to

$$\boldsymbol{f}\,(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} {}^{\mathcal{V}1}\dot{x}\,(t)\,{}^{\mathcal{G}}o_x\,(t) - {}^{\mathcal{V}1}\dot{y}\,(t)\,{}^{\mathcal{G}}o_y\,(t) \\ {}^{\mathcal{V}1}\dot{x}\,(t)\,{}^{\mathcal{G}}o_y\,(t) + {}^{\mathcal{V}1}\dot{y}\,(t)\,{}^{\mathcal{G}}o_x\,(t) \\ {}^{\mathcal{G}}\dot{z}\,(t) \\ -{}^{\mathcal{G}}o_y\,(t)\cdot{}^{\mathcal{V}}\dot{\Psi}\,(t) \\ {}^{\mathcal{G}}o_x\,(t)\cdot{}^{\mathcal{V}}\dot{\Psi}\,(t) \\ -1.0\cdot{}^{\mathcal{V}1}\dot{x}\,(t) + 10.0\cdot u_{\mathcal{V}1_x}\,(t) \\ -1.0\cdot{}^{\mathcal{V}1}\dot{y}\,(t) + 10.0\cdot u_{\mathcal{V}1_y}\,(t) \\ -2.0\cdot{}^{\mathcal{V}1}\dot{z}\,(t) + 8.0\cdot u_{\mathcal{V}1_z}\,(t) \\ -5.0\cdot{}^{\mathcal{V}1}\dot{\Psi}\,(t) + 8.0\cdot u_{\mathcal{V}1_\omega}\,(t) \end{bmatrix}. \tag{3.55}$$

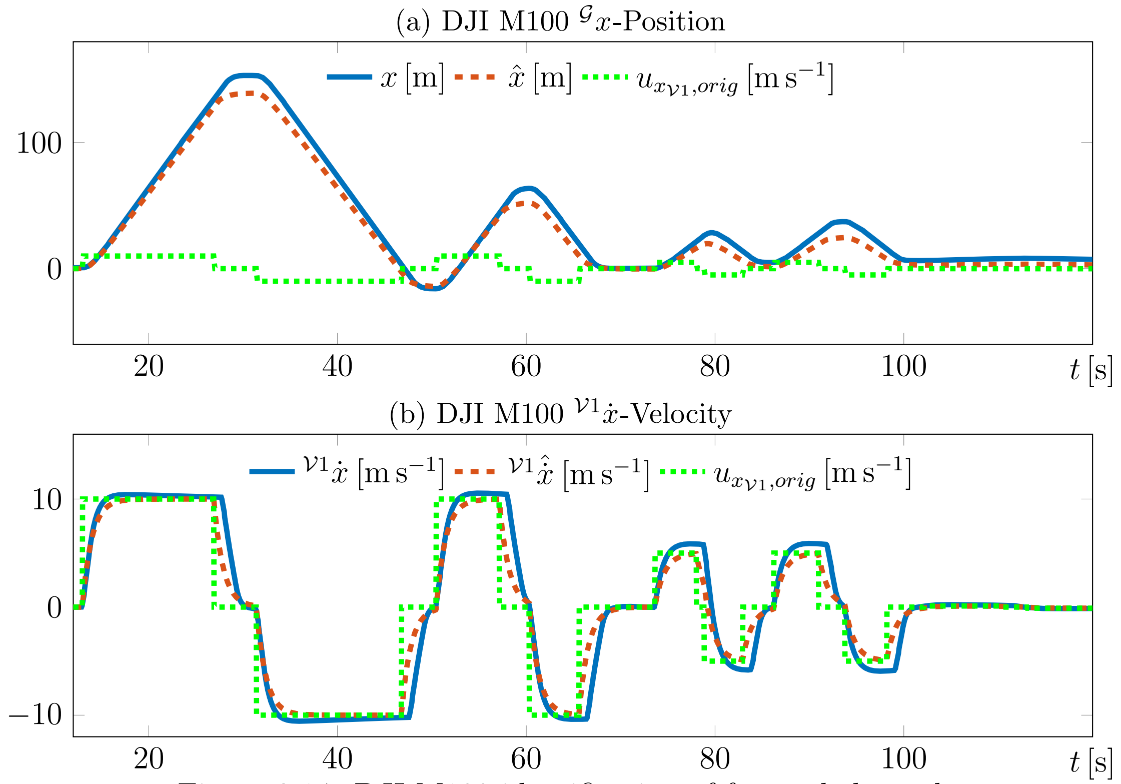


Figure 3.14: DJI M100 identification of forward channel

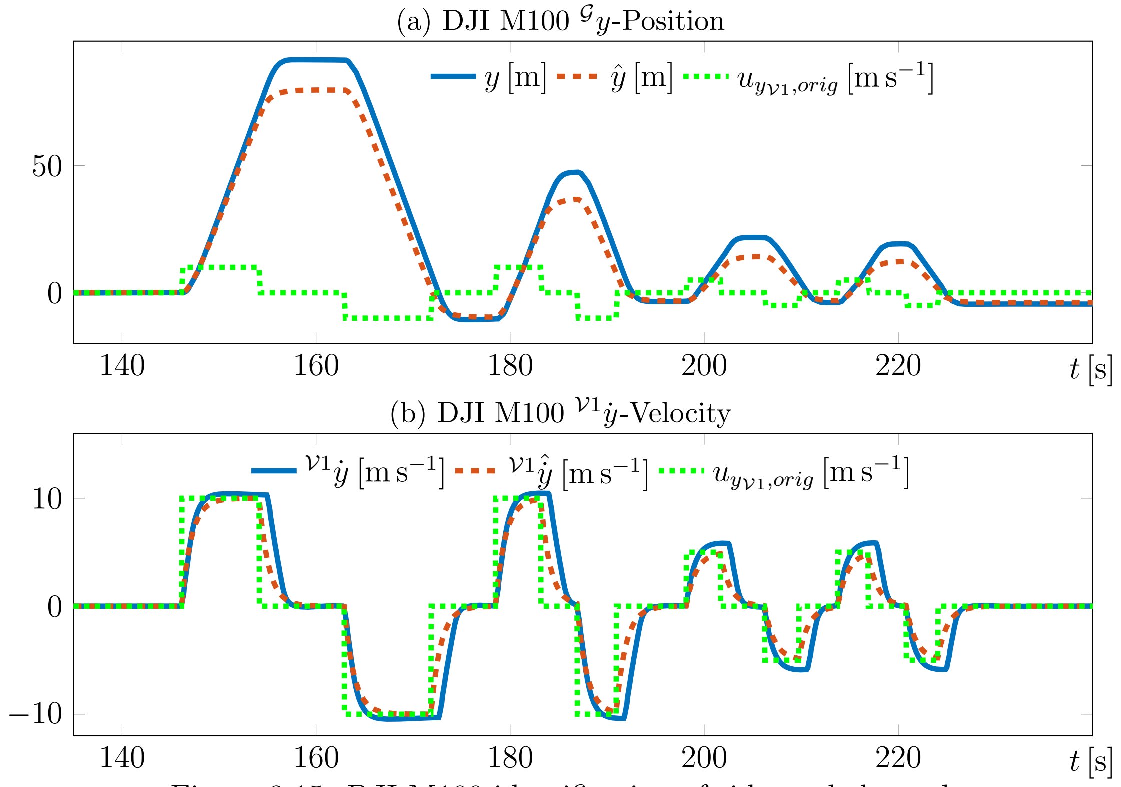


Figure 3.15: DJI M100 identification of sideward channel

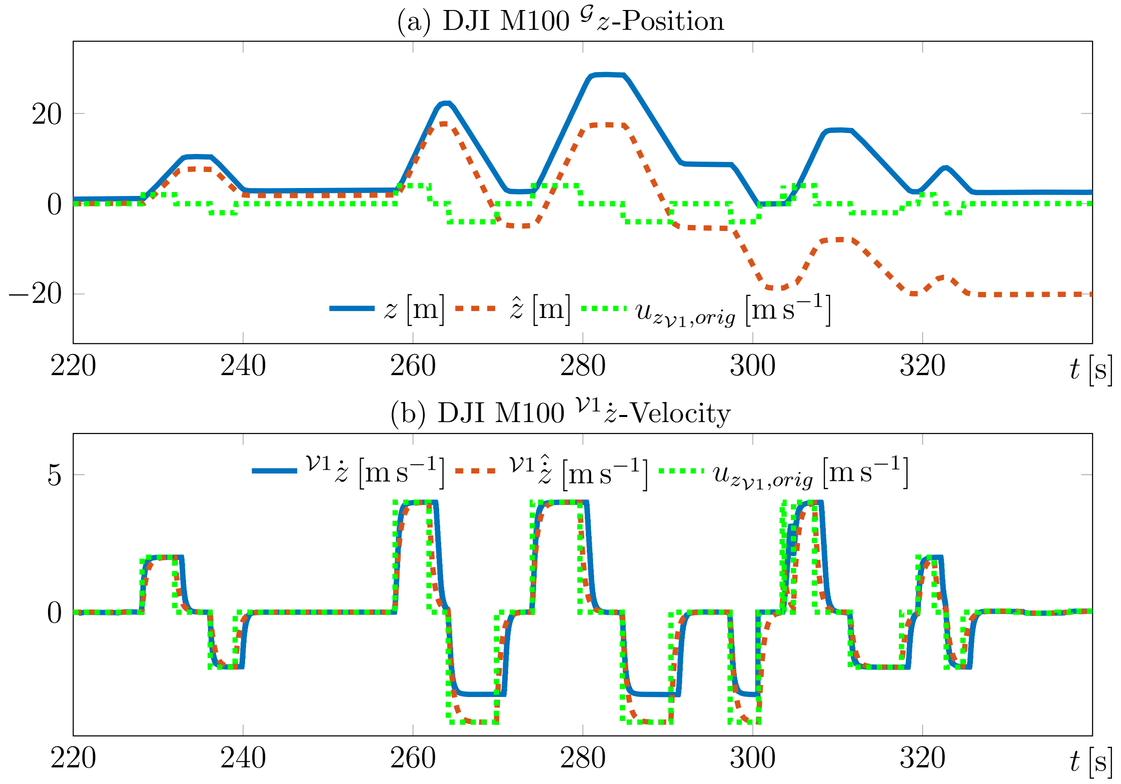


Figure 3.16: DJI M100 identification of upward channel

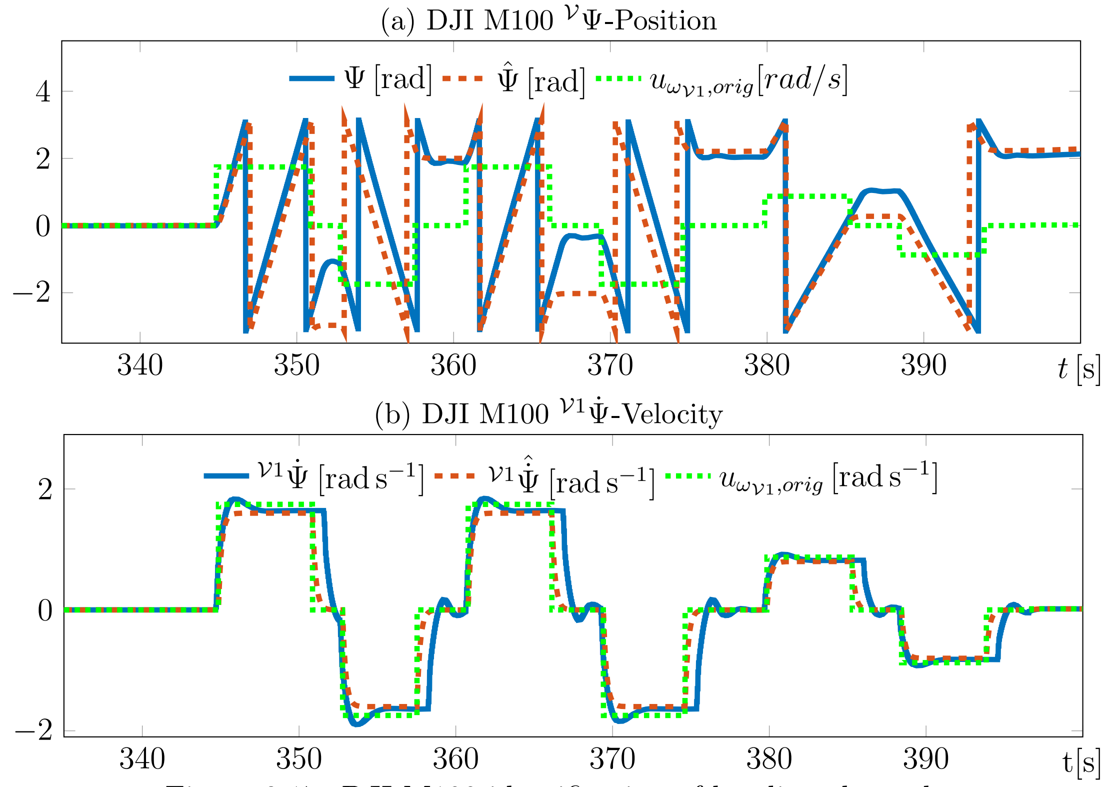


Figure 3.17: DJI M100 identification of heading channel

3.8 Conclusion

This chapter has presented the modeling of a quadrotor *UAV* system. This includes a traditional physical modeling approach with rotor velocity inputs, respectively torque and thrust inputs. To describe the *UAV* attitude without Euler-angle singularities, the physical model is extended using a quaternion attitude description.

The main contribution of this chapter is the development and identification of a kinematic model which allows describing the behavior of commercial quadrotors with unknown internal controller and physical parameters (e.g. inertia). The idea of this model is an operation of the *UAV* close to its stationary hover attitude. As a result, the heading ${}^{\mathcal{V}}\Psi$ suffices to describe its attitude. To describe the input response of the *UAV*, the settling behavior of the forward, sideward, upward and heading velocities have been approximated with a *PT1*. The resulting hover model with ${}^{\mathcal{V}}\Psi$ description (*HMDV*) has inherited the property of the Euler angle singularity. This has been addressed by substituting the ${}^{\mathcal{V}}\Psi$ with a direction vector attitude description (*HMDV*). To address *MPC* in different computational environments, the reduced models *RHMY* and *RHMDV* have been presented. These are using less states and thus memory. Finally, the model parameters have been identified for a real *AR.Drone 2.0* and *DJI M100* system. While the identification of the *AR.Drone 2.0* has been conducted via a motion capture system, the higher velocity and limited laboratory space allowed the identification of the *DJI M100* only from the vendor's data-driven simulator. Nevertheless, the closed-loop behavior of both models is validated in §5, discussing *UAV* control.

Regarding the *AR.Drone 2.0*, the identification data revealed nonlinearities which cannot be fully accommodated by the proposed *PT1* description. Furthermore, the *AR.Drone 2.0* behavior has shown to be prone to hull and rotor deformations. For this reasons, some experiments within this thesis have been conducted with a set of deprecated model parameters. In these cases, the utilized parameters are explicitly given for the experimental setup. The parameters presented in §3.7.1 have been identified using undeformed *UAV*s.

The advantage of the presented hover model for commercial *UAV*s is its compact state-vector description and low amount of mathematical operations. This is advantageous in terms of memory and computation time and particularly important if it is serving as *MPC* prediction model on embedded computers. On systems with sufficient computational resources, it might be advantageous to use a full quaternion attitude description instead of the direction vector approach. This could increase the dynamic performance, particularly regarding the control of aerial manipulators. Such a full attitude description requires the identification of Θ and Φ behavior and can be developed straight-forward using the mapping (2.24), (3.17). However, depending on the *UAV*'s internal attitude control, the *PT1* approximation for Θ and Φ velocities might be not very accurate. This can be addressed using higher order models or a nonlinear approximation. Hence, a full quaternion model for commercial UAVs has been omitted in the context of this work in order to limit the scope of this work and to maintain a generic model.

Chapter 4

Nonlinear model predictive control

This chapter is dedicated to fast Nonlinear Model Predictive Control (*NMPC*) approaches. Model Predictive Control (*MPC*) is the use of optimization techniques for closed-loop control. The basic idea of *MPC* is to back-propagate the influence of a plant's inputs $\boldsymbol{u}$ on the predicted future behavior in order to determine the current optimal input $\boldsymbol{u}^{*}(t)$ for a desired objective e.g. a desired trajectory $\underline{\boldsymbol{x}}_{des}(\tau)$, a minimal response time, etc. In order to predict the plant's future state $\underline{\boldsymbol{x}}(\tau) \in \mathbb{R}^{n_x}$, *NMPC* is using a nonlinear model of the plant dynamics $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}^{n_x}$. For real-time applications, the considered prediction horizon length $T \in \mathbb{R}^{+}$ is typically limited, due to the resulting computational burden and the limited precision of prediction models. As a result, the time variable for the prediction horizon at time instance $t \in \mathbb{R}$ is defined on the interval $\tau = \{\tau \in \mathbb{R} | t \leq \tau \leq t+T\}$. According to the convention defined in §2, the variables evolving over this horizon are indicated with bold and underlined symbols e.g. $\underline{\boldsymbol{x}}$.

As for controllable systems $\underline{\boldsymbol{x}}(\tau)$ is dependent on $\underline{\boldsymbol{u}}(\tau)$, the prediction $\underline{\boldsymbol{x}}(\tau)$ and computation of $\underline{\boldsymbol{u}}(\tau)$ is an iterative process. In order to determine the optimal controls $\boldsymbol{u}^{*}(t)$, it is necessary to define which behavior is considered to be optimal. At time instance t this can be described in form of an optimal control problem

$$\min_{\underline{\boldsymbol{u}}(\cdot)} \quad J(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}) = V(\underline{\boldsymbol{x}}(t+T), t+T) + \int_{t}^{t+T} L(\underline{\boldsymbol{x}}(\tau), \underline{\boldsymbol{u}}(\tau), \tau) \, \mathrm{d}\tau \tag{4.1}$$

$$\text{s.t.} \quad \dot{\underline{\boldsymbol{x}}}(\tau) = \boldsymbol{f}(\underline{\boldsymbol{x}}(\tau), \underline{\boldsymbol{u}}(\tau), \tau) \tag{4.2}$$

$$\mathbf{0} = \boldsymbol{c_{eq}}(\underline{\boldsymbol{x}}(\tau), \underline{\boldsymbol{u}}(\tau), \tau) \tag{4.3}$$

$$\mathbf{0} \geq \boldsymbol{c_{in}}(\underline{\boldsymbol{x}}(\tau), \underline{\boldsymbol{u}}(\tau), \tau) \tag{4.4}$$

$$\boldsymbol{x}(t) = \underline{\boldsymbol{x}}(\tau_0), \tag{4.5}$$

where a performance functional $J(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ (4.1) in Bolza-form is minimized. This means that the minimization is considering a stage cost functional L which is integrated over the horizon, as well as a terminal cost functional V which appears at the end of the horizon. To take into account the plant dynamics, the minimization of the *OCP* is subject to the system dynamics $\boldsymbol{f}(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau)$ given by (4.2). The *OCP* can express further restrictions in terms of equality (4.3) and inequality constraints (4.4). A typical example for such constraints are for example the input limitations of a plant. For the feedback of the plant's behavior at each time instance t, the state prediction is initiated with the real system state $\boldsymbol{x}(t)$ (4.5). To finally close the loop, $\underline{\boldsymbol{u}}$ is computed to solve *OCP* (4.1)-(4.5) and applied to the real plant. Within the context of this work the standard control policy is applied:

$$\underline{\boldsymbol{u}}(\tau_0 = t) = \boldsymbol{u}(t). \tag{4.6}$$

The technical implementation of *MPC* requires a discretization of time $t_k = \Delta t \cdot k$ and the horizon $\tau_k[j] = t_k + \Delta\tau \cdot j$ at each time instance t_k. This leads to the characteristic receding horizon of *MPC*, as illustrated in Figure 4.1.

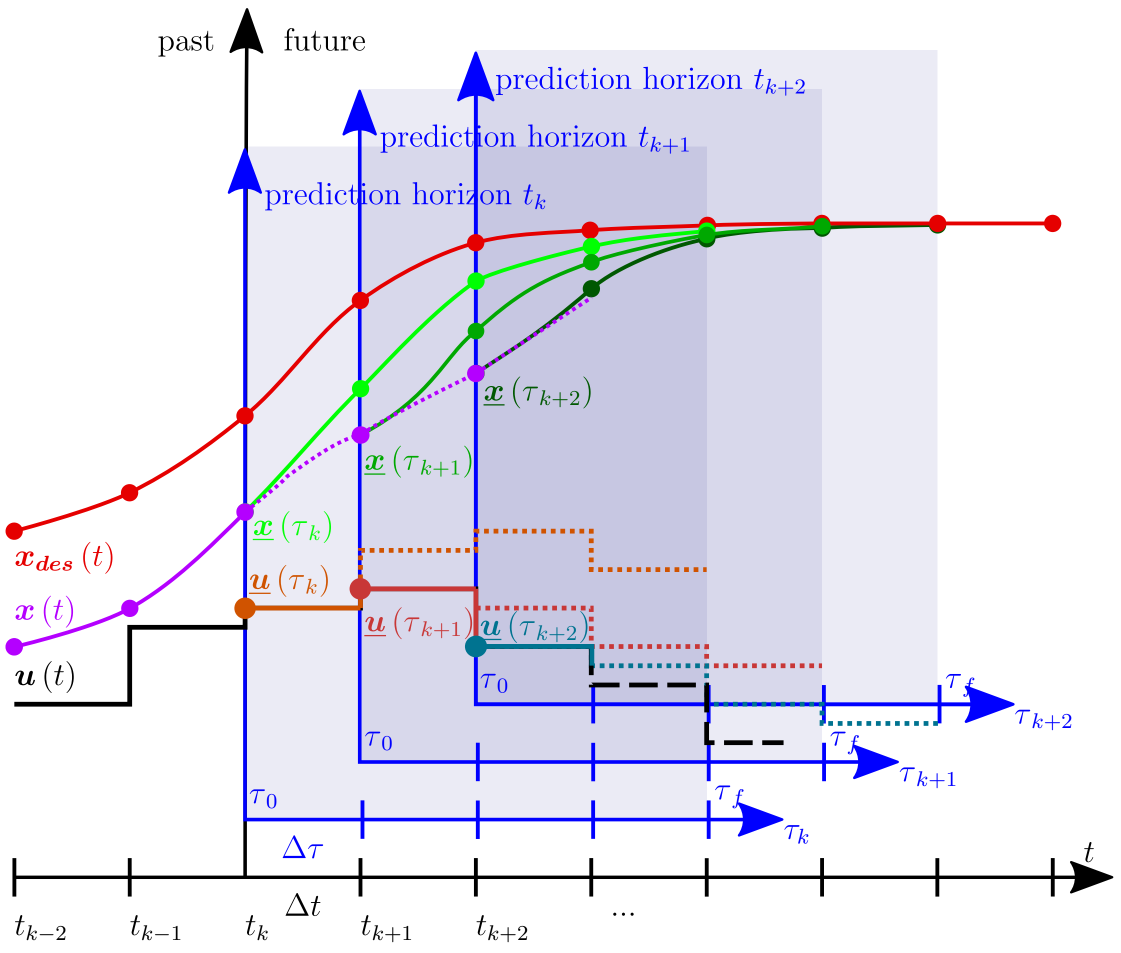

past
future
prediction horizon t_{k+2}
prediction horizon t_{k+1}
prediction horizon t_k
$\underline{\boldsymbol{x}}(\tau_{k+2})$
$\underline{\boldsymbol{x}}(\tau_{k+1})$
$\underline{\boldsymbol{x}}(\tau_k)$
$\boldsymbol{x_{des}}(t)$
$\underline{\boldsymbol{u}}(\tau_k)$
$\boldsymbol{x}(t)$
$\underline{\boldsymbol{u}}(\tau_{k+1})$
$\underline{\boldsymbol{u}}(\tau_{k+2})$
$\boldsymbol{u}(t)$
τ_0
τ_f
τ_{k+2}
τ_{k+1}
τ_k
$\Delta\tau$
Δt
t
t_{k-2}
t_{k-1}
t_k
t_{k+1}
t_{k+2}
...

Figure 4.1: *MPC* receding horizon scheme

The *MPC* algorithms considered within this work are following the scheme of Figure 4.1 which can be summarized in Table 1.

Algorithm 1: Generic *MPC* Algorithm

```
while Time loop $t_k = t_{k-1} + \Delta t$ do
    Initiate prediction horizon with measured state $\underline{\boldsymbol{x}}[t_k] = \boldsymbol{x}[t_k]$;
    while Iterative solution of OCP in horizon $\tau = [t_k, t_k + T]$ do
        Predict future plant behavior $\underline{\boldsymbol{x}}(\tau)$ with $\boldsymbol{f}(\underline{\boldsymbol{x}}(\tau), \underline{\boldsymbol{u}}(\tau), \tau)$;
        Compute/Optimize plant controls $\underline{\boldsymbol{u}}(\tau)$ by minimizing $J(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}})$
    end
    Apply control e.g. $\underline{\boldsymbol{u}}[t_{k+1}] \rightarrow \boldsymbol{u}(t_{k+1})$
end
```

The algorithm in Table 1 clarifies that the actual challenge of *MPC* is to solve its inherent *OCP*. This problem can be arbitrarily complex according to the systems dynamics, control objectives and constraints. The resulting challenges are the complexity of solver algorithms, high computational burden, variety of solver parameters and the necessity of a plant model. Consequently, the use of *MPC* on embedded systems and for fast real-time applications is limited. However, the possibility of fully exploiting the plant's dynamics, the simplicity in defining complex control objectives while maintaining a simple control policy and the consideration of constraints makes *MPC* the choice of controller for this work. A summary of advantages and disadvantages of *MPC* is given Table 4.1.

MPC advantages and disadvantages

Advantages:

+ 1. plant dynamics can be fully exploited
+ 2. generic consideration of complex control goals
+ 3. simple control policy for complex systems
+ 4. generic consideration of constraints

Disadvantages:

- 1. plant model is required
- 2. high computational load
- 3. high algorithmic complexity
- 4. high number of control parameters

Table 4.1: MPC: advantages and disadvantages

The critical point for the *MPC* of *UAV*s is the real-time capability. In this context, §4 is discussing fast *OCP*-solvers and constraint handling techniques which are efficient in terms of memory usage and computation time.

4.1 State of the art in model predictive control

The real-time capability is a crucial aspect of *MPC* which has led to a variety of fast optimization algorithms. A theoretically well-established and widely used fast *MPC* algorithm is sequential quadratic programming (*SQP*) in combination with Newton-type solvers with e.g. Gauß-Newton or Broyden-Fletcher-Goldfarb-Shannon (*BFGS*) Hessian approximation. A compact overview on the convergence of Newton's method with different Hessian approximations is provided in [Die14]. A variety of related algorithms are integrated in the comprehensive *NMPC* framework *ACADO* [DFH09]. The computational efficiency and real-time feasibility for fast mobile robot systems have been validated experimentally. One example is given in [GM16] where collision avoidance scenarios with an aerial manipulator are presented using *BFGS*. Another comprehensive nonlinear optimization framework which contains an *SQP* implementation is *NLopt* [Kra88].

An example of a gradient-descent-based fast *NMPC* framework is *GRAMPC* (accessible via [GU14]). The package offers transparent fast code in *C*. The advantage of gradient-descent-based approaches is the intuitive parametrization, and that only first-order derivatives of system/cost functions are required. The major drawback is the lower convergence in comparison to Newton-type methods.

A computationally efficient *NMPC* alternative is the Continuation Generalized Minimal RESidual (*C/GMRES*) method as presented in [Oht04]. *C/GMRES* is using a continuation method [RD83] to trace the optimal solution trajectory. Its underlying concept is introduced in §4.3.3. A compact version in *C++* code is freely available under [Oht]. The low computational burden of *C/GMRES* makes it particularly suitable for the control of fast systems, such as e.g gasoline engines [KSJ14], hover crafts [SO02, SO03] and eco-cruise control scenarios [SAVD16]. To increase the numerical stability, the multiple shooting derivative *MSC/GMRES* has been developed in [SOD06]. The related computational effort is minimized using condensing as shown in [SOD09] and [SO10]. The result is a computationally efficient and robust condensed multiple shooting derivative *CMSC/GMRES*. In the previous publications [DKB^{+}18, DKMV17, DKMV16b, Den16], *CMSC/GMRES* has been successfully implemented to control a commercial quadrotor. The low computation time and real-time capability of *CMSC/GMRES* has been confirmed experimentally for the given scenarios. For this reason, this work is also based on *CMSC/GMRES*.

Another *MPC* approach is to compute the *OCP* costs for a predefined set of feasible control sequences [BMPL^{+}14]. The control sequence with the lowest cost function value is then applied to the system. The underlying idea is, that a limited set of control sequences also limits the computational burden. The disadvantage of this method is, that the limited set can decrease the optimality of the solution and the set size has to be increased with the nonlinearity of the system.

Most solvers are solving *OCP*s by using the corresponding first-order optimality conditions. In this context, a generic approach to describe the optimality of *OCP*s is Pontryagin's Maximum Principle (*PMP*). A comprehensive summary of *PMP* for different *OCP*s and applied constraints is given in [HSV95, Cha07]. The necessary first-order optimality conditions are thereby derived using the calculus of variations.

The handling of inequality constraints in *MPC* itself is a wide field of research. An overview and benchmark of computationally efficient inequality constraint handling techniques with *C/GMRES* is given in [HNB$^+$15]. This includes the exterior penalty, auxiliary variable and the Fischer-Burmeister semi-smooth transformation. A very popular family of constraint handling methods are primal barrier methods. In the context of economic optimization, a variety of barrier functions and their modifications are discussed in [Gas13, Liu03]. This encyclopedia of economic optimization also features active set constraint handling methods for *QP*. The major disadvantage of primal barrier and auxiliary variable methods is that a violation leads to an infeasible *OCP* and accordingly to a crash of the *MPC* solver. This is particularly problematic for light-weight solvers, as they do not prevent from invalid values (*inf*, *nan*) within the prediction horizon. Consequently, they do not automatically recover from an infeasible state. If a small constraint violation can be accepted, one way to avoid this problem is the use of soft constraints. An example for such a transformation into a soft constraint is the saturation function approach. A saturation function approach is approximating the inequality constraint switching behavior by an analytical function (e.g. *sigmoid* [LL17], *tanh*). The result is a potential function. A comprehensive study on such saturation function approaches in combination with *MPC* of multi *UAV* scenarios is given in [BMPL$^+$14]. The provided examples include Collision Avoidance (*CA*), area exploration and formation flying.

Due to the complexity of *NMPC* algorithms, a major challenge is to prove closed-loop stability. In order to limit the scope of this work, the theoretical stability analysis has been omitted throughout this thesis. A short stability discussion is given in §4.6. The local stability is stated by experimental results. More detailed information about *NMPC* stability analysis of discrete *NMPC* has been published in [GP17]. It provides stability proofs using stabilizing terminal constraints and relaxation of dynamic programming. Another generic framework for *NMPC* stability proofs is Input-to-State-Stability [DFH09] which exploits the boundness and convergence of the cost function. Less known algorithms like *C/GMRES* are still suffering from limited theoretical analysis. While [Oht04] is stating the boundness of error for the optimality condition under certain conditions, there has not been any publication which is theoretically stating the closed-loop stability of any physical system to the author's knowledge. In his lecture notes [Die17] Prof. Diehl is mentioning, that the stability proof for *C/GMRES* is in principle covered by the stability analysis of a real-time iteration scheme presented in [DFA07]. Nevertheless, this has to be combined with the theoretical analysis of the *C/GMRES* inherent continuation method and Krylov-space solver *FDGMRES* [Kel95]. An introduction of Krylov-space techniques for linear systems is given in [Gut07].

4.2 Optimality

In the context of *MPC*, fast solvers are typically assuming convexity and exploiting the first-order optimality conditions to solve *OCP*s. For static optimization problems these optimality conditions are known as Karush-Kuhn-Tucker (*KKT*) conditions as given in §4.2.1. By using calculus of variations, similar conditions can be derived for dynamic *OCP*s known as Pontryagin's maximum principle (*PMP*), as further discussed in §4.2.2.

4.2.1 Static optimization problems

Considering a constrained static optimization problem of the form

$$\min_{\boldsymbol{u}} \quad J(\boldsymbol{u}) \qquad J : \mathbb{R}^{n_u} \to \mathbb{R}, \qquad J \in \mathcal{C}^2 \tag{4.7}$$

$$s.\ t. \quad \mathbf{0} \geq \boldsymbol{c_{in}}(\boldsymbol{u}) \qquad \boldsymbol{c_{in}} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_{c_{in}}}, \qquad \boldsymbol{c_{in}} \in \mathcal{C}^2 \tag{4.8}$$

$$\mathbf{0} = \boldsymbol{c_{eq}}(\boldsymbol{u}) \qquad \boldsymbol{c_{eq}} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_{c_{eq}}}, \qquad \boldsymbol{c_{eq}} \in \mathcal{C}^2, \tag{4.9}$$

the optimal point $\boldsymbol{u}^*$ is minimizing the cost functional $J(\boldsymbol{u})$ while satisfying equality $\boldsymbol{c_{eq}}$ and inequality constraints $\boldsymbol{c_{in}}$. In this context, $\mathcal{C}^2$ is the class of continuous functions with continuous first and second derivative. We assume the existence of a unique solution of (4.7)-(4.9) based on the convexity of $J(\boldsymbol{u})$. For equality constraints, the optimal point is given if the negative cost gradient $-\nabla_{\boldsymbol{u}} J(\boldsymbol{u})$ and the constraint gradient $\nabla_{\boldsymbol{u}} c_{eq}(\boldsymbol{u})$ are parallel:

$$-\nabla_{\boldsymbol{u}} J(\boldsymbol{u}) = \lambda_{eq} \nabla_{\boldsymbol{u}} c_{eq}(\boldsymbol{u}). \tag{4.10}$$

For an inequality constraint there are two cases to distinguish. In the first case, the minimum of $J(\boldsymbol{u})$ lays within the feasible region. As a result, the problem solution is independent of the inequality constraint. The second case considers the minimum of $J(\boldsymbol{u})$ to lay outside the region compliant with $\boldsymbol{c_{in}}$. To evaluate this case, (4.10) for equality constraints has to be extended. For inequality constraints the constraint gradient $\nabla_{\boldsymbol{u}} c_{in}(\boldsymbol{u})$ is pointing in the direction of the incompliant region. For minimization problems the negative gradient $-\nabla_{\boldsymbol{u}} J(\boldsymbol{u})$ of the cost constraint is pointing towards the minimum of $J(\boldsymbol{u})$. This requires $-\nabla_{\boldsymbol{u}} J(\boldsymbol{u})$ and $\nabla_{\boldsymbol{u}} c_{in}(\boldsymbol{u})$ to point in opposite directions at the optimal point $\boldsymbol{u}^*$

$$-\nabla_{\boldsymbol{u}} J(\boldsymbol{u}) = \lambda_{in} \nabla_{\boldsymbol{u}} c_{in}(\boldsymbol{u}) \quad \wedge \quad \lambda_{in} \geq 0. \tag{4.11}$$

The optimality conditions (4.10)-(4.11) for the constraint problem (4.7)-(4.9) can be generalized with the Lagrangian

$$\mathcal{L}(\boldsymbol{u}) = \nabla_{\boldsymbol{u}} J(\boldsymbol{u}) + \boldsymbol{\lambda_{in}}^{\top} \nabla_{\boldsymbol{u}} \boldsymbol{c_{in}}(\boldsymbol{u}) + \boldsymbol{\lambda_{eq}}^{\top} \nabla_{\boldsymbol{u}} \boldsymbol{c_{eq}}(\boldsymbol{u}). \tag{4.12}$$

This results in the Karush-Kuhn-Tucker (*KKT*) optimality conditions [Noc06]:

Karush Kuhn Tucker (*KKT*): first order optimality conditions

$$\text{Stationary condition: } \mathbf{0} = \nabla_{\boldsymbol{u}} \mathcal{L}(\boldsymbol{u}) \tag{4.13}$$

$$\text{Complementary slackness condition: } \mathbf{0} = \boldsymbol{\lambda_{in}}^{\top} \boldsymbol{c_{in}}(\boldsymbol{u}) \tag{4.14}$$

$$\begin{aligned} \text{Primal feasibility condition: } \mathbf{0} &\geq \boldsymbol{c_{in}}(\boldsymbol{u}) \\ \mathbf{0} &= \boldsymbol{c_{eq}}(\boldsymbol{u}) \end{aligned} \tag{4.15}$$

$$\text{Dual feasibility condition: } \mathbf{0} \geq \boldsymbol{\lambda_{in}} \tag{4.16}$$

The first-order optimality conditions (4.13)-(4.16) are necessary conditions. To reach to sufficient optimality conditions, it has to be ensured that the optimal point $\boldsymbol{u}^*$ is a local minimum. For an unconstrained problem, this is guaranteed if the Hessian is positive semi definite

$$\nabla^2_{\boldsymbol{uu}} \mathcal{L}(\boldsymbol{u}) > \mathbf{0}. \tag{4.17}$$

For the constraint case, the minimum of $J(\boldsymbol{u})$ can also lay on an active constraint. Accordingly, the Hessian has to increase in the feasible area around the considered $\boldsymbol{u}$

$$\boldsymbol{s}^{\top} \nabla^2_{\boldsymbol{uu}} \mathcal{L}(\boldsymbol{u})\, \boldsymbol{s} > 0 \ \ \forall \boldsymbol{s} \tag{4.18}$$

$$\boldsymbol{s} \in \mathbb{R}^{n_{\boldsymbol{u}}} \text{ s.t. } \mathbf{0} = \boldsymbol{c_{eq}}(\boldsymbol{u}) \wedge \mathbf{0} \geq \boldsymbol{c_{in}}(\boldsymbol{u}) \tag{4.19}$$

4.2.2 Optimality of dynamic optimal control problems

Control is typically applied on dynamically changing systems, wherefore the static optimization approach (4.7)-(4.9) has to be extended. First, the unconstrained case of a dynamic *OCP*s is considered, as following

$$\min_{\underline{\boldsymbol{u}}(\cdot)} \quad J(\underline{\boldsymbol{u}}) = V(\underline{\boldsymbol{x}}(t+T), t+T) + \int_t^{t+T} L(\underline{\boldsymbol{x}}(\tau), \underline{\boldsymbol{u}}(\tau), \tau)\, \mathrm{d}\tau \tag{4.20}$$

$$s.\ t. \quad \dot{\underline{\boldsymbol{x}}}(\tau) = \boldsymbol{f}(\underline{\boldsymbol{x}}(\tau), \underline{\boldsymbol{u}}(\tau), \tau) \tag{4.21}$$

$$\underline{\boldsymbol{x}}(t) = \boldsymbol{x}(t). \tag{4.22}$$

The particular condition here is the explicit appearance of the system dynamics (4.21). The optimality conditions of dynamic *OCP*s can be derived similarly to the *KKT* optimality conditions for static *OCP*s. For means of visibility, the time dependency within the horizon is not explicitly shown in the following, e.g. $L\left(\underline{\boldsymbol{x}}\left(\tau\right), \underline{\boldsymbol{u}}\left(\tau\right), \tau\right) \Rightarrow L\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right)$ Analog to (4.12) for static optimization problems, the optimality conditions for the dynamic optimization problem (4.20)-(4.22) can be generalized using the Hamiltonian

$$\mathcal{H}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right) = L\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) + \underline{\boldsymbol{\lambda}}^\top \boldsymbol{f}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right). \tag{4.23}$$

Under the assumption of a given horizon length T, the necessary optimality conditions for *OCP* (4.20)-(4.22) can be derived using calculus of variations [Cha07, p.114,129]

Optimality conditions for unconstrained *OCP*

First order optimality conditions:

$$\dot{\underline{\boldsymbol{x}}}\left(\tau\right) = +\nabla_{\underline{\boldsymbol{\lambda}}}\mathcal{H}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right) = \boldsymbol{f}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) \tag{4.24}$$

$$\dot{\underline{\boldsymbol{\lambda}}}\left(\tau\right) = -\nabla_{\underline{\boldsymbol{x}}}\mathcal{H}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right) = -\nabla_{\underline{\boldsymbol{x}}}L\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right) - \left(\nabla_{\underline{\boldsymbol{x}}}\boldsymbol{f}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right)\right)^\top \underline{\boldsymbol{\lambda}} \tag{4.25}$$

$$\boldsymbol{0} = +\nabla_{\underline{\boldsymbol{u}}}\mathcal{H}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right) = +\nabla_{\underline{\boldsymbol{u}}}L\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right) + \left(\nabla_{\underline{\boldsymbol{u}}}\boldsymbol{f}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right)\right)^\top \underline{\boldsymbol{\lambda}} \tag{4.26}$$

Transversality condition: $$\underline{\boldsymbol{\lambda}}\left(t+T\right) = \nabla_{\underline{\boldsymbol{x}}}V\left(t+T\right)\Big|_{t+T} \tag{4.27}$$

Boundary condition: $$\underline{\boldsymbol{x}}\left(t\right) = \boldsymbol{x}\left(t\right) \tag{4.28}$$

Legendre-Clebsch condition: $$\nabla^2_{\underline{\boldsymbol{uu}}}\mathcal{H}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right)\ \ p.s.d. \tag{4.29}$$

If equality constraints

$$\boldsymbol{c_{eq}} : \mathbb{R}^{n_{\boldsymbol{u}}} \times \mathbb{R}^{n_{\boldsymbol{u}}} \times \mathbb{R} \rightarrow \mathbb{R}^{n_{\boldsymbol{ceq}}}, \boldsymbol{c_{eq}} \in \mathcal{C}^2, \tag{4.30}$$

have to be considered, the *OCP* results to

$$\min_{\underline{\boldsymbol{u}}(\cdot)} \quad J\left(\underline{\boldsymbol{u}}\right) = V\left(\underline{\boldsymbol{x}}\left(t+T\right), t+T\right) + \int_{t}^{t+T} L\left(\underline{\boldsymbol{x}}\left(\tau\right), \underline{\boldsymbol{u}}\left(\tau\right), \tau\right) \mathrm{d}\tau \tag{4.31}$$

$$s.\ t. \quad \dot{\underline{\boldsymbol{x}}}\left(\tau\right) = \boldsymbol{f}\left(\underline{\boldsymbol{x}}\left(\tau\right), \underline{\boldsymbol{u}}\left(\tau\right), \tau\right) \tag{4.32}$$

$$\boldsymbol{0} = \boldsymbol{c_{eq}}\left(\underline{\boldsymbol{x}}\left(\tau\right), \underline{\boldsymbol{u}}\left(\tau\right), \tau\right), \tag{4.33}$$

$$\underline{\boldsymbol{x}}\left(t\right) = \boldsymbol{x}\left(t\right), \tag{4.34}$$

Accordingly the Hamiltonian is extended to the Lagrangian by the constraint terms

$$\mathcal{L}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right) = \mathcal{H}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right) + \underline{\boldsymbol{\lambda_{eq}}}^\top \boldsymbol{c_{eq}}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) \tag{4.35}$$

$$\mathcal{L}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}, \tau\right) = L\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) + \underline{\boldsymbol{\lambda}}^\top \boldsymbol{f}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) + \underline{\boldsymbol{\lambda_{eq}}}^\top \boldsymbol{c_{eq}}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right). \tag{4.36}$$

Calculus of variations confirms the direct extension of the Euler-Lagrange equations (4.24)-(4.26) to

$$\begin{aligned}
\dot{\underline{\boldsymbol{x}}}(\tau) &= \nabla_{\underline{\boldsymbol{\lambda}}}\mathcal{L}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau) && (4.37)\\
\dot{\underline{\boldsymbol{\lambda}}}(\tau) &= -\nabla_{\underline{\boldsymbol{x}}}\mathcal{L}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau) && (4.38)\\
&= -\nabla_{\underline{\boldsymbol{x}}}L(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau) - \left(\nabla_{\underline{\boldsymbol{x}}}\boldsymbol{f}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau)\right)^{\top}\underline{\boldsymbol{\lambda}} - \left(\nabla_{\underline{\boldsymbol{x}}}\boldsymbol{c_{eq}}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau)\right)^{\top}\underline{\boldsymbol{\lambda_{eq}}}\\
\boldsymbol{0} &= \nabla_{\underline{\boldsymbol{u}}}\mathcal{L}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau)\\
&= \nabla_{\underline{\boldsymbol{u}}}L(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}}) + \left(\nabla_{\underline{\boldsymbol{u}}}\boldsymbol{f}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau)\right)^{\top}\underline{\boldsymbol{\lambda}} + \left(\nabla_{\underline{\boldsymbol{u}}}\boldsymbol{c_{eq}}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau)\right)^{\top}\underline{\boldsymbol{\lambda_{eq}}} && (4.39)\\
\boldsymbol{0} &= \boldsymbol{c_{eq}}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau) && (4.40)
\end{aligned}$$

For inequality constraints

$$\boldsymbol{c_{in}}:\mathbb{R}^{n_u}\times\mathbb{R}^{n_u}\times\mathbb{R}\rightarrow\mathbb{R}^{n_{c_{in}}},\boldsymbol{c_{in}}\in\mathcal{C}^2, \qquad (4.41)$$

the resulting optimality conditions are differing according to the nature of each constraint. A comprehensive collection on these are given by [Cha07, p.133-161] which includes *OCP*s with mixed sets of pure and mixed state constraints. To limit the scope of this work, here only the *PMP* for *OCP*s with mixed inequality constraints from [Cha07, p.149] shall be given:

$$\begin{aligned}
\min_{\underline{\boldsymbol{u}}(\cdot)} \quad & J(\underline{\boldsymbol{u}}) = V(\underline{\boldsymbol{x}}(t+T),t+T) + \int_t^{t+T} L(\underline{\boldsymbol{x}}(\tau),\underline{\boldsymbol{u}}(\tau),\tau)\,\mathrm{d}\tau && (4.42)\\
s.\ t. \quad & \dot{\underline{\boldsymbol{x}}}(\tau) = \boldsymbol{f}(\underline{\boldsymbol{x}}(\tau),\underline{\boldsymbol{u}}(\tau),\tau) && (4.43)\\
& \boldsymbol{0} \geq \boldsymbol{c_{in}}(\underline{\boldsymbol{x}}(\tau),\underline{\boldsymbol{u}}(\tau),\tau), && (4.44)\\
& \boldsymbol{0} = \boldsymbol{c_{eq}}(\underline{\boldsymbol{x}}(\tau),\underline{\boldsymbol{u}}(\tau),\tau), && (4.45)\\
& \underline{\boldsymbol{x}}(t) = \boldsymbol{x}(t)\,. && (4.46)
\end{aligned}$$

It is assumed, that each inequality constraint concatenated in $\boldsymbol{c_{in}}(\underline{\boldsymbol{x}}(\tau),\underline{\boldsymbol{u}}(\tau),\tau)$ depends explicitely on $\underline{\boldsymbol{u}}$. This can be checked with the rank condition:

$$\operatorname{rank}\left(\nabla_{\underline{\boldsymbol{u}}}\boldsymbol{c_{in}}(\underline{\boldsymbol{x}}(\tau),\underline{\boldsymbol{u}}(\tau),\tau),\operatorname{diag}\left(\boldsymbol{c_{in}}(\underline{\boldsymbol{x}}(\tau),\underline{\boldsymbol{u}}(\tau),\tau)\right)\right) = n_{\boldsymbol{c_{in}}}. \qquad (4.47)$$

Under use of the Lagrangian

$$\mathcal{L}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau) = L(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\tau) + \underline{\boldsymbol{\lambda}}^{\mathrm{T}}\boldsymbol{f}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\tau) + \underline{\boldsymbol{\lambda_{in}}}^{\mathrm{T}}\boldsymbol{c_{in}}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\tau) + \underline{\boldsymbol{\lambda_{eq}}}^{\mathrm{T}}\boldsymbol{c_{eq}}(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\tau)\,. \qquad (4.48)$$

the necessary first-order optimality conditions can be written as [Cha07, p.150]:

Necessary first order optimality conditions for *OCP* using *PMP*

Maximum principle:

$$\mathcal{H}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{\varphi}},\underline{\boldsymbol{\lambda}},\tau\right) \geq \mathcal{H}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right) \qquad \forall \underline{\boldsymbol{\varphi}} \in \left\{\underline{\boldsymbol{u}} \in \mathbb{R}^{n_{\boldsymbol{u}}} : \mathbf{0} \geq \boldsymbol{c_{in}}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\tau\right)\right\} \tag{4.49}$$

Euler-Lagrange equation:

$$\dot{\underline{\boldsymbol{x}}} = \nabla_{\underline{\boldsymbol{\lambda}}}\mathcal{L}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right) = \boldsymbol{f}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right) \tag{4.50}$$

$$\begin{aligned}\dot{\underline{\boldsymbol{\lambda}}} = -\nabla_{\underline{\boldsymbol{x}}}\mathcal{L}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right) &= -\nabla_{\underline{\boldsymbol{x}}}L\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right) - \left(\nabla_{\underline{\boldsymbol{x}}}\boldsymbol{f}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right)\right)^{\top}\underline{\boldsymbol{\lambda}} \\ &- \left(\nabla_{\underline{\boldsymbol{x}}}\boldsymbol{c_{eq}}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right)\right)^{\top}\underline{\boldsymbol{\lambda_{eq}}} - \left(\nabla_{\underline{\boldsymbol{x}}}\boldsymbol{c_{in}}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right)\right)^{\top}\underline{\boldsymbol{\lambda_{in}}}\end{aligned} \tag{4.51}$$

$$\begin{aligned}\mathbf{0} = \nabla_{\underline{\boldsymbol{u}}}\mathcal{L}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right) &= \nabla_{\underline{\boldsymbol{u}}}L\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}}\right) + \left(\nabla_{\underline{\boldsymbol{u}}}\boldsymbol{f}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right)\right)^{\top}\underline{\boldsymbol{\lambda}} \\ &+ \left(\nabla_{\underline{\boldsymbol{u}}}\boldsymbol{c_{eq}}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right)\right)^{\top}\underline{\boldsymbol{\lambda_{eq}}} + \left(\nabla_{\underline{\boldsymbol{u}}}\boldsymbol{c_{in}}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right)\right)^{\top}\underline{\boldsymbol{\lambda_{in}}}\end{aligned} \tag{4.52}$$

Complementary Condition:

$$0 = \nabla_{\underline{\boldsymbol{u}}}c_{in,k}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right)\lambda_{ink}, \qquad 0 \leq \lambda_{in,k}\left(\tau\right) \qquad k \in \left[1,\ldots,n_{\boldsymbol{c_{in}}}\right] \tag{4.53}$$

Constant Hamiltonian Condition:

$$\mathcal{H}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right) = -\int\limits_{t}^{t+T}\nabla_{\tau}\mathcal{L}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda}},\tau\right)d\tau \tag{4.54}$$

$$\text{with } \mathcal{H}\left(\underline{\boldsymbol{x}}\left(t+T\right),\underline{\boldsymbol{u}}\left(t+T\right),\underline{\boldsymbol{\lambda}}\left(t+T\right),t+T\right) = 0$$

The maximum principle (4.49) for the problem (4.42)-(4.46) is mathematically stating the following. The control space is constrained by the given inequality constraints. Every Hamiltonian of this constrained control space is higher than the Hamiltonian of the optimal control trajectory. If the unconstrained optimal solution lays outside of this constrained control space, the constrained optimal solution is accordingly found on the boundary of the constrained control space. Otherwise, the unconstrained and constrained optimal solution coincide. More detailed information regarding *PMP* is given in [Cha07, p.150]. Particularly the canonical equations (4.50)-(4.52) are important as they are used to solve *OCP*s in fast *NMPC* solvers.

4.3 Solvers

The major effort in *MPC* is to solve the underlying *OCP*. As every technical problem can be reformulated as optimization problem, there is vast literature on optimization techniques and solvers. Of particular interest for this thesis are fast solvers that are able to solve nonlinear problems in the form of *OCP* (4.1)-(4.5). In the following, §4.3 assesses a selection of promising algorithms for *UAV* control. This includes *ACADO*'s Gauß-Newton *SQP*, *GRAMPC*'s adaptive gradient-descent approach and *C/GMRES*, respectively *CMSC/GMRES*. As related work for *C/GMRES* (e.g. [Oht04]) and *CMSC/GMRES* (e.g. [SOD09]) are written very compact, the related sections §4.3.3, respectively §4.3.4, give a detailed step by step derivation of the algorithms in order to facilitate future work in the direction of stability proofs.

4.3.1 Gauß-Newton sequential quadratic programming

Omitting inequality constraints for means of visibility, the optimality conditions for the static optimization problem (4.7)-(4.9) can be concatenated to [Die14]

$$\underline{\boldsymbol{w}} := \begin{bmatrix} \underline{\boldsymbol{u}} \\ \underline{\boldsymbol{\lambda}} \end{bmatrix}, \qquad \boldsymbol{0} = \mathfrak{f}\left(\underline{\boldsymbol{w}}\right) = \begin{bmatrix} \nabla_{\underline{\boldsymbol{u}}} \mathcal{L}\left(\underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}\right) \\ \boldsymbol{c}\left(\underline{\boldsymbol{u}}\right) \end{bmatrix}. \tag{4.55}$$

As a remark, the concatenated optimality conditions $\mathfrak{f}$ are differing from the system dynamics $\boldsymbol{f}$. Assuming convexity, the optimal $\underline{\boldsymbol{u}}$ can be computed iteratively with Newton's method [Die14]

$$\mathfrak{f}\left(\underline{\boldsymbol{w}}^j\right) + \frac{\partial \mathfrak{f}\left(\underline{\boldsymbol{w}}^j\right)}{\partial \underline{\boldsymbol{w}}^j}\left(\underline{\boldsymbol{w}}^{j+1} - \underline{\boldsymbol{w}}^j\right) = \boldsymbol{0}. \tag{4.56}$$

which uses a linearization of $\mathfrak{f}$ at current iterate $\underline{\boldsymbol{w}}^j$. With the concatenated optimality conditions (4.55), the Newton iteration (4.56) yields [Die14]

$$\begin{bmatrix} \nabla_{\underline{\boldsymbol{u}}} \mathcal{L}\left(\underline{\boldsymbol{u}}^j, \underline{\boldsymbol{\lambda}}^j\right) \\ \boldsymbol{c}\left(\underline{\boldsymbol{u}}^j\right) \end{bmatrix} + \begin{bmatrix} \nabla^2_{\underline{\boldsymbol{u}}\underline{\boldsymbol{u}}} \mathcal{L}\left(\underline{\boldsymbol{u}}^j, \underline{\boldsymbol{\lambda}}^j\right) & \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c}\left(\underline{\boldsymbol{u}}^j\right) \\ \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c}\left(\underline{\boldsymbol{u}}^j\right)^\top & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{u}}^{j+1} - \underline{\boldsymbol{u}}^j \\ \underline{\boldsymbol{\lambda}}^{j+1} - \underline{\boldsymbol{\lambda}}^j \end{bmatrix} = \boldsymbol{0}. \tag{4.57}$$

Using the Lagrangian derivative

$$\nabla_{\underline{\boldsymbol{u}}} \mathcal{L}\left(\underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}\right) = \nabla_{\underline{\boldsymbol{u}}} J\left(\underline{\boldsymbol{u}}\right) + \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c}\left(\underline{\boldsymbol{u}}\right) \underline{\boldsymbol{\lambda}} \tag{4.58}$$

the Newton iteration (4.57) can be further simplified to [Die14]

$$\begin{bmatrix} \nabla_{\underline{\boldsymbol{u}}} J\left(\underline{\boldsymbol{u}}^j\right) \\ \boldsymbol{c}\left(\underline{\boldsymbol{u}}^j\right) \end{bmatrix} + \begin{bmatrix} \nabla^2_{\underline{\boldsymbol{u}}\underline{\boldsymbol{u}}} \mathcal{L}\left(\underline{\boldsymbol{u}}^j, \underline{\boldsymbol{\lambda}}^j\right) & \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c}\left(\underline{\boldsymbol{u}}^j\right) \\ \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c}\left(\underline{\boldsymbol{u}}^j\right)^\top & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{u}}^{j+1} - \underline{\boldsymbol{u}}^j \\ \underline{\boldsymbol{\lambda}}^{j+1} \end{bmatrix} = \boldsymbol{0} \tag{4.59}$$

For the system 4.59, the Newton iteration results to [Die14]

$$\begin{bmatrix} \underline{\boldsymbol{u}}_{k+1} \\ \underline{\boldsymbol{\lambda}}_{k+1} \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{u}}_{k+1} \\ 0 \end{bmatrix} + \begin{bmatrix} \nabla^2_{\underline{\boldsymbol{u}}\underline{\boldsymbol{u}}} \mathcal{L}\left(\underline{\boldsymbol{u}}^j, \underline{\boldsymbol{\lambda}}^j\right) & \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c}\left(\underline{\boldsymbol{u}}^j\right) \\ \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c}\left(\underline{\boldsymbol{u}}^j\right)^\top & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \nabla_{\underline{\boldsymbol{u}}} J\left(\underline{\boldsymbol{u}}^j\right) \\ \boldsymbol{c}\left(\underline{\boldsymbol{u}}^j\right) \end{bmatrix} \tag{4.60}$$

As shown in (4.60), the exact Newton method requires the inverse of the Hessian which is computationally expensive. Newton-type methods try to overcome this by approximating the Hessian $\nabla^2_{\underline{\boldsymbol{u}}} \mathcal{L}\left(\underline{\boldsymbol{u}}^j, \underline{\boldsymbol{\lambda}}^j\right) \approx \mathbf{A}^j$. A common approximation for a least square cost-function is the Gauß-Newton approximation $\nabla^2_{\underline{\boldsymbol{u}}} \mathcal{L}\left(\underline{\boldsymbol{u}}^j, \underline{\boldsymbol{\lambda}}^j\right) \approx \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{R} \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{R}^\top$ The most common types of approximations are given in Table 4.2 [Die14].

Method	Exact Hessian	Gauss-Newton	BFGS
$\boldsymbol{A} \approx$ Hessian	$\nabla^2_{\underline{\boldsymbol{u}}} \mathcal{L}\left(\underline{\boldsymbol{u}}, \underline{\boldsymbol{\lambda}}\right)$	$\nabla_{\underline{\boldsymbol{u}}} \boldsymbol{R} \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{R}^\top$	$\boldsymbol{A} - \frac{\boldsymbol{A}\boldsymbol{s}(\boldsymbol{s})^\top \boldsymbol{A}}{(\boldsymbol{s})^\top \boldsymbol{A}\boldsymbol{s}} + \frac{\boldsymbol{y}\boldsymbol{y}^\top}{(\boldsymbol{s})^\top \boldsymbol{y}}$
Convergence $\Vert\underline{\boldsymbol{w}}_{k+1} - \underline{\boldsymbol{w}}^*\Vert \leq$	Quadratic: $\frac{\boldsymbol{w}}{2}\Vert\underline{\boldsymbol{w}}_{k+1} - \underline{\boldsymbol{w}}^*\Vert^2$	Linear: $\kappa\Vert\underline{\boldsymbol{w}}_{k+1} - \underline{\boldsymbol{w}}^*\Vert$ $\kappa = \mathcal{O}\left(\Vert\boldsymbol{R}\left(x^*\right)\Vert\right)$	Super-linear: $\kappa\Vert\underline{\boldsymbol{w}}_{k+1} - \underline{\boldsymbol{w}}^*\Vert$ $\kappa \rightarrow 0$

Table 4.2: Newton-type algorithms with different approximations of the Hessian

The Newton iteration (4.59) is equally representing the optimality conditions for the quadratic program [Die14]

$$\min_{\underline{\boldsymbol{u}}} \quad \nabla J\left(\underline{\boldsymbol{u}}^j\right)^\top \left(\underline{\boldsymbol{u}} - \underline{\boldsymbol{u}}^j\right) + \frac{1}{2}\left(\underline{\boldsymbol{u}} - \underline{\boldsymbol{u}}^j\right)^\top \boldsymbol{A}^j \left(\underline{\boldsymbol{u}} - \underline{\boldsymbol{u}}^j\right) \tag{4.61}$$

$$s.\ t. \quad \boldsymbol{c_{eq}}\left(\underline{\boldsymbol{u}}^j\right) + \nabla \boldsymbol{c_{eq}}\left(\underline{\boldsymbol{u}}^j\right)^\top \left(\underline{\boldsymbol{u}} - \underline{\boldsymbol{u}}^j\right). \tag{4.62}$$

As a consequence, the sequential iteration of Newton's method can be interpreted as Sequential Quadratic Programming *SQP* [Die14]. In order to solve a dynamic *OCP*s with *SQP*, the horizon is discretized and the controls are considered as independent optimization variables. This transforms the dynamic *OCP* into a static optimization program, which can accordingly be solved using *SQP*. In the context of this work, the *ACADO* inherent *SQP* implementation is used as a representative *SQP* implementation.

4.3.2 GRAMPC

A well-documented platform for fast *NMPC* which offers a compact efficient code structure is *GRAMPC*. It features *NMPC* for *OCP*s with control input limitations. The inherent *OCP* is solved by finding the root of the first-order optimality condition with a gradient-based line-search. The *GRAMPC* package offers a variety of different step width calculation approaches and integration methods. Besides the computationally efficient algorithms, one advantage of this library is its transparency and lean design in *C* which facilitates modifications. The package also contains examples for *NMPC* problems, specifically also for a quadrotor. Further details are given in the related publications [GU14, Gra13]. In the following section *GRAMPC* is applied to the nonlinear physical *UAV* model introduced in §3.2.

Table 2 shows the control algorithm. The presented *GRAMPC* configuration uses an adaptive step size calculation that is based on a parabolic interpolation of $\mathcal{H}(\alpha)$. The forward and backward integration is executed with the *HEUN* scheme.

Algorithm 2: Sequence of used *GRAMPC* configuration [GU14]

```
while Time loop $t_k = t_{k-1} + \Delta t \;\; k \in \mathbb{N}^+$ do
    Measure present states $\underline{\boldsymbol{x}}(t_k) = \boldsymbol{x}(t_k)$
    while Gradient Iterations $j \in \mathbb{N}^+$ do
        Adaptation of step width interval $\alpha \in [\alpha_{low}, \alpha_{high}]$
          $[\alpha_{low}, \alpha_{high}] \leftarrow \begin{cases} \frac{2}{3}[\alpha_{low}, \alpha_{high}] & \text{if } \alpha^{j-1} \geq \alpha_{high} + 0.1(\alpha_{high} - \alpha_{low}) \\ \frac{3}{2}[\alpha_{low}, \alpha_{high}] & \text{if } \alpha^{j-1} \geq \alpha_{high} + 0.1(\alpha_{high} - \alpha_{low}) \\ [\alpha_{low}, \alpha_{high}] & \text{otherwise} \end{cases}$
        Calculate control trajectory for the different step widths $\alpha^j$, $\alpha_{low}$, $\alpha_{high}$
        $\underline{\boldsymbol{u}}_k^j(\tau) = \underline{\boldsymbol{u}}_{k-1}(t_k) - \alpha \cdot \mathcal{L}_{\underline{\boldsymbol{u}}}\left(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{u}}_{k-1}, \underline{\boldsymbol{\lambda}}_{k-1}, t_k\right)$
        Forward integration with HEUN-integrator of equation 4.24, 4.50
        $\underline{\boldsymbol{x}}_k^j(\tau) = \int_{t_k}^{t_k+T} f\left(\underline{\boldsymbol{x}}_k^{j-1}, \underline{\boldsymbol{u}}_k^j, \tau\right) \partial\tau$
        Backward integration with HEUN-integrator of equation 4.24, 4.50
        $\underline{\boldsymbol{\lambda}}_k^j(\tau) = \nabla_{\underline{\boldsymbol{x}}} V(t_k + T) - \int_{t_k+T}^{t_k} \nabla_{\underline{\boldsymbol{x}}} \mathcal{L}\left(\underline{\boldsymbol{x}}_k^j, \underline{\boldsymbol{u}}_k^j, \tau\right) d\tau$
        Calculate $\alpha_{opt}^j$ as parabolic minimum of over $\alpha^j$, $\alpha_{low}$, $\alpha_{high}$
        $\min_{\alpha^j} J\left(\underline{\boldsymbol{x}}_k^j(t_k), \underline{\boldsymbol{u}}_k^j(t_k)\right)$
        Calculate and apply control trajectory for $\alpha_{opt}^j$
        $\underline{\boldsymbol{u}}_k(t_k) = \underline{\boldsymbol{u}}_{k-1}(t_k) - \alpha_{opt,k} \cdot \mathcal{H}_{\underline{\boldsymbol{u}}}\left(\underline{\boldsymbol{x}}_{k-1}, \underline{\boldsymbol{u}}_{k-1}, \underline{\boldsymbol{\lambda}}_{k-1}, t_k\right)$
    end
    Apply control $\underline{\boldsymbol{u}}_k(t_{k+1}) \rightarrow \boldsymbol{u}(t_{k+1})$
end
```

4.3.3 C/GMRES

Another fast *NMPC* algorithm is the *C/GMRES*-Method proposed in [Oht04]. In contrast to the line-search method introduced in §4.3.2, *C/GMRES* calculates the control trajectory update explicitly. To explain its principle of operation, we consider following discrete *OCP* [Oht04]

$$\begin{array}{lrcll}
\min\limits_{\underline{\boldsymbol{u}}} & J\left(\underline{\boldsymbol{x}}\left(t\right),\underline{\boldsymbol{u}}\left(t\right),t\right) & = & V\left(\underline{\boldsymbol{x}}_N\left(t\right),t+\Delta\tau\cdot N\right)+\sum\limits_{l=0}^{N-1}L\left(\underline{\boldsymbol{x}}_l\left(t\right),\underline{\boldsymbol{u}}_l\left(t\right),t_l\right)\mathrm{d}t & \\
s.\ t. & \underline{\boldsymbol{x}}_{l+1}\left(t\right) & = & \underline{\boldsymbol{x}}_l\left(t\right)+\Delta\tau\cdot\boldsymbol{f}\left(\underline{\boldsymbol{x}}_l\left(t\right),\underline{\boldsymbol{u}}_l\left(t\right),t_l\right) & \in\mathbb{R}^{n_{\underline{\boldsymbol{x}}}} \\
 & \mathbf{0} & = & \boldsymbol{c_{eq}}\left(\underline{\boldsymbol{x}}_l\left(t\right),\underline{\boldsymbol{u}}_l\left(t\right),t_l\right) & \in\mathbb{R}^{n_{\underline{\boldsymbol{c_{eq}}}}} \\
 & \underline{\boldsymbol{x}}_0\left(t\right) & = & \underline{\boldsymbol{x}}\left(t\right) &
\end{array} \tag{4.63}$$

which leads to the Lagrangian [Oht04]

$$\mathcal{L}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{\lambda}},\underline{\boldsymbol{u}},\underline{\boldsymbol{\lambda_{eq}}}\right):=L\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}}\right)+\underline{\boldsymbol{\lambda}}^{\mathrm{T}}\boldsymbol{f}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}}\right)+\underline{\boldsymbol{\lambda_{eq}}}^{\mathrm{T}}\boldsymbol{c_{eq}}\left(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}}\right). \tag{4.64}$$

Where $\boldsymbol{c_{eq}}$ are equality constraints and $\underline{\boldsymbol{\lambda_{eq}}}$ are the corresponding Lagrange multipliers. The necessary first-order condition according to calculus of variations (see §4.2) results in

$$0=\nabla_{\underline{\boldsymbol{u}}}\mathcal{L}\left(\underline{\boldsymbol{x}}_l\left(t\right),\underline{\boldsymbol{\lambda}}_{l+1}\left(t\right),\underline{\boldsymbol{u}}_l\left(t\right),\underline{\boldsymbol{\lambda_{eq}}}_l\left(t\right)\right)\ \in\mathbb{R}^{n_{\underline{\boldsymbol{u}}}}. \tag{4.65}$$

Ohtsuka in [Oht04] considers the equality constraint Lagrange multipliers $\underline{\boldsymbol{\lambda_{eq}}}$ as additional input, which leads to the combined discretized control vector

$$\underline{\boldsymbol{w}}\left(t\right):=\left[\underline{\boldsymbol{u}}_0^{\mathrm{T}}\left(t\right)\quad\underline{\boldsymbol{\lambda_{eq}}}_0^{\mathrm{T}}\left(t\right)\quad\underline{\boldsymbol{u}}_1^{\mathrm{T}}\left(t\right)\quad\underline{\boldsymbol{\lambda_{eq}}}_1^{\mathrm{T}}\left(t\right)\quad\ldots\quad\underline{\boldsymbol{u}}_{N-1}^{\mathrm{T}}\left(t\right)\quad\underline{\boldsymbol{\lambda_{eq}}}_{N-1}^{\mathrm{T}}\left(t\right)\right]\in\mathbb{R}^{n_{\underline{\boldsymbol{u}}}\times N}. \tag{4.66}$$

Furthermore the optimality conditions and constraints can be combined to the vector condition [Oht04]

$$\mathbf{0}=\mathfrak{f}\left(\underline{\boldsymbol{w}}\left(t\right),\underline{\boldsymbol{x}}\left(t\right),t\right)=\begin{bmatrix}\nabla_{\underline{\boldsymbol{u}}}\mathcal{L}^{\mathrm{T}}\left(\underline{\boldsymbol{x}}_0\left(t\right),\underline{\boldsymbol{\lambda}}_1\left(t\right),\underline{\boldsymbol{u}}_0\left(t\right),\underline{\boldsymbol{\lambda_{eq}}}_0\left(t\right)\right)\\ \boldsymbol{c_{eq}}\left(\underline{\boldsymbol{x}}_0\left(t\right),\underline{\boldsymbol{u}}_0\left(t\right),t\right)\\ \vdots\\ \nabla_{\underline{\boldsymbol{u}}}\mathcal{L}^{\mathrm{T}}\left(\underline{\boldsymbol{x}}_{N-1}\left(t\right),\underline{\boldsymbol{\lambda}}_N\left(t\right),\underline{\boldsymbol{u}}_{N-1}\left(t\right),\underline{\boldsymbol{\lambda_{eq}}}_{N-1}\left(t\right)\right)\\ \boldsymbol{c_{eq}}\left(\underline{\boldsymbol{x}}_{N-1}\left(t\right),\underline{\boldsymbol{u}}_{N-1}\left(t\right),t\right)\end{bmatrix} \tag{4.67}$$

The idea of *C/GMRES* is to trace the time-dependent variation of the optimal solution, instead of directly solving (4.67) with Newton's method. This is computationally efficient. To start the *MPC* within a feasible set, $\underline{\boldsymbol{u}}$ is chosen to satisfy $\underline{\boldsymbol{u}}\left(0\right)=\mathbf{0}$. The basic idea is to stabilize the system asymptotically in the origin with the constraint (4.68)

$$\dot{\mathfrak{f}} = -\xi\mathfrak{f}. \tag{4.68}$$

This allows determining $\dot{\mathfrak{f}}$ explicitly via the derivative of (4.68) which results in

$$\underline{\dot{\boldsymbol{w}}} = \mathfrak{f}_{\underline{\boldsymbol{w}}}^{-1}\left(-\xi\mathfrak{f} - \mathfrak{f}_{\underline{\boldsymbol{x}}}\underline{\dot{\boldsymbol{x}}} - \mathfrak{f}_t\right). \tag{4.69}$$

The stabilization factor ξ is chosen according to the system dynamics. Determining the inverse of the Jacobian $\mathfrak{f}_{\underline{\boldsymbol{u}}}^{-1}$ and the Jacobian themselves with analytical methods is computational expensive. To tackle this issue, equation (4.69) is reformulated with the finite forward difference method. This method is explained in detail in [KK04, AWF11]. An example for $\mathbb{R}^2$ is shown in (4.70) as

$$\frac{\partial\mathfrak{f}(x_1,x_2)}{\partial\left[x_1,x_2\right]^{\mathrm{T}}}\boldsymbol{\sigma} = \begin{bmatrix}\frac{\partial\mathfrak{f}_1}{\partial x_1} & \frac{\partial\mathfrak{f}_1}{\partial x_2}\\ \frac{\partial\mathfrak{f}_2}{\partial x_1} & \frac{\partial\mathfrak{f}_2}{\partial x_2}\end{bmatrix}\boldsymbol{\sigma} \approx \begin{bmatrix}\frac{\mathfrak{f}_1(x_1+h\sigma_1,x_2+h\sigma_2)-\mathfrak{f}_1(x_1,x_2)}{h}\\ \frac{\mathfrak{f}_2(x_1+h\sigma_1,x_2+h\sigma_2)-\mathfrak{f}_1(x_1,x_2)}{h}\end{bmatrix}, \tag{4.70}$$
$$\text{with } \boldsymbol{x} = \begin{bmatrix}x_1 & x_2\end{bmatrix}^{\mathrm{T}},\ \boldsymbol{\sigma} = \begin{bmatrix}\sigma_1 & \sigma_2\end{bmatrix}^{\mathrm{T}}, \mathfrak{f}(x_1,x_2) = \begin{bmatrix}\mathfrak{f}_1(x_1,x_2) & \mathfrak{f}_2(x_1,x_2)\end{bmatrix}^{\mathrm{T}}.$$

The same forward difference approximation can be used to approximate the Jacobian $\dot{\mathfrak{f}}$ in (4.68). This is shown in (4.72) and can be reformulated to (4.73)

$$\dot{\mathfrak{f}}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}},t) = -\xi\mathfrak{f}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}},t) \tag{4.71}$$
$$\frac{\mathfrak{f}(\underline{\boldsymbol{w}}+h\underline{\dot{\boldsymbol{w}}},\underline{\boldsymbol{x}}+h\underline{\dot{\boldsymbol{x}}},t+h)-\mathfrak{f}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}},t)}{h} = -\xi\mathfrak{f}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}},t) \tag{4.72}$$
$$\frac{\mathfrak{f}(\underline{\boldsymbol{w}}+h\underline{\dot{\boldsymbol{w}}},\underline{\boldsymbol{x}}+h\underline{\dot{\boldsymbol{x}}},t+h)}{h} = -\xi\mathfrak{f}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}},t)+\frac{\mathfrak{f}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}},t)}{h}. \tag{4.73}$$

To create sparse vectors, equation (4.73) is expanded by $\mathfrak{f}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}}+h\underline{\dot{\boldsymbol{x}}},t+h)/h$. In the resulting equation (4.74)-(4.75) the unknown $\underline{\dot{\boldsymbol{w}}}$ appears only on the left side

$$lhs := \frac{\mathfrak{f}(\underline{\boldsymbol{w}}+h\underline{\dot{\boldsymbol{w}}},\underline{\boldsymbol{x}}+h\underline{\dot{\boldsymbol{x}}},t+h)-\mathfrak{f}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}}+h\underline{\dot{\boldsymbol{x}}},t+h)}{h} \tag{4.74}$$
$$rhs := \frac{(1-\xi h)\,\mathfrak{f}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}},t)-\mathfrak{f}(\underline{\boldsymbol{w}},\underline{\boldsymbol{x}}+h\underline{\dot{\boldsymbol{x}}},t+h)}{h}. \tag{4.75}$$

The solution of the equation system can be calculated via *FDGMRES*. *FDGMRES* is the nonlinear implementation of the Krylov-space solver *GMRES*. It avoids matrix-matrix multiplications and therefore exploits the sparsity of the *rhs* and *lhs*. The result is a fast convergence to a sufficient approximation of $\underline{\boldsymbol{w}}$. According to the *lhs* (4.74), each computation of the *lhs* requires only two evaluations of $\mathfrak{f}$. Finally the update of $\underline{\boldsymbol{w}}$ results to

$$\underline{\boldsymbol{w}}_{k+1} = \underline{\boldsymbol{w}}_k + \Delta t\cdot\underline{\dot{\boldsymbol{w}}}_k \quad k = \mathbb{N}^+. \tag{4.76}$$

The resulting algorithm is schematically displayed in Table 3.

Algorithm 3: Sequence of used *C/GMRES* configuration

```
while Time loop $t_k = t_{k-1} + \Delta t \;\; k \in \mathbb{N}^+$ do
    Measure present states $\underline{\boldsymbol{x}}(t_k) = \boldsymbol{x}(t_k)$
    while Iterative solution of OCP $j \in \mathbb{N}^+$ do
        Forward integration with explicit Euler-integrator of equation 4.24, 4.50
        $\underline{\boldsymbol{x}}_k^j(\tau) = \int_{t_k}^{t_k+T} f\left(\underline{\boldsymbol{x}}_k^{j-1}, \underline{\boldsymbol{u}}_k^j, \tau\right) \partial\tau$
        Backward integration with explicit Euler-integrator of equation 4.24, 4.50
        $\underline{\boldsymbol{\lambda}}_k^j(\tau) = \nabla_{\underline{\boldsymbol{x}}} V(t_k + T) - \int_{t_k+T}^{t_k} \nabla_{\underline{\boldsymbol{x}}} \mathcal{L}\left(\underline{\boldsymbol{x}}_k^j, \underline{\boldsymbol{w}}_k^j, \tau\right) d\tau$
        Calculate optimal $\dot{\underline{\boldsymbol{w}}}^j$ with FDGMRES method by solving
        $\frac{\mathfrak{f}\left(\underline{\boldsymbol{w}}^j + h\dot{\underline{\boldsymbol{w}}}^j, \underline{\boldsymbol{x}}^j + h\dot{\underline{\boldsymbol{x}}}^j, t+h\right) - \mathfrak{f}\left(\underline{\boldsymbol{w}}^j, \underline{\boldsymbol{x}}^j + h\dot{\underline{\boldsymbol{x}}}^j, t+h\right)}{h} = \frac{(1-\xi h)\mathfrak{f}\left(\underline{\boldsymbol{w}}^j, \underline{\boldsymbol{x}}^j, t\right) - \mathfrak{f}\left(\underline{\boldsymbol{w}}^j, \underline{\boldsymbol{x}}^j + h\dot{\underline{\boldsymbol{x}}}^j, t+h\right)}{h}$
        Update control trajectory
        $\underline{\boldsymbol{w}}_k^{j+1} = \underline{\boldsymbol{w}}_k^j + \Delta t \cdot \dot{\underline{\boldsymbol{w}}}_k^j$
    end
    $\underline{\boldsymbol{w}}_k = \underline{\boldsymbol{w}}_k^{j+1}$
    Apply control $\underline{\boldsymbol{w}}_k(t_k) \rightarrow \boldsymbol{w}(t_k)$
end
```

4.3.4 CMSCGMRES

The stability of *C/GMRES* presented in §4.3.3 can be increased with a multiple shooting extension [SOD06]. This is connected with a higher computational effort per iteration, but is advantageous in terms of convergence and stabilization of nonlinear systems. The idea of multiple shooting is not to integrate the states consecutively from $\underline{\boldsymbol{x}}(t)$ to $\underline{\boldsymbol{x}}(t+T)$, but instead to integrate from every state $\underline{\boldsymbol{x}}(\tau_l)$ only one step. l is hereby the iterator within the prediction horizon. Subsequently, the states are then updated according to the condition for a continuous state trajectory:

$$0 = \underline{\boldsymbol{x}}^j(\tau_{l+1}) - \left(\underline{\boldsymbol{x}}^j(\tau_l) + \int_{\tau_l}^{\tau_l+\Delta\tau} \boldsymbol{f}\left(\underline{\boldsymbol{x}}(\tau), \underline{\boldsymbol{u}}(\tau), \tau\right) d\tau\right) \quad l \in \mathbb{N}^+. \tag{4.77}$$

The index l is hereby indicating the position within the prediction horizon. According to the duality of the system, this continuity condition can also be derived for λ. Under use of the concatenated state and adjoint state vector over the horizon

$$\underline{\boldsymbol{\varpi}}(t) := \left[\underline{\boldsymbol{x}}_1^{\mathrm{T}}(t) \quad \underline{\boldsymbol{\lambda}}_1^{\mathrm{T}}(t) \quad \ldots \quad \underline{\boldsymbol{x}}_N^{\mathrm{T}}(t) \quad \underline{\boldsymbol{\lambda}}_N^{\mathrm{T}}(t)\right] \in \mathbb{R}^{2n_{\underline{\boldsymbol{x}}} \times N}, \tag{4.78}$$

the continuity conditions can be composed accordingly

$$\mathbf{0} = \mathfrak{g}\left(\underline{\boldsymbol{w}}\left(t\right), \underline{\boldsymbol{\varpi}}\left(t\right), \underline{\boldsymbol{x}}\left(t\right), t\right) = \begin{bmatrix} \underline{\boldsymbol{x}}_1 - \left(\underline{\boldsymbol{x}}_0 + \Delta\tau f\left(\underline{\boldsymbol{x}}_0, \underline{\boldsymbol{w}}_0\right)\right) \\ \underline{\boldsymbol{\lambda}}_1 - \left(\underline{\boldsymbol{\lambda}}_2 - \frac{\partial \mathcal{L}}{\partial \underline{\boldsymbol{x}}}\left(\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{\lambda}}_1, \underline{\boldsymbol{w}}_2\right)^{\mathrm{T}}\right) \\ \vdots \\ \underline{\boldsymbol{x}}_N - \left(\underline{\boldsymbol{x}}_{N-1} + \Delta\tau f\left(\underline{\boldsymbol{x}}_{N-1}, \underline{\boldsymbol{w}}_{N-1}\right)\right) \\ \underline{\boldsymbol{\lambda}}_{N-1} - \left(\underline{\boldsymbol{\lambda}}_N - \frac{\partial \mathcal{L}}{\partial \underline{\boldsymbol{x}}}\left(\underline{\boldsymbol{x}}_{N-1}, \underline{\boldsymbol{\lambda}}_{N-1}, \underline{\boldsymbol{w}}_N\right)^{\mathrm{T}}\right) \\ \underline{\boldsymbol{\lambda}}_N - V\left(\underline{\boldsymbol{x}}_N, \underline{\boldsymbol{w}}_N\right) \end{bmatrix}. \tag{4.79}$$

This has the advantage that the initial state is not integrated over the whole horizon (Single shooting), but system is always integrated with respect to the previous prediction. As a result the numerical stability is higher for nonlinear systems. To use this technique with the *C/GMRES* method, the continuation method (equation 4.68-4.69) is extended [SOD06] to

$$\dot{\mathfrak{f}} = -\xi_{\underline{\boldsymbol{w}}}\mathfrak{f} \tag{4.80}$$

$$\dot{\mathfrak{g}} = -\xi_{\underline{\boldsymbol{\varpi}}}\mathfrak{g}. \tag{4.81}$$

The idea here is that the state prediction error shall also converge to zero. However, the additional solving of (4.81) leads to a larger problem dimension and a related high computation time. The idea of condensing is thus to combine (4.80)-(4.81) to reduce the system order. Starting with the partial derivative formulation of (4.81)-(4.81)

$$\mathfrak{f}_{\underline{\boldsymbol{w}}}\dot{\underline{\boldsymbol{w}}} + \mathfrak{f}_{\underline{\boldsymbol{\varpi}}}\dot{\underline{\boldsymbol{\varpi}}} + \mathfrak{f}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} + \mathfrak{f}_t = \xi_{\underline{\boldsymbol{w}}}\mathfrak{f} \tag{4.82}$$

$$\mathfrak{g}_{\underline{\boldsymbol{w}}}\dot{\mathfrak{g}} + \mathfrak{g}_{\underline{\boldsymbol{\varpi}}}\dot{\underline{\boldsymbol{\varpi}}} + \mathfrak{g}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} + \mathfrak{g}_t = \xi_{\underline{\boldsymbol{\varpi}}}\mathfrak{g}, \tag{4.83}$$

$\dot{\underline{\boldsymbol{\varpi}}}$ can be isolated in (4.83):

$$\dot{\underline{\boldsymbol{\varpi}}} = \mathfrak{g}_{\underline{\boldsymbol{\varpi}}}^{-1}\left(\xi_{\underline{\boldsymbol{\varpi}}}\mathfrak{g} - \mathfrak{g}_{\underline{\boldsymbol{w}}}\dot{\underline{\boldsymbol{w}}} - \mathfrak{g}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} - \mathfrak{g}_t\right). \tag{4.84}$$

Inserting (4.84) in (4.82)

$$\mathfrak{f}_{\underline{\boldsymbol{w}}}\dot{\underline{\boldsymbol{w}}} + \mathfrak{f}_{\underline{\boldsymbol{\varpi}}}\mathfrak{g}_{\underline{\boldsymbol{\varpi}}}^{-1}\left(\xi_{\underline{\boldsymbol{\varpi}}}\mathfrak{g} - \mathfrak{g}_{\underline{\boldsymbol{w}}}\dot{\underline{\boldsymbol{w}}} - \mathfrak{g}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} - \mathfrak{g}_t\right) + \mathfrak{f}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} + \mathfrak{f}_t = \xi_{\underline{\boldsymbol{w}}}\mathfrak{f} \tag{4.85}$$

To use the *GMRES* method efficiently, the right-hand side has to be independent from the variable to be determined. For this reason, all terms with the desired update variable $\dot{\underline{\boldsymbol{w}}}$ in (4.86) are isolated to the left-hand side

$$\left(\mathfrak{f}_{\underline{\boldsymbol{w}}} + \mathfrak{f}_{\underline{\boldsymbol{\varpi}}}\mathfrak{g}_{\underline{\boldsymbol{\varpi}}}^{-1}\mathfrak{g}_{\underline{\boldsymbol{w}}}\right)\dot{\underline{\boldsymbol{w}}} = \xi_{\underline{\boldsymbol{w}}}\mathfrak{f} - \mathfrak{f}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} - \mathfrak{f}_t - \mathfrak{f}_{\underline{\boldsymbol{\varpi}}}\mathfrak{g}_{\underline{\boldsymbol{\varpi}}}^{-1}\left(\xi_{\underline{\boldsymbol{\varpi}}}\mathfrak{g} - \mathfrak{g}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} - \mathfrak{g}_t\right). \tag{4.86}$$

The result is a nonlinear correspondence to the linear input affine form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.87}$$

which can be solved with *FDGMRES*.

Left-hand side

The main issue of (4.86) is the calculation of $\boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{\varpi}}}^{-1}$. To address this issue, first only the left-hand side is considered and the dummy vector $\boldsymbol{\sigma}$ is introduced:

$$\text{lhs} = \boldsymbol{\mathfrak{f}}_{\underline{\boldsymbol{w}}}\underline{\dot{\boldsymbol{w}}} + \boldsymbol{\mathfrak{f}}_{\underline{\boldsymbol{\varpi}}}\boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{\varpi}}}^{-1}\boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{w}}}\underline{\dot{\boldsymbol{w}}} \tag{4.88}$$

$$\text{lhs} = \boldsymbol{\mathfrak{f}}_{\underline{\boldsymbol{w}}}\underline{\dot{\boldsymbol{w}}} + \boldsymbol{\mathfrak{f}}_{\underline{\boldsymbol{\varpi}}}\boldsymbol{\sigma} \tag{4.89}$$

$$\text{with } \boldsymbol{\sigma} = \boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{\varpi}}}^{-1}\boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{w}}}\underline{\dot{\boldsymbol{w}}}. \tag{4.90}$$

As the inversion of matrix $\boldsymbol{\mathfrak{g}}$ in (4.90) is expensive, $\boldsymbol{\sigma}$ is approximated with a forward approximation quotient. For this reason, (4.90) is reformulated

$$\boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{\varpi}}}\boldsymbol{\sigma} = \boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{w}}}\underline{\dot{\boldsymbol{w}}} \tag{4.91}$$

$$\boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{w}}}\underline{\dot{\boldsymbol{w}}} - \boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{\varpi}}}\boldsymbol{\sigma} = 0. \tag{4.92}$$

This can be approximated with the forward difference quotient

$$\boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{w}}}\underline{\dot{\boldsymbol{w}}} - \boldsymbol{\mathfrak{g}}_{\underline{\boldsymbol{\varpi}}}\boldsymbol{\sigma} \approx \frac{\boldsymbol{\mathfrak{g}}\left(\underline{\boldsymbol{w}} + h\underline{\dot{\boldsymbol{w}}}, \underline{\boldsymbol{\varpi}} + h\boldsymbol{\sigma}, \underline{\boldsymbol{x}} + h\underline{\dot{\boldsymbol{x}}}, t+h\right) - \boldsymbol{\mathfrak{g}}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}} + h\underline{\dot{\boldsymbol{x}}}, t+h\right)}{h}, \tag{4.93}$$

which inserted in (4.92) leads to the relation

$$\boldsymbol{\mathfrak{g}}\left(\underline{\boldsymbol{w}} + h\underline{\dot{\boldsymbol{w}}}, \underline{\boldsymbol{\varpi}} + h\boldsymbol{\sigma}, \underline{\boldsymbol{x}} + h\underline{\dot{\boldsymbol{x}}}, t+h\right) = \boldsymbol{\mathfrak{g}}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}} + h\underline{\dot{\boldsymbol{x}}}, t+h\right). \tag{4.94}$$

(4.94) can be solved iteratively at each control input computation iteration. The result has still to be inserted into (4.89). By introducing a dummy vector $\boldsymbol{\sigma}$, (4.89) can be formulated with a forward difference quotient. This avoids the explicit computation of the partial derivatives of $\boldsymbol{\mathfrak{f}}$

$$\text{lhs} = \boldsymbol{\mathfrak{f}}_{\underline{\boldsymbol{w}}}\underline{\dot{\boldsymbol{w}}} + \boldsymbol{\mathfrak{f}}_{\underline{\boldsymbol{\varpi}}}\boldsymbol{\sigma} \approx \frac{\boldsymbol{\mathfrak{f}}\left(\underline{\boldsymbol{w}} + h\underline{\dot{\boldsymbol{w}}}, \underline{\boldsymbol{\varpi}} + h\boldsymbol{\sigma}, \underline{\boldsymbol{x}} + h\underline{\dot{\boldsymbol{x}}}, t+h\right) - \boldsymbol{\mathfrak{f}}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}} + h\underline{\dot{\boldsymbol{x}}}, t+h\right)}{h}. \tag{4.95}$$

Here, (4.95) is dependent on the next state approximation $\underline{\boldsymbol{\varpi}} + h\boldsymbol{\sigma}$ which has to be eliminated. Before, $\boldsymbol{\mathfrak{g}}$ has been defined as the state integration difference ζ for each point on the time grid within the horizon. Therefore the next state can be approximated by integrating the system and adding this error ζ. This is achieved by the mapping

$$\underline{\boldsymbol{\varpi}} + h\boldsymbol{\sigma} = \mathfrak{f}\left(\underline{\boldsymbol{w}}, \zeta, \underline{\boldsymbol{x}}, t\right) = \begin{bmatrix} \underline{\boldsymbol{x}}_k + hf\left(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{u}}_k\right) + \boldsymbol{e}_{\underline{\boldsymbol{x}},k+1} \\ \underline{\boldsymbol{\lambda}}_{k+2} - h\frac{\partial \mathcal{L}}{\partial \underline{\boldsymbol{x}}}\left(\underline{\boldsymbol{x}}_{k+1}, \underline{\boldsymbol{\lambda}}_{k+2}, \underline{\boldsymbol{u}}_{k+1}, \underline{\boldsymbol{\lambda}_{\boldsymbol{eq}}}_{k+1}\right) + \boldsymbol{e}_{\underline{\boldsymbol{\lambda}},k+1} \\ \vdots \\ V\left(\underline{\boldsymbol{x}}_N\right) + \boldsymbol{e}_{\underline{\boldsymbol{\lambda}},N} \end{bmatrix}. \tag{4.96}$$

In (4.95) $\underline{\boldsymbol{\varpi}}+h\boldsymbol{\sigma}$ appears in the context of the approximation with $\underline{\boldsymbol{w}}+h\dot{\underline{\boldsymbol{w}}}$, $\underline{\boldsymbol{x}}+h\dot{\underline{\boldsymbol{x}}}$, $t+h$ in $\mathfrak{f}$. Accordingly the following integration error has to be found

$$\zeta = \mathfrak{g}\left(\underline{\boldsymbol{w}} + h\dot{\underline{\boldsymbol{w}}}, \underline{\boldsymbol{\varpi}} + h\boldsymbol{\sigma}, \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t + h\right) \tag{4.97}$$

which appears in relation (4.94). As a consequence, (4.94) can be inserted into (4.96) and subsequentially (4.95). This finally results to the left-hand side:

$$\begin{aligned} \text{lhs} := &-\frac{\mathfrak{f}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t + h\right)}{h} \\ &+ \frac{\mathfrak{f}\left(\underline{\boldsymbol{w}} + h\dot{\underline{\boldsymbol{w}}}, \mathfrak{f}\left(\underline{\boldsymbol{w}} + h\dot{\underline{\boldsymbol{w}}}, \mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t + h\right), \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t + h\right), \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t + h\right)}{h} \end{aligned} \tag{4.98}$$

Right-hand side

Going back to the right-hand side of (4.86)

$$\text{rhs} := \xi_{\underline{\boldsymbol{\varpi}}}\mathfrak{f} - \mathfrak{f}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} - \mathfrak{f}_t - \mathfrak{f}_{\underline{\boldsymbol{\varpi}}}\mathfrak{g}_{\underline{\boldsymbol{\varpi}}}^{-1}\left(\xi\mathfrak{g} - \mathfrak{g}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} - \mathfrak{g}_t\right). \tag{4.99}$$

which again contains an inverse expression that can be approximated by introducing a dummy $\boldsymbol{\sigma}$. This is a new dummy variable and not connected with the dummy $\boldsymbol{\sigma}$ introduced for the left-hand side!

$$\text{rhs} := -\xi_{\underline{\boldsymbol{w}}}\mathfrak{f} - \mathfrak{f}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} - \mathfrak{f}_t - \mathfrak{f}_{\underline{\boldsymbol{\varpi}}}\boldsymbol{\sigma} \tag{4.100}$$

$$\text{with } \boldsymbol{\sigma} = \mathfrak{g}_{\underline{\boldsymbol{\varpi}}}^{-1}\left(\xi_{\underline{\boldsymbol{\varpi}}}\mathfrak{g} - \mathfrak{g}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} - \mathfrak{g}_t\right). \tag{4.101}$$

Building the products in (4.101) leads to

$$\mathfrak{g}_{\underline{\boldsymbol{\varpi}}}\boldsymbol{\sigma} + \mathfrak{g}_{\underline{\boldsymbol{x}}}\dot{\underline{\boldsymbol{x}}} - \mathfrak{g}_t = -\xi_{\underline{\boldsymbol{\varpi}}}\mathfrak{g} \tag{4.102}$$

which can be approximated with the forward difference to

$$\frac{\mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}} + h\boldsymbol{\sigma}, \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t + h\right) - \mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}}, t\right)}{h} \approx -\xi_{\underline{\boldsymbol{\varpi}}}\mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}}, t\right). \tag{4.103}$$

(4.103) can be reformulated to

$$\mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}} + h\boldsymbol{\sigma}, \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\right) = \left(1 - h\xi_{\underline{\boldsymbol{\varpi}}}\right) \mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}}, t\right). \tag{4.104}$$

This can be inserted in the forward difference approximation of equation (4.100)

$$\text{rhs} :\approx \qquad -\xi_{\underline{\boldsymbol{w}}}\mathfrak{f}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}}, t\right) - \frac{\mathfrak{f}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}} + h\boldsymbol{\varphi}, \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\right) - \mathfrak{f}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}}, t\right)}{h} \tag{4.105}$$

Similarly to the left-hand side, the mapping $\mathfrak{f}$ is used to determine $\underline{\boldsymbol{\varpi}} + h\boldsymbol{\varphi}$.

$$\underline{\boldsymbol{\varpi}} + h\boldsymbol{\varphi} = \mathfrak{f}\left(\underline{\boldsymbol{w}}, \mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}} + h\boldsymbol{\varphi}, \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\right), \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\right). \tag{4.106}$$

As a remark, the difference to the *lhs* is that there is no partial derivative $\mathfrak{f}_{\underline{\boldsymbol{w}}}$ on the right-hand side, and thus also no $\underline{\boldsymbol{w}} + h\dot{\underline{\boldsymbol{w}}}$. Finally, (4.104) is inserted into (4.106) to get rid of $\underline{\boldsymbol{\varpi}} + h\boldsymbol{\varphi}$. With another insertion into (4.105), this results to the full right-hand side:

$$\begin{aligned}\text{rhs} :\approx &\left(\frac{1}{h} - \xi_{\underline{\boldsymbol{w}}}\right)\mathfrak{f}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}}, t\right)\\ &- \frac{\mathfrak{f}\big(\underline{\boldsymbol{w}}, \mathfrak{f}\big(\underline{\boldsymbol{w}}, \big(1 - h\xi_{\underline{\boldsymbol{\varpi}}}\big)\, \mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}}, t\right), \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\big), \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\big)}{h}\end{aligned} \tag{4.107}$$

To finally solve the complete condensed multiple shooting problem, the combination of left-hand side (4.98) and right-hand side (4.107) are solved together:

$$\begin{aligned}\mathfrak{f}\left(\underline{\boldsymbol{w}} + h\dot{\underline{\boldsymbol{w}}}, \mathfrak{f}\left(\underline{\boldsymbol{w}} + h\dot{\underline{\boldsymbol{w}}}, \mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\right), \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\right), \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\right)&\\ -\mathfrak{f}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\right) = \big(1 - h\xi_{\underline{\boldsymbol{w}}}\big)\,\mathfrak{f}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}}, t\right)&\\ -\mathfrak{f}\big(\underline{\boldsymbol{w}}, \mathfrak{f}\big(\underline{\boldsymbol{w}}, \big(1 - h\xi_{\underline{\boldsymbol{\varpi}}}\big)\,\mathfrak{g}\left(\underline{\boldsymbol{w}}, \underline{\boldsymbol{\varpi}}, \underline{\boldsymbol{x}}, t\right), \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\big), \underline{\boldsymbol{x}} + h\dot{\underline{\boldsymbol{x}}}, t+h\big).&\end{aligned} \tag{4.108}$$

4.4 NMPC Benchmark

To assess the potential of the *NMPC* solvers introduced in §4.3, they are used to control the physical model (3.14) in a simulative environment. The scenario is simulating a closed-loop state (3.1) change from the initial ($\boldsymbol{x}(0)$, $\boldsymbol{u}(0)$) to the desired state ($\boldsymbol{x}_{des}$, $\boldsymbol{u}_{des}$)

$$\boldsymbol{x}(0) = (1, -1, 1, 0, 0, 0, 0, 0, 0.5\pi, 0, 0, 0)^{\mathrm{T}} \tag{4.109}$$

$$\boldsymbol{u}(0) = (9.81, 0, 0, 0)^{\mathrm{T}} \tag{4.110}$$

$$\boldsymbol{x}_{des} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^{\mathrm{T}} \tag{4.111}$$

$$\boldsymbol{u}_{des} = (9.81, 0, 0, 0)^{\mathrm{T}} \tag{4.112}$$

$$\boldsymbol{Q} = \mathrm{diag}\left(\left[10, 10, 10, 0, 0, 0, 1, 1, 1, 0, 0, 0\right]\right) \tag{4.113}$$

$$\boldsymbol{R} = \mathrm{diag}\left(\left[0.1, 0.1, 0.1, 0.1\right]\right) \tag{4.114}$$

using the trajectory error penalty ($\boldsymbol{Q}$, $\boldsymbol{R}$). The parameters of model (3.14) are chosen in the scenario to $g = 9.81\,\mathrm{m\,s^{-1}}$, $m = 1\,\mathrm{kg}$, $\iota_x = 1\,\mathrm{kg\,m^2}$, $\iota_y = 1\,\mathrm{kg\,m^2}$ and $\iota_z = 0.5\,\mathrm{kg\,m^2}$. The *NMPC* solvers are applied using the same prediction horizon length $T = 2\,\mathrm{s}$, horizon discretization $N = 20\,\mathrm{s} \leftrightarrow \Delta\tau = 0.1\,\mathrm{s}$ and control update interval $\Delta t = 0.1\,\mathrm{s}$. The inherent *OCP* with the physical model (3.14) accordingly results to

$$\min_{\underline{\boldsymbol{u}}(\cdot)} = \int_{t}^{t+T} (\boldsymbol{x}_{des} - \underline{\boldsymbol{x}}(\tau))^{\mathrm{T}} \boldsymbol{Q} (\boldsymbol{x}_{des} - \underline{\boldsymbol{x}}(\tau)) + (\boldsymbol{u}_{des} - \underline{\boldsymbol{u}}(\tau))^{\mathrm{T}} \boldsymbol{R} (\boldsymbol{u}_{des} - \underline{\boldsymbol{u}}(\tau))\, \mathrm{d}\tau$$

$$\text{s.t.}\ \dot{\underline{\boldsymbol{x}}}(\tau) = \begin{bmatrix} {}^{\mathcal{G}}\dot{\underline{x}} \\ {}^{\mathcal{G}}\dot{\underline{y}} \\ {}^{\mathcal{G}}\dot{\underline{z}} \\ {}^{\mathcal{B}}\underline{\Gamma}_z(\cos{}^{\mathcal{V}2}\underline{\Phi}\cos{}^{\mathcal{V}}\underline{\Psi}\sin{}^{\mathcal{V}1}\underline{\Theta} + \sin{}^{\mathcal{V}2}\underline{\Phi}\sin{}^{\mathcal{V}}\underline{\Psi}) \\ {}^{\mathcal{B}}\underline{\Gamma}_z(-\cos{}^{\mathcal{V}}\underline{\Psi}\sin{}^{\mathcal{V}2}\underline{\Phi} + \cos{}^{\mathcal{V}2}\underline{\Phi}\sin{}^{\mathcal{V}1}\underline{\Theta}\sin{}^{\mathcal{V}}\underline{\Psi}) \\ -9.81 + {}^{\mathcal{B}}\underline{\Gamma}_z\cos{}^{\mathcal{V}1}\underline{\Theta}\cos{}^{\mathcal{V}2}\underline{\Phi} \\ {}^{\mathcal{B}}\underline{\varrho} + {}^{\mathcal{B}}\underline{\omega}\cos{}^{\mathcal{V}2}\underline{\Phi}\tan{}^{\mathcal{V}1}\underline{\Theta} + {}^{\mathcal{B}}\underline{\vartheta}\sin{}^{\mathcal{V}2}\underline{\Phi}\tan{}^{\mathcal{V}1}\underline{\Theta} \\ {}^{\mathcal{B}}\underline{\vartheta}\cos{}^{\mathcal{V}2}\underline{\Phi} - {}^{\mathcal{B}}\underline{\omega}\sin{}^{\mathcal{V}2}\underline{\Phi} \\ ({}^{\mathcal{B}}\underline{\omega}\cos{}^{\mathcal{V}2}\underline{\Phi} + {}^{\mathcal{B}}\underline{\vartheta}\sin{}^{\mathcal{V}2}\underline{\Phi})\sec{}^{\mathcal{V}1}\underline{\Theta} \\ {}^{\mathcal{B}}\underline{\Lambda}_x + 0.5{}^{\mathcal{B}}\underline{\vartheta}{}^{\mathcal{B}}\underline{\omega} \\ {}^{\mathcal{B}}\underline{\Lambda}_y - 0.5{}^{\mathcal{B}}\underline{\varrho}{}^{\mathcal{B}}\underline{\omega} \\ \frac{1}{0.5}{}^{\mathcal{B}}\underline{\Lambda}_z \end{bmatrix}$$

$$\underline{\boldsymbol{x}}(t) = \boldsymbol{x}(t)\,.$$

Under the defined conditions, no stable parametrization for *C/GMRES* could be found. For this reason, this benchmark comprises the simulation results for *GRAMPC*, *CMSC/GMRES* and the *ACADO* inherent multiple shooting *SQP* with Gauß-Newton approximation of the Hessian. The solver specific parametrizations are given in Table 4.3 in the solver related nomenclature. If not explicitly given, the default solver properties are used. As each solver provides different integration schemes, the computationally least demanding scheme which provides a stable solution is used. Table 4.3 demonstrates the extent and complexity of *NMPC* solvers and their parametrization. Due to this complexity, a qualitative comparison in detail could not be conducted in the context of this work. Important properties are hereby the numerical stability, convergence and computational effort. For this reason, the goal of this benchmark scenario is not the qualitative analysis, but the demonstration of feasibility and potential of the considered solvers.

GRAMPC					
Nmaxit(i_{max})	=	100	ShiftControl	=	on
ScaleProblem	=	on	CostIntegrator	=	simpson
Integrator	=	heun	IntegratorRelTol	=	10^{-4}
IntegratorAbsTol	=	10^{-3}	LineSearchType	=	adaptive
LineSearchMax	=	2.0	LineSearchMin	=	10^{-5}
LineSearchInit	=	0.1	LineSearchIntervalFactor	=	0.5
LineSearchAdaptFactor	=	2.0	JacobianX	=	sysjacxadj
LineSearchIntervalTol	=	10^{-3}	JacobianU	=	sysjacuadj
SQP					
INTEGRATOR_TYPE	=	INT_RK4			
DYNAMIC_SENSITIVITY	=	FORWARD_SENSITIVITY			
USE_IMMEDIATE_FEEDBACK	=	YES			
HESSIAN_APPROXIMATION	=	GAUSS_NEWTON			
DISCRETIZATION_TYPE	=	MULTIPLE_SHOOTING			
CMSC/GMRES					
Integrator	=	Explicite Euler	alpha(υ)	=	1.5
zeta(ξ)	=	10	hdir(h)	=	10^{-3}
rtol(ϵ)	=	10^{-5}	kmax(i_{max})	=	5

Table 4.3: *NMPC* solver parametrization for benchmark scenario, given in the solver inherent nomenclature

The resulting system trajectories are given in Figure 4.2-4.4. Figure 4.2 shows the *UAV* pose displacement, while Figure 4.3 is giving the related velocities. The *UAV*'s computed force and torque inputs are plotted in Figure 4.4. All trajectories show the desired convergence from the initial states (4.109) towards the desired system states

(4.111). This is further confirmed by the decrease of the cost function J in Figure 4.4f.

Regarding the optimality, the suboptimal solution and limited horizon length lead to small overshoot which is visible e.g. in the $y_{\mathcal{G}}$ position in Figure 4.2b at $t \approx 2\,\mathrm{s}$. Furthermore, in the initial phase, the *SQP* and *CMSC/GMRES* solutions are showing significant suboptimal behavior in the form of discontinuous trajectory peaks. These peaks are clearly visible, for example in the $\vartheta_{\mathcal{B}}$ trajectory shown in Figure 4.3e. This suboptimal behavior is caused by the limited solver iterations, sampling times and solution tolerance. From the three configured solvers, *CMSC/GMRES* shows the least optimal behavior. However, setting this has to be considered in relation with the computation time t_c. With a maximum computation time $max(t_c) = 0.28\,\mathrm{ms}$ and an average computation time of $\overline{t_c} = 0.17\,\mathrm{ms}$ per control update interval, *CMSC/GMRES* outperforms *GRAMPC* with $max(t_c) = 14.0\,\mathrm{ms}$, $\overline{t_c} = 8.71\,\mathrm{ms}$ as well as *SQP* with $max(t_c) = 168.025\,\mathrm{ms}$, $\overline{t_c} = 146.56\,\mathrm{ms}$. The corresponding computation time for each control update interval are displayed in Figure 4.4e. These computation times demonstrate the potential of *CMSC/GMRES*. However, they are not representative for neither *SQP* nor *GRAMPC* for the following reasons. For the *SQP* evaluation, the standard *ACADO* implementation has been used. As the framework is very complex and does include many algorithmic options, the code is not lightweight. An isolated efficient implementation using only the utilized solver and integration scheme is supposed to lead to significantly lower computation times. In addition, the *SQP* algorithm is using higher order integrators which allow reducing the amount of samples within the horizon. The related reduction of optimization variables permits significant lower computation times, while maintaining stability. For example, also with $N = 5$ a stable behavior is achieved resulting in $max(t_c) = 36.21\,\mathrm{ms}$, $\overline{t_c} = 31.77\,\mathrm{ms}$. The number of iterations is directly depending on the given solution tolerance. Accordingly, also the computation time of *GRAMPC* could also be further reduced using a lower solution optimality tolerance. Nevertheless, both algorithms do not show sub-ms performance without significant additional implementation effort. For this reason, *CMSC/GMRES* is used as *NMPC*-solver within this thesis.

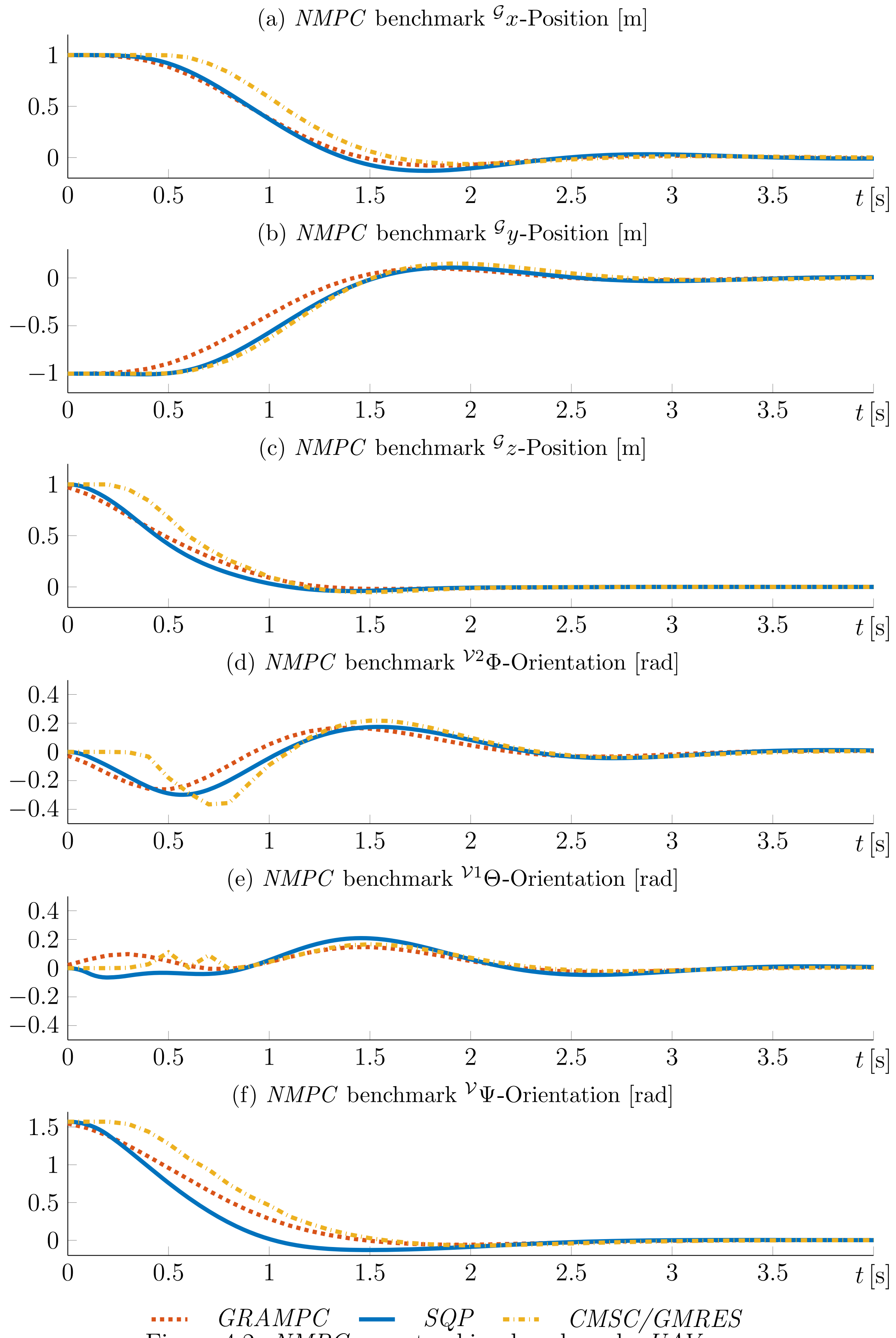


Figure 4.2: *NMPC* pose tracking benchmark: *UAV* pose

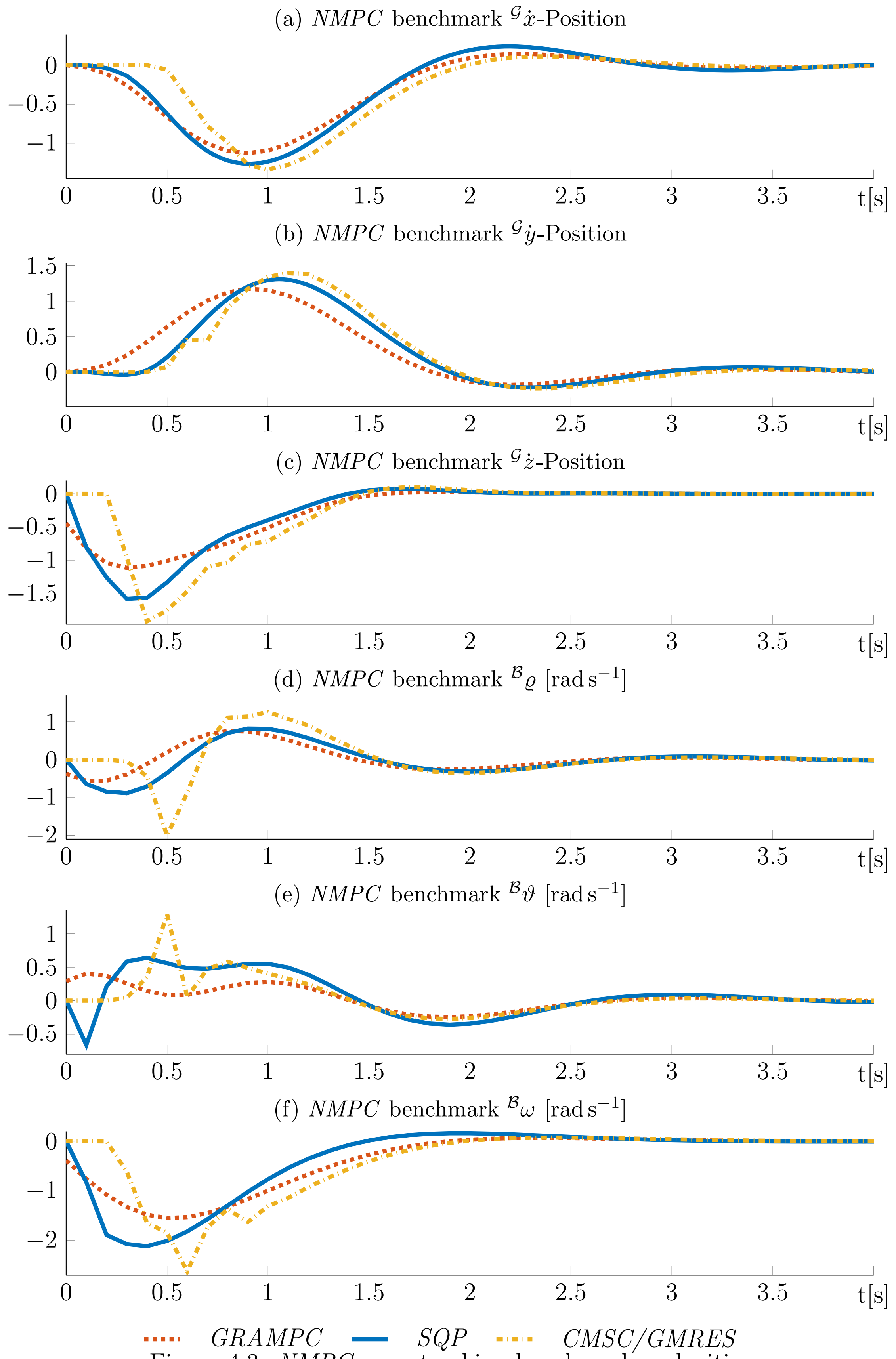


Figure 4.3: *NMPC* pose tracking benchmark: velocities

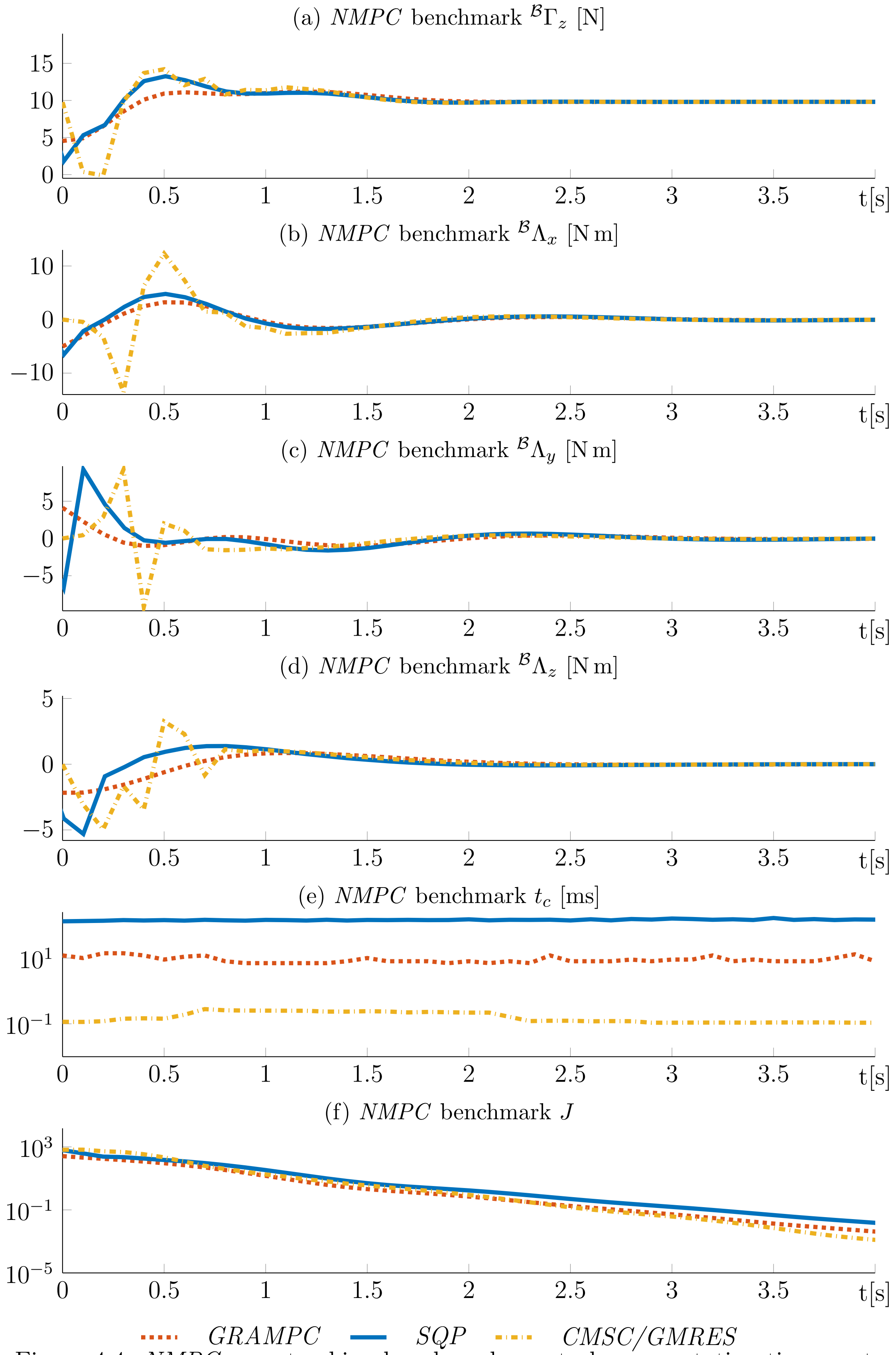


Figure 4.4: *NMPC* pose tracking benchmark: controls, computation time, costs

4.5 Constraint handling

One major advantage of *MPC* in comparison to regular control strategies is the ability to consider constraints. While equality constraints are typically treated analogue to the system dynamic constraint, the treatment of inequality constraints within the *MPC* is not trivial. For this reason, this chapter gives a short introduction on several constraint handling strategies which have been used in the context of this thesis. Due to the fact that *OCP*s are typically defined as minimization problems with the optimum $J = 0$, the idea is to impose additional costs for a constraint violation.

4.5.1 Primal barrier method

Considering the static optimization problem (4.7)-(4.9), inequality constraints can be treated by smoothing the complementary slackness condition (4.14) in the *KKT*-optimality conditions. An example for such a simple interior point method is the primal barrier method that applies a smoothing constant $q_{c_{in}}$ [Die14]

$$\nabla_{\underline{\boldsymbol{x}}} J(\underline{\boldsymbol{x}}) + \nabla_{\underline{\boldsymbol{x}}} \boldsymbol{c}(\underline{\boldsymbol{x}}) \underline{\boldsymbol{\lambda_{eq}}} + \nabla_{\underline{\boldsymbol{x}}} \boldsymbol{c_{in}}(\underline{\boldsymbol{x}}) \underline{\boldsymbol{\lambda_{in}}} = 0 \tag{4.115}$$

$$\boldsymbol{c}(\underline{\boldsymbol{x}}) = 0 \tag{4.116}$$

$$\lambda_{in,l} c_{in,l}(\underline{\boldsymbol{x}}) + q_{c_{in}l} = 0, \quad l = 1, ..., n_{\boldsymbol{c_{in}}} \tag{4.117}$$

$\underline{\boldsymbol{\lambda_{eq}}}$ are the equality constraint and $\underline{\boldsymbol{\lambda_{in}}}$ respective the inequality constraint multipliers. Isolation of λ_{in} in (4.117) can be directly inserted into (4.115) to

$$\nabla_{\underline{\boldsymbol{x}}} c_{in,l}(\underline{\boldsymbol{x}}) \lambda_{in,l} \rightarrow -q_{c_{in}l} \frac{\nabla_{\underline{\boldsymbol{x}}} c_{in,l}(\underline{\boldsymbol{x}})}{c_{in,l}(\underline{\boldsymbol{x}})}, \quad l = 1, ..., n_{\boldsymbol{c_{in}}}. \tag{4.118}$$

Integrating (4.118) leads to an equivalent representation of an optimization problem, including the inequality relaxation

$$\min_{\underline{\boldsymbol{x}}} J(\underline{\boldsymbol{x}}) - \sum_{l=1}^{n_{\boldsymbol{c_{in}}}} q_{c_{in}l} \log(-c_{in,l}(\underline{\boldsymbol{x}})) \tag{4.119}$$

$$\text{s.t. } \boldsymbol{c}(\underline{\boldsymbol{x}}) = \boldsymbol{0} \tag{4.120}$$

For small $q_{c_{in}}$ the non-smoothness of the inequality is approximated better, but the condition of the optimization problem gets worse. The simple interior point method of (4.118) is a primal barrier method. More sophisticated interior point methods are optimizing this trade-off by adaptation of the relaxation value $q_{c_{in}}$. The main disadvantage of this simple logarithmic barrier method is that violated constraints lead to an infeasible problem. For a violated constraint, the argument of the logarithm becomes negative and is therefore not defined anymore. Therefore, it has to be ensured that the inequality constraints are never violated.

4.5.2 Auxiliary variable method

Another strategy is to implement inequality constraints using a slack variable ν as presented in [Oht04]. This method has been implemented in the *C/GMRES*-Package accessible under [Oht]. The underlying idea is to add a positive variable to an inequality constraint in order to transform it into an equality constraint. To ensure the positiveness of this slack variable, its quadrature is used. The problem is accordingly shifted to the question which ν is fulfilling the equality constraint. This can be solved by taking the ν into account as additional optimization variable. To avoid the singularity for $\nu = 0$ a small displacement is created by a small penalty $-\boldsymbol{r}_\nu^\top \boldsymbol{\nu}$ in the cost function.

$$\min_{\underline{\boldsymbol{u}}} J = \int_{\tau=0}^{T} L\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) - \boldsymbol{r}_\nu^\top \boldsymbol{\nu}\left(\tau\right) d\tau \tag{4.121}$$

$$s.\ t.\ \ \boldsymbol{c}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) = \boldsymbol{0} \tag{4.122}$$

$$c_{in,l}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) + \nu_l\left(\tau\right)^2 = 0, \quad l = 1, ..., n_{\boldsymbol{c_{in}}} \tag{4.123}$$

This *OCP* leads to the Lagrangian

$$\mathcal{L} = L\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}\right) + \underline{\boldsymbol{\lambda}}^\top \boldsymbol{c}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) - \boldsymbol{r}_\nu^\top \nu\left(\tau\right) + \sum_{l=0}^{n_{\boldsymbol{c_{in}}}} \lambda_{in,l}\left(c_{in,l}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) + \underline{\nu}_l^2\right). \tag{4.124}$$

Considering λ_{in} and ν as additional optimization variables, the *KKT*-conditions yield

$$\nabla_{\underline{\boldsymbol{u}}} \mathcal{L} = \nabla_{\underline{\boldsymbol{u}}} L\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}\right) + \underline{\boldsymbol{\lambda}}^\top \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}\right) + \underline{\boldsymbol{\lambda_{in}}}^\top \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c_{in}}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}\right) = \boldsymbol{0} \tag{4.125}$$

$$\frac{\partial \mathcal{L}}{\partial \nu_i} = 2\lambda_{in,l}\nu_l - r_{\nu,l} = 0, \quad l = 1, ..., n_{\boldsymbol{c_{in}}} \tag{4.126}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{in,l}} = c_{in,l}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) + \nu_l^2 = 0, \quad l = 1, ..., n_{\boldsymbol{c_{in}}} \tag{4.127}$$

$$\boldsymbol{c}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) = \boldsymbol{0} \tag{4.128}$$

$$\lambda_{in} \geq \boldsymbol{0}. \tag{4.129}$$

At the point of optimality, ν in (4.127) can be isolated

$$0 = c_{in,l}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) + \nu_l^2 = 0 \quad \Rightarrow \quad \nu_l = \sqrt{-c_{in,l}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right)} \;\; l = 1, ..., n_{\boldsymbol{c_{in}}}. \tag{4.130}$$

Then, (4.130) is inserted into (4.126)

$$2\lambda_{in,l}\nu_l - \boldsymbol{r}_\nu = 0 \quad \Rightarrow \quad \lambda_{in,l} = \frac{\boldsymbol{r}_\nu}{2\sqrt{-c_{in,l}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right)}} \;\; l = 1, ..., n_{\boldsymbol{c_{in}}}. \tag{4.131}$$

Accordingly λ_{in} can be substituted in (4.125) by condition (4.131):

$$\nabla_{\underline{\boldsymbol{u}}} J\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}\right) + \underline{\boldsymbol{\lambda}}^\top \nabla_{\underline{\boldsymbol{u}}} \boldsymbol{c}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}\right) + \sum_{l=1}^{n_{\boldsymbol{c_{in}}}} \frac{\boldsymbol{r}_\nu}{2\sqrt{-c_{in,l}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right)}} \nabla_{\underline{\boldsymbol{u}}} c_{in,l}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}\right) = \boldsymbol{0} \tag{4.132}$$

Integration of (4.132) finally leads the equivalent *OCP* around the optimal point [DFH09]

$$\min_{\underline{\boldsymbol{u}}} J\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}\right) - \sum_{l=1}^{n_{\boldsymbol{c_{in}}}} \boldsymbol{r}_\nu \sqrt{-c_{in,l}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right)} \tag{4.133}$$

$$s.\ t.\ \boldsymbol{c}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) = \boldsymbol{0} \tag{4.134}$$

As a remark, the experimental evaluation has shown a high sensitivity regarding initial ν and penalty r_ν values. This can lead to an unstable parametrization. The result is a negative λ_{in} which is contradicting the first-order optimality conditions. Furthermore, the solver does not necessarily recover from this state.

4.5.3 Saturation function approach

Another inequality constraint handling approach is to approximate the switching behavior of constraint by a saturation function. Due to the fact that *OCP*s are typically defined as minimization problems with the optimum $J = 0$, a constraint violation has to be penalized with a higher cost. The intuitive approach is to use a unit step function ϵ

$$\epsilon\left(c_{in}\right) \begin{cases} 1 & \text{if} \quad c_{in} \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.135}$$

to translate the constraint $c_{in} \leq 0$ to the soft constraint $L_{c_{in}}$

$$c_{in} \leq 0 \quad \Rightarrow \quad L_{c_{in}} = \epsilon\left(c_{in}\right). \tag{4.136}$$

With this approach, a violation of the constraint is penalized with $L = 1$. However, the lack of continuous differentiability at the constraint border $c_{in} = 0$ is problematic for *MPC* solvers. Most fast *OCP* solvers require the differentiability of the cost function. This can be addressed by relaxing the cost step via a saturation function approximation. A popular saturation function is in this context the sigmoid function

$$\epsilon\left(c_{in}\right) \approx sig\left(c_{in}, \kappa_A\right) = \frac{1}{1+e^{-\kappa_A c_{in}}}. \tag{4.137}$$

Here, κ_A can be used to tune the sharpness of the switching behavior. The derivative of $sig\left(c_{in}, \kappa_A\right)$ can be determined analogously

$$\frac{\partial sig\left(c_{in}, \kappa_A\right)}{\partial c_{in}} = \frac{\kappa_A(1+e^{-\kappa_A c_{in}})}{(1+e^{-\kappa_A c_{in}})^2}. \tag{4.138}$$

Figure 4.5 is showing the behavior of the sigmoid function and its derivative for a variation of κ_A.

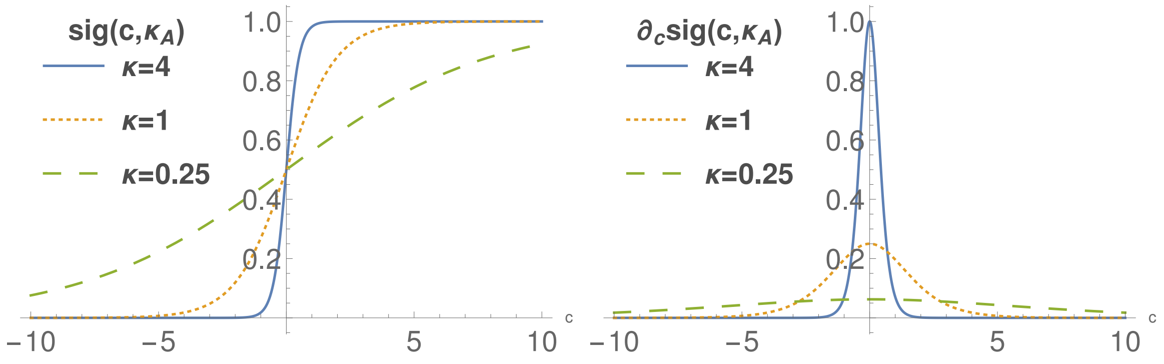


Figure 4.5: Sigmoid approximation of unit step

As desired, $sig\left(c_{in}, \kappa_A\right)$ is converging towards a unit step $\epsilon(c_{in})$ for increasing κ_A. Its derivative is accordingly converging towards a δ impulse

$$\lim_{\kappa_A \to \infty} sig\left(c_{in}, \kappa_A\right) \to \epsilon(c_{in}) \tag{4.139}$$

$$\lim_{\kappa_A \to \infty} \frac{\partial sig\left(c_{in}, \kappa_A\right)}{\partial c_{in}} \to \delta(c_{in}). \tag{4.140}$$

As the transformed inequality constraints are added to the cost-function, the result is a potential function. The transformed *OCP* results to

$$\min_{\underline{\boldsymbol{u}}} \; J = \int\limits_{\tau=0}^{T} L\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) + \sum_{l=1}^{n_{c_{in}}} \kappa_H sig\left(c_{in,l}, \kappa_A\right) d\tau \tag{4.141}$$

$$s.\ t.\ \mathbf{0} = \boldsymbol{c}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right), \tag{4.142}$$

where κ_H is adjusting the height of the penalty value introduced by a constraint violation. Same approaches exist for other saturation functions e.g. tanh [BMPL+14]. The disadvantage of the saturation function is, that constraint violations are possible depending on the cost height and gradient. This relaxation of constraint borders is often referred to as soft constraint and leads to a trade-off between constraint compliance and other goals in the cost-function. There exist more sophisticated saturation function approaches which address this issue by reformulation of the inequality constraints and runtime adjustment of the approximation parameters [GKPC10]. A second disadvantage is, that non-convexity can be introduced into the system. This has to be taken into consideration during the constraint design, as discussed in §5.6. The advantage is, no additional optimization variables are required wherefore the problem dimension and related computational effort stays low. Furthermore, the *OCP* stays feasible even when an inequality constraint would be violated. This makes it particularly well suited in robotic applications with state constraints. To give an example, if a minimum distance between two *UAV*s shall be kept, small violations of a given safety distance are acceptable. However, if the *UAV*s initially are below this safety distance the *OCP* is still feasible and will force the *UAV*s into a compliant distance.

4.6 Stability discussion

There are several approaches to discuss the stability of *MPC*. Prof. Grüne and Prof. Pannek are discussing stability and suboptimality in [GP17], both with and without the use of stabilizing constraints. One discussed approach is to apply equilibrium end-point constraints. The underlying idea is to formulate the desired convergence towards the desired trajectory as additional constraint in the problem [GP17, p.89].

Another very generic approach for *MPC* stability proofs is to prove the input to state stability (*ISS*) [LAR+09]. Here, a $\mathcal{KL}$ function is used to confine the *OCP* solution which proves a convergence towards the optimal solution. The same can be expressed by a $\mathcal{K}_\infty$-bounded (α_1,α_2,α_3) Lyapunov function $\alpha_1(|x|) < V(\boldsymbol{x}) < \alpha_2(|\boldsymbol{x}|)$ in combination with bounded disturbance $V(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\varsigma}) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{0})) < \alpha_2(|\boldsymbol{x}|)$.

The main challenge in applying these stability proof frameworks on real-world application is the combined complexity of the *OCP* solvers, constraint handling methods and system nonlinearities. For the *C/GMRES*, respectively *CMSC/GMRES* algorithm used within this thesis, the boundness of error has been discussed in a very compact form in [Oht04], [SOD09] and [SO10]. This discussion is based on the algebraic analysis of the internally used subalgorithms. One example is *FDGMRES* which is analysed in [Kel99] regarding robustness and numerical properties. In the compact form of the presented *CMSC/GMRES* analysis in [SO10], the extent of not clearly defined variables (e.g. Lipschitz boundaries) complicate the traceability of the algebraic analysis. In the context of this work, it was not possible to reproduce the proof and conditions for a bounded error. As a consequence, no stability proofs are given. Nevertheless, experimental results in the context of this thesis indicate the local stability. To provide easier access to the algorithmic details for future analysis, *CMSC/GMRES* has been explained very detailed in §4.3.4.

4.7 Conclusion

This chapter has introduced the concept of fast *NMPC*. In this context, an overview on the related work has been given. As fast *NMPC* solvers typically rely on the first-order optimality conditions of the inherent *OCP*, the optimality of optimization problems has been discussed in §4.2. This includes static optimization and dynamic optimization problems with equality and inequality constraints.

In search for a lightweight *NMPC* solver which is capable of real-time *UAV* control, four different *NMPC* algorithms have been presented and evaluated. This includes *ACADO*'s *SQP* with Gauß-Newton approximation, the *GRAMPC* gradient-descent, the continuation-based *C/GMRES* method and its multiple shooting derivative *CMSC/GMRES*. In §4.4, all four solvers have been evaluated in a *UAV* pose change scenario in simulation. With $\overline{t_c} = 0.17\,\text{ms}$, respectively $max(t_c) = 0.28\,\text{ms}$ per iteration, *CMSC/GMRES* has shown the lowest computation time in the conducted benchmark. It shall be mentioned, that the given benchmark of computation time and stability does not allow an absolute comparison of the analysed *NMPC* algorithms' performance. This is varying with the scenario and parametrization, such as the control update interval Δt, prediction horizon T, horizon discretization $\Delta\tau$, forward difference step h, the system condition, etc. Nevertheless, the parametrization and scenario have been chosen to demonstrate the potential of the given methods. As a result, *CMSC/GMRES* is the solver of choice for this thesis.

To be able to solve *OCP*s with inequality constraints, a selection of lightweight constraint handling techniques has been introduced in §4.5. While the primal barrier method offers a very intuitive parametrization and very steep cost gradient towards a constraint violation, it does lead to an infeasible *OCP* when violated. Furthermore, the steep gradient at the constraint border leads to a drift towards the compliant regions which leads to a suboptimal problem. For this reason, it is best suited for implementing constraints on variables that are directly influencable for example control limits. The auxiliary variable method has shown to be less intuitive in its parametrization. However, its main disadvantage is that the inequality Lagrange-multiplier and the slack-variable are treated as separate optimization variables. Hence, a discretization of two more variables over the full horizon is considered per constraint. This can be critical in terms of memory usage and computation time. Finally, the saturation function approach is used to transform inequality constraints into potential functions. This transformation is using a saturation function (e.g. sigmoid) in order to approximate the switching behavior of an inequality constraint with a continuously differentiable function. The result is a soft constraint which allows violations according to the parametrization of the approximation. This constraint handling is well suited if minor violations are acceptable. It does not introduce additional optimization variables, wherefore it is computationally lightweight and widely used in this thesis.

In order to use the proposed *NMPC* in safety critical applications, future work will address the algebraic analysis of *CMSC/GMRES* to determine bounds and a proof of stability. In addition, the constraint handling techniques will be researched in terms of convergence and stability properties.

Chapter 5

UAV control

This chapter is dedicated to the *MPC* of *UAV*s which follow velocity reference commands. In combination with an unknown internal controller, this is typical for commercially available *UAV*s. In this context, §5.1 is discussing the related work. §5.2 is assessing the controllability of a *UAV* based on the developed *HMDV* (3.24). The *UAV NMPC*-strategy for this thesis is contributed in §5.3. The proposed control is subsequently validated on real *AR.Drone 2.0* and *DJI M100* systems. This includes the discussion and evaluation of the *NMPC* parametrization. Furthermore, the experimental results are compared with simulation results. This comparison is used to assess the quality of the developed *HMDV* (3.40).

One challenge of using standard *MPC* for *UAV*s is the tracking of constantly moving targets. This leads to a constant offset, respectively tracking error. To address this issue, §5.4 is contributing a target position control *TPC*. Another challenge is to determine the optimal control action when obstacles are considered. For this purpose, §5.5 is giving an experimental introduction into (*CA*) with *MPC*. §5.6 is further generalizing this to task-based control by contributing a constraint design workflow. This workflow facilitates the incorporation of control objectives, sensor constraints and environmental constraints in the *NMPC* strategy.

5.1 Related work

A well established Cartesian *UAV* position control approach is a hover-*PID* controller which is based on a linearization around the *UAV*'s stationary state [Cor13]. The *UAV* position is thereby controlled in an outer loop, with an inner loop stabilizing its attitude. The global position and heading errors are computed and transformed into the vehicle frame according to

$$\begin{bmatrix} {}^{\mathcal{V}1}e_x \\ {}^{\mathcal{V}1}e_y \end{bmatrix} = \begin{bmatrix} \cos\left({}^{\mathcal{V}}\Psi\left(t\right)\right) & \sin\left({}^{\mathcal{V}}\Psi\left(t\right)\right) \\ -\sin\left({}^{\mathcal{V}}\Psi\left(t\right)\right) & \cos\left({}^{\mathcal{V}}\Psi\left(t\right)\right) \end{bmatrix} \begin{bmatrix} {}^{\mathcal{G}}x_{des} - {}^{\mathcal{G}}x \\ {}^{\mathcal{G}}y_{des} - {}^{\mathcal{G}}y \end{bmatrix} \tag{5.1}$$

$$e_{\mathcal{V}z} = {}^{\mathcal{G}}z_{des} - {}^{\mathcal{G}}z \tag{5.2}$$

$$e_{\mathcal{V}\Psi} = {}^{\mathcal{V}}\Psi_{des} - {}^{\mathcal{V}}\Psi. \tag{5.3}$$

After this transformation each channel $u \in \{u_{\mathcal{V}1_x}, u_{\mathcal{V}1_y}, u_{\mathcal{V}1_z}, u_{\omega_{\mathcal{V}1}}\}$ is controlled separately using the *PID* control law

$$u\left(t\right) = k_p e\left(t\right) + k_i \int_0^t e\left(t\right) dt + k_d \frac{\partial e\left(t\right)}{\partial t} \tag{5.4}$$

with proportional k_p, integral k_i and derivative gain k_d.

In the context of the OS4 project at the Swiss Federal Institute of Technology [BMS04], such a *PID* controller is derived. In addition, a Lyapunov stability proof for a full nonlinear quadrotor model is given. The authors conclude that the *PID* hover control shows a poor disturbance rejection. A more detailed study on the influence of aerial disturbance on hover-*PID* controlled *UAV*s is presented in [JHA+13]. In [Bea08], such a controller is applied for vision-based control in combination with a state estimator. More detailed information on visual quadrotor control strategies is given by [GDLP13]. An example for a vision based fuzzy position control is presented in [OmKV14]. [BNS04] is presenting a quadrotor *LQR*-controller and comparing it with the classical *PID* hover control. In contrast to the author's expectations, the *LQR* approach showed a less dynamic behavior and a steady-state error in the experiment. Another example for the use of *LQR* is given in [HD11], where an inverted pendulum is balanced on a quadrotor. To act in a bigger trust region than a conventional *PID* hover controller, [ST14] presents an adaptive control concept for *UAV*s. The concept is the adaptation of controller parameters based on a linearization of the *UAV* dynamics. With the objective of improving the control performance, a multiple surfaces sliding mode control is presented in [STSK13].

However, to fully exploit the nonlinear dynamics of a quadrotor, a nonlinear control approach is desired. For this purpose a backstepping control is developed in [HMLO02]. This includes a Lyapunov stability proof for the presented controller. In [BS05] a backstepping control is developed for the linear translational subsystem of a quadrotor. The attitude subsystem is controlled with a sliding mode controller. As a consequence, the closed-loop system shows a strong resistance towards disturbance. However, it introduces high frequencies into the system's controls which are causing drifting of the onboard sensors. More detailed information is given in [BS07a]. To address this issue, [BS07b] is presenting an integral backstepping controller as further development. The result is a control law for the full state model.

With the increasing computational power of computers, real-time *MPC* for fast systems has become feasible. An *LQR* control, as presented in [BNS04], is equal to the analytical solution for an unconstrained infinite horizon linear *MPC*. However, the big advantage of *MPC* is the consideration of constraints. This is particularly interesting for implementing safety measures. Linear *MPC* approaches for *UAV*s have been presented in [Bou12] and [ANT10]. A linear *MPC* example with state constraints for quadrotors is simulated in [LQS$^+$11]. In the experimental evaluation, the *MPC* is compared with a backstepping controller. The authors conclude, that the backstepping controller is performing better, while *MPC* allows safe operation by imposing dynamical constraints. However, a linear *MPC* does not cope with the nonlinearities of a *UAV*'s behavior. This results in a lower control performance. In [GM16], this is tackled with a multiple-shooting *SQP* pose *NMPC* for a *UAV* with inverse pendulum. However, the resulting *NMPC* computation time of $t_c = 0.35\,\mathrm{s}$ with 20 shooting nodes is high in relation to a *UAV*'s fast dynamics. [BMPL$^+$14] is therefore proposing to use predefined finite control trajectory sets within the *MPC* to reduce the computational burden. In the context of this thesis, the computational burden of *NMPC* is addressed by *CMSC/GMRES*, as discussed in §4.4.

A typical *MPC* controller is minimizing an objective function to track a desired trajectory under minimization of the energy effort. The result is a trade-off between state and control tracking. This trade-off typically leads to a constant tracking error for constantly moving targets. For safety purposes, such an "offset" is not desired. One strategy for an offset-free trajectory *MPC* for quadrotors is presented in [ROR08]. The author is separating the control problem of a quadrotor into a path tracking *MPC* with underlying $H\infty$ attitude control. Offset-free tracking is achieved by considering a disturbance error model in the path planning dynamics. The *MPC* control policy minimizes the offset by an integral part. The disadvantage of the proposed method is, that first the optimal controls are computed with *MPC* where the implemented constraints are respected. Afterwards, the controls are altered with the integral part, which then might lead to constraint violation. A more detailed analysis of the described reference tracking is given in [ROR10]. To tackle *UAV* tracking errors induced by e.g. wind gusts, [ANT10] is proposing a similar hierarchical *UAV* control scheme considering tracking errors as output errors in the *MPC* scheme. [LQS$^+$11] is presenting an offset-free linear *MPC* for quadrotors by using disturbance modeling. The advantage of modeling the disturbance to minimize the offset is, that information about the disturbance can be used to optimize the system's behavior. The drawback of this approach is the increased computational effort, as the disturbance has to be estimated and modeled by for example augmenting the system dynamics. A summary of offset-free linear *MPC* control strategies is provided in [Pan15], including disturbance model and observer.

Within this chapter, the tracking control for a *UAV* with limited vision sensor perception is discussed. One approach for vision sensors is visual servoing which is based on optical flow or features, as shown in [dRB08], [KNFL09] and [EIB12]. For traditional backstepping controllers, sensor perception limits can be addressed by switching the robot's formation control according to the compliance with the sensor constraints [WLY15]. Another approach is to compute the optimal boundary trajectories to satisfy the sensor constraint and track these, as shown for holonomic robots in [BH06] or for visual servoing in [BMCH07, LNGB$^+$10]. Nevertheless, the task dependency of the control laws makes it challenging to formulate control laws for complex scenarios with constraints. One way to avoid this loss of generality is to use *MPC* which allows defining tasks and constraints as optimization problem in a generic way. An example of a cooperative *MPC* using barrier function constraint handling with sensor and *CA* constraint is given in [DGS$^+$16] . With the use of switching parameters, *CA* constraints can also be formulated linear. This approach is for example presented in [FKM13], where a swarm of unicycles is controlled using mixed integer quadratic programming. In [SKK17a], an aerial manipulator is presented with a camera attached to the end effector. The camera is controlled using a stochastic *MPC* method for visual servoing in order to keep the target in the field of view. A direct implementation of sensor constraints in *MPC* for holonomic mobile manipulators is discussed in [AZR15]. In [OSB16], *MPC* is used to incorporate *CA* on a teleoperated *UAV*s.

5.2 Controllability

Considering the hover model (3.24) with ${}^{\mathcal{V}}\Psi$-angle description, the system velocities ${}^{\mathcal{V}1}\dot{x}, {}^{\mathcal{V}1}\dot{y}, {}^{\mathcal{V}1}\dot{z}, {}^{\mathcal{V}}\dot{\Psi}$ are directly influenced by its inputs $\boldsymbol{u}$. As all other states are velocity dependent, it is intuitive that all system states are controllable as it will be analyzed in the following. For system (3.24), the stationary condition

$$\mathbf{0} = \boldsymbol{f}\left(\boldsymbol{x}_s, \boldsymbol{u} = \mathbf{0}\right) \tag{5.5}$$

holds for the stationary points

$$\boldsymbol{x}_s = \left[{}^{\mathcal{G}}x_s, {}^{\mathcal{G}}y_s, {}^{\mathcal{G}}z_s, {}^{\mathcal{V}}\Psi_s, 0, 0, 0, 0\right]^{\mathrm{T}} \qquad \{{}^{\mathcal{G}}x_s, {}^{\mathcal{G}}y_s, {}^{\mathcal{G}}z_s, {}^{\mathcal{V}}\Psi_s\} \in \mathbb{R}. \tag{5.6}$$

To be able to steer the plant to arbitrary points around these static equilibrium points, the local accessibility is examined. The detailed algebraic analysis of the local reachability of input affine nonlinear systems with drift terms $\boldsymbol{f}_D(\boldsymbol{x})$ is given in [Sas99, Gra13] by following theorem:

Theorem 1. *[Sas99] A nonlinear system of the form*

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}\right) = \boldsymbol{f}_D\left(\boldsymbol{x}\right) + \sum_{i=1}^{n_u} \boldsymbol{f}_{i,I}\left(\boldsymbol{x}\right) u_i \;\; \forall \boldsymbol{x} \in \mathbb{R}^{n_u}, u_i \in \mathbb{R} \tag{5.7}$$

is locally accessible around state $\boldsymbol{x}_s$, if a set around $\boldsymbol{x}_s$ exists, so that

$$n_x = dim\left(\boldsymbol{\chi}\right) \tag{5.8}$$

holds with

$$\begin{aligned}\boldsymbol{\chi} = span\{&\boldsymbol{f}_{i,I}\left(\boldsymbol{x}\right), ..., \boldsymbol{f}_{n_u,I}\left(\boldsymbol{x}\right), \mathfrak{L}(\boldsymbol{f}_D\left(\boldsymbol{x}\right), ..., \boldsymbol{f}_{i,I}\left(\boldsymbol{x}\right)), \ldots \\ &\ldots \mathfrak{L}(\boldsymbol{f}_{i,I}\left(\boldsymbol{x}\right), ..., \boldsymbol{f}_{n_u,I}\left(\boldsymbol{x}\right)), ...\} \;\; i, j = 1, ..., n_u,\end{aligned} \tag{5.9}$$

where $\mathfrak{L}()$ is the Lie-Bracket [Sas99, Gra13]

$$\mathfrak{L}(\boldsymbol{f}_D, \boldsymbol{f}_I) := \frac{\partial \boldsymbol{f}_I\left(\boldsymbol{x}\right)}{\partial \boldsymbol{x}} \boldsymbol{f}_D\left(\boldsymbol{x}\right) - \frac{\partial \boldsymbol{f}_D\left(\boldsymbol{x}\right)}{\partial \boldsymbol{x}} \boldsymbol{f}_I\left(\boldsymbol{x}\right). \tag{5.10}$$

For the model (3.24) the system function $\boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}\right)$ is accordingly separated into its drift $\boldsymbol{f}_D\left(\boldsymbol{x}\right)$ and input-affine term $\boldsymbol{f}_I\left(\boldsymbol{x}\right)$

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{f}_D(\boldsymbol{x}) + \boldsymbol{f}_I(\boldsymbol{x})\,\boldsymbol{u} \tag{5.11}$$

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} {}^{\mathcal{V}1}\dot{x}(t)\cos\left({}^{\mathcal{V}}\Psi(t)\right) - {}^{\mathcal{V}1}\dot{y}(t)\sin\left({}^{\mathcal{V}}\Psi(t)\right) \\ {}^{\mathcal{V}1}\dot{x}(t)\sin\left({}^{\mathcal{V}}\Psi(t)\right) + {}^{\mathcal{V}1}\dot{y}(t)\cos\left({}^{\mathcal{V}}\Psi(t)\right) \\ {}^{\mathcal{G}}\dot{z}(t) \\ {}^{\mathcal{V}}\dot{\Psi}(t) \\ a_x \cdot {}^{\mathcal{V}1}\dot{x}(t) \\ a_y \cdot {}^{\mathcal{V}1}\dot{y}(t) \\ a_z \cdot {}^{\mathcal{V}1}\dot{z}(t) \\ a_\Psi \cdot {}^{\mathcal{V}}\dot{\Psi}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ b_x & 0 & 0 & 0 \\ 0 & b_y & 0 & 0 \\ 0 & 0 & b_z & 0 \\ 0 & 0 & 0 & b_\Psi \end{bmatrix} \begin{bmatrix} u_{\mathcal{V}1_x}(t) \\ u_{\mathcal{V}1_y}(t) \\ u_{\mathcal{V}1_z}(t) \\ u_{\omega_{\mathcal{V}1}}(t) \end{bmatrix}.$$

Accordingly $\boldsymbol{\chi}$ can be derived

$$\begin{aligned} \boldsymbol{\chi} &= \left[\boldsymbol{f}_I(\boldsymbol{x}), \mathfrak{L}(,\boldsymbol{f}_I(\boldsymbol{x})\,\boldsymbol{f}_D(\boldsymbol{x}))\right] \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 & -a_x\cos\left({}^{\mathcal{V}}\Psi(t)\right) & a_y\sin\left({}^{\mathcal{V}}\Psi(t)\right) & 0 & 0 \\ 0 & 0 & 0 & 0 & -a_x\sin\left({}^{\mathcal{V}}\Psi(t)\right) & -a_y\cos\left({}^{\mathcal{V}}\Psi(t)\right) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -a_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_\Psi \\ a_x & 0 & 0 & 0 & -a_x^2 & 0 & 0 & 0 \\ 0 & a_y & 0 & 0 & 0 & -a_y^2 & 0 & 0 \\ 0 & 0 & a_z & 0 & 0 & 0 & -a_z^2 & 0 \\ 0 & 0 & 0 & a_\Psi & 0 & 0 & 0 & -a_\Psi^2 \end{bmatrix} \end{aligned} \tag{5.12}$$

which has full rank for every $\boldsymbol{x}$. Therefore, according to theorem 1 there exists an input $\boldsymbol{u}$ to steer the plant with the dynamics (3.24) to any arbitrary state $\boldsymbol{x}$. Due to its similarity, the controllability for the *HMDV* (3.40) is omitted here.

5.3 NMPC pose tracking

To track a desired state $\boldsymbol{x_{des}}$ with *MPC*, the idea is to penalize the state error quadratically. In the same way the invested power can be limited by penalizing the controls $\boldsymbol{u}$ quadratically. A generic approach to track the orientation and angular velocity of a $3D$ rigid body is to minimize the attitude e_R and angular velocity error vector e_Ω. For an orientation in form of a rotation matrix $\boldsymbol{R} \in \mathcal{SO}3$ and angular velocity vector $\Omega = \begin{bmatrix} \varrho & \vartheta & \omega \end{bmatrix}^{\mathrm{T}}$, these can be written as [LLM12]

$$\boldsymbol{e}_R = 0.5(\boldsymbol{R}_{des}^{\mathrm{T}}\boldsymbol{R} - \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}_{des}) \tag{5.13}$$

$$\boldsymbol{e}_\Omega = \Omega - \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}_d\Omega_d. \tag{5.14}$$

The orientation tracking can be equally achieved by directly minimizing the scalar error [LLM12]

$$e_R = \mathrm{spur}(\boldsymbol{I}_3 - \boldsymbol{R}_{des}^{\mathrm{T}}\boldsymbol{R}) \tag{5.15}$$

which is computationally advantageous. While this tracking methodology is used for $3D$-orientation, the hover-models only consider their Ψ orientation around the ${}^{\mathcal{V}}z$ axis. This allows to track the orientation of the *HMDV* directly with a quadratic state error penalty (3.38). Under assumption of stable communication channels and constrained by the *HMDV* dynamics (3.40) and control limits, the *OCP* therefore yields

$$\min_{\boldsymbol{u}} \; J = \int_{t}^{t+T} \left(\underline{\boldsymbol{x_{des}}} - \underline{\boldsymbol{x}}\right)^{\mathrm{T}} \boldsymbol{Q} \left(\underline{\boldsymbol{x_{des}}} - \underline{\boldsymbol{x}}\right) + \underline{\boldsymbol{u}}^{\mathrm{T}}\boldsymbol{R}\underline{\boldsymbol{u}} \; d\tau \tag{5.16}$$

$s.\ t.$

$$\boldsymbol{f}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}\right) = \begin{bmatrix} {}^{\mathcal{V}1}\underline{\dot{x}}\left(\tau\right)\underline{o_x}\left(\tau\right) - {}^{\mathcal{V}1}\underline{\dot{y}}\left(\tau\right)\underline{o_y}\left(\tau\right) \\ {}^{\mathcal{V}1}\underline{\dot{x}}\left(\tau\right)\underline{o_y}\left(\tau\right) + {}^{\mathcal{V}1}\underline{\dot{y}}\left(\tau\right)\underline{o_x}\left(\tau\right) \\ {}^{\mathcal{G}}\underline{\dot{z}}\left(\tau\right) \\ -\underline{o_y}\left(\tau\right) \cdot {}^{\mathcal{V}}\underline{\Psi}\left(\tau\right) \\ \underline{o_x}\left(\tau\right) \cdot {}^{\mathcal{V}}\underline{\Psi}\left(\tau\right) \\ a_x \cdot {}^{\mathcal{V}1}\underline{\dot{x}}\left(\tau\right) + b_x \cdot \underline{u}_{\mathcal{V}1_x}\left(\tau\right) \\ a_y \cdot {}^{\mathcal{V}1}\underline{\dot{y}}\left(\tau\right) + b_y \cdot \underline{u}_{\mathcal{V}1_y}\left(\tau\right) \\ a_z \cdot {}^{\mathcal{V}1}\underline{\dot{z}}\left(\tau\right) + b_z \cdot \underline{u}_{\mathcal{V}1_z}\left(\tau\right) \\ a_\Psi \cdot {}^{\mathcal{V}}\underline{\dot{\Psi}}\left(\tau\right) + b_\Psi \cdot \underline{u}_{\mathcal{V}1_\omega}\left(\tau\right) \end{bmatrix} \tag{5.17}$$

with input limits: $|u| < 1$:

$$\boldsymbol{0} \leq \boldsymbol{c_{in}} = \left[\underline{u}_{\mathcal{V}1_x}{}^2 - 1, \underline{u}_{\mathcal{V}1_y}{}^2 - 1, \underline{u}_{\mathcal{V}1_z}{}^2 - 1, \underline{u}_{\mathcal{V}1_\omega}{}^2 - 1\right]^{\mathrm{T}} \tag{5.18}$$

$$\underline{\boldsymbol{x}}\left(t\right) = \boldsymbol{x}\left(t\right) \tag{5.19}$$

This *OCP* is solved with *CMSC/GMRES* using the parametrization (see §4.3.4):

$$\xi = 1,\ \upsilon \ = \ 1,\ \Delta t \ = \ 0.01\,\mathrm{s}, T \ = \ 1\,\mathrm{s}, \tag{5.20}$$

$$N = 10,\ h \ = \ 0.001\,\mathrm{s},\ h \ = \ 10^{-8},\ i_{max} = 10 \tag{5.21}$$

The corresponding penalty matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ in (5.16) are chosen according to the system dynamics and desired closed-loop response. As the parametrization is an enhancement problem and related to the system dynamics, the penalty values are determined empirically for each individual type of drone. To measure the convergence towards a desired pose, the position tracking error

$$e_p = \left\| \begin{matrix} e_{\mathcal{G}_x} \\ e_{\mathcal{G}_y} \\ e_{\mathcal{G}_z} \end{matrix} \right\|_2 = \left\| \begin{matrix} {}^{\mathcal{G}}x_{des} - {}^{\mathcal{G}}x \\ {}^{\mathcal{G}}y_{des} - {}^{\mathcal{G}}y \\ {}^{\mathcal{G}}z_{des} - {}^{\mathcal{G}}z \end{matrix} \right\|_2 \tag{5.22}$$

and orientation tracking error

$$e_{\Psi} = \left\| {}^{\mathcal{V}}\Psi_{des} - {}^{\mathcal{V}}\Psi \right\|_2 \tag{5.23}$$

can be used. In the following, this is shown for real *AR.Drone 2.0* and *DJI M100* quadrotors.

5.3.1 AR.Drone 2.0 NMPC pose tracking parametrization

To control the pose of the *AR.Drone 2.0*, the *OCP* (5.16)-(5.19) is solved using *CMSC/GMRES* with the parametrization (5.20-5.21) and the *AR.Drone 2.0 HMDV* (3.49). The control design parameters are the constraint matrices $\boldsymbol{Q}$, $\boldsymbol{R}$ which balance between trajectory tracking and energy optimality. In order to demonstrate this effect on the closed-loop behavior, $\boldsymbol{Q}$ is chosen to

$$\boldsymbol{Q} = \mathrm{diag}\left(\left[1, 1, 1, 1, 1, 0, 0, 0, 0\right]\right) \tag{5.24}$$

to track the *UAV* pose, but no velocities. Then, the real *AR.Drone 2.0* response is determined for different control penalties $\boldsymbol{R}$

$$\boldsymbol{R} = \operatorname{diag}\left(\begin{bmatrix}1.0, 1.0, 1.0, 1.0\end{bmatrix}\right) \tag{5.25}$$

$$\boldsymbol{R}_H = \operatorname{diag}\left(\begin{bmatrix}2.0, 2.0, 2.0, 2.0\end{bmatrix}\right) \tag{5.26}$$

$$\boldsymbol{R}_L = \operatorname{diag}\left(\begin{bmatrix}0.5, 0.5, 0.5, 0.5\end{bmatrix}\right). \tag{5.27}$$

$\boldsymbol{R}$ is hereby indicating the nominal control penalty, $\boldsymbol{R}_H$ a higher control penalty and $\boldsymbol{R}_L$ lower control penalty. The nominal parametrization for this work has been chosen to provide a fast closed-loop response in ${}^{\mathcal{G}}x$,${}^{\mathcal{G}}y$,${}^{\mathcal{V}}\Psi$ and due to the low performance of the internal altitude controller, a highly damped response in ${}^{\mathcal{G}}z$ direction.

The experimental setup consists of a real *AR.Drone 2.0* quadrotor which is following a sequence of desired pose changes. The *UAV* is localized by means of a motion capture system. The complete scenario is consisting of 4 distinguishable test cases (*TC*). In each scenario, the *UAV* orientation is indicated by an arrow pointing in ${}^{\mathcal{V}1}x$-direction.

The first *TC* is shown in Figure 5.1a and represents a lateral position change in the ${}^{\mathcal{G}}x$, ${}^{\mathcal{G}}y$ plane. The orientation ${}^{\mathcal{V}}\Psi_{des} = 0\,\text{rad}$ and altitude ${}^{\mathcal{G}}z_{des} = 1.5\,\text{m}$ are fixed and the ${}^{\mathcal{G}}x_{des}$, ${}^{\mathcal{G}}y_{des}$ displacement is executed stepwise in the form of a square. This *TC* allows to evaluate the damping of the closed-loop system for the forward and sideward channels.

Figure 5.1b is showing the second *TC* of a combined ${}^{\mathcal{G}}x_{des}$, ${}^{\mathcal{G}}y_{des}$-position and ${}^{\mathcal{V}}\Psi_{des}$ orientation change. As previously, the position changes form a square in the ${}^{\mathcal{G}}x_{des}$, ${}^{\mathcal{G}}y_{des}$-plane, while the front of the drone is oriented towards the center of the square. This scenario is chosen to evaluate the trade-off between position and orientation tracking.

For the third *TC*, the *UAV* position is fixed in $\vec{\boldsymbol{p}}_{des} = [0, 0, 1.5]^{\mathrm{T}}\text{m}$, while only the orientation is varied in steps of $\Delta{}^{\mathcal{V}}\Psi_{des} = \pm\pi\,\text{rad}$. This *TC* is suitable to evaluate the orientation tracking. The resulting *UAV* trajectory is given in Figure 5.1c.

The final fourth *TC* is evaluating the closed-loop altitude tracking, by displacing the *UAV*'s target position between $\vec{\boldsymbol{p}}_{des} = [0, 0, 1.5]^{\mathrm{T}}\text{m}$ and $\vec{\boldsymbol{p}}_{des} = [0, 0, 2.0]^{\mathrm{T}}\text{m}$. For this *TC*, the orientation is fixed to ${}^{\mathcal{V}}\Psi_{des} = 0\,\text{rad}$. The altitude displacement is limited to $\Delta{}^{\mathcal{G}}z_{des} = 0.5\,\text{m}$ to avoid strong air current ground effects which have not been considered in the identified model. Furthermore, the perception space of the motion capture system is limited upwards. Figure 5.1d is showing the resulting *UAV* altitude changes.

To analyze the *AR.Drone 2.0*'s closed-loop behavior in detail, Figure 5.2a is showing the step response in ${}^{\mathcal{G}}x$-direction during *TC*1. As expected the high control

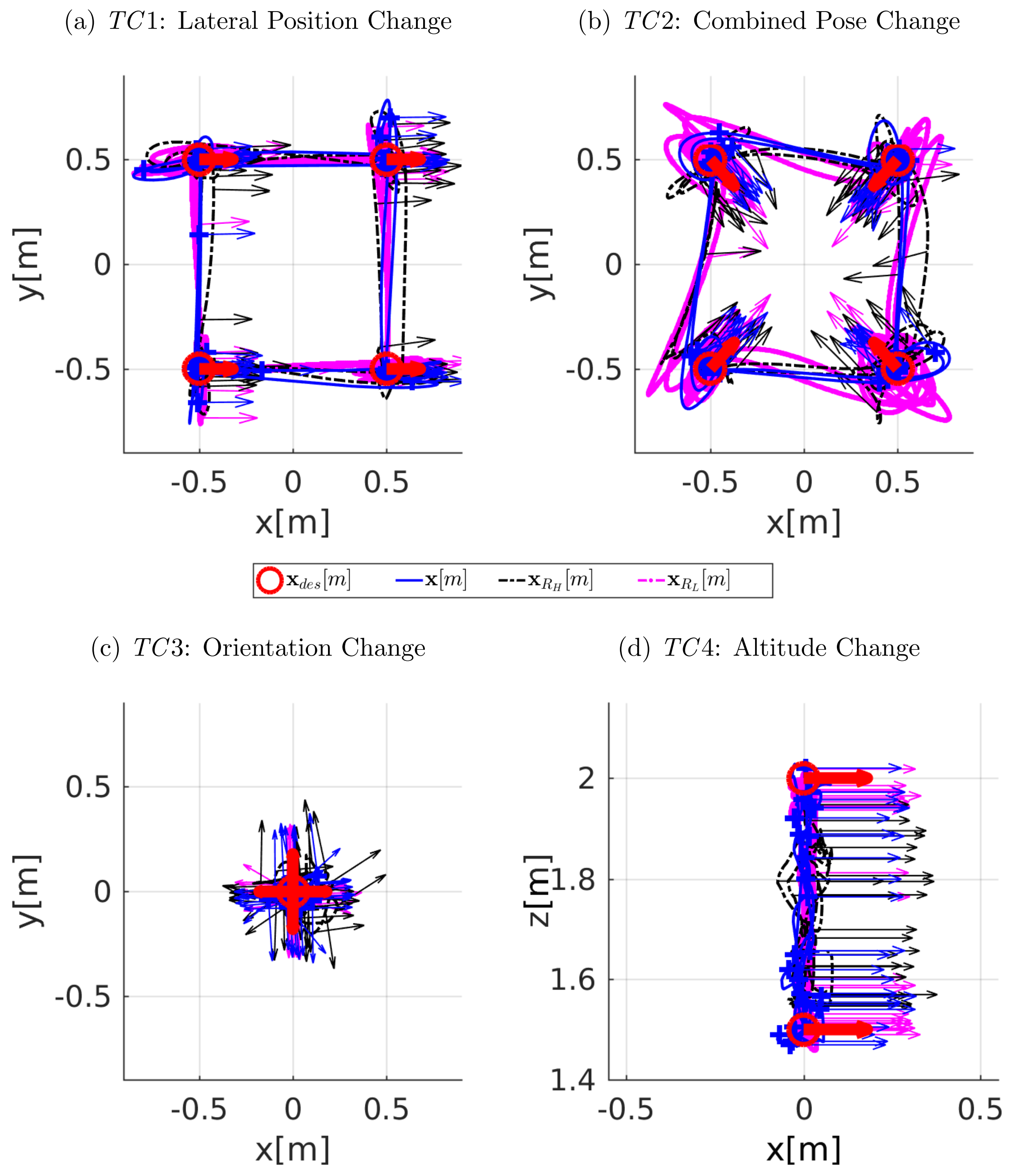


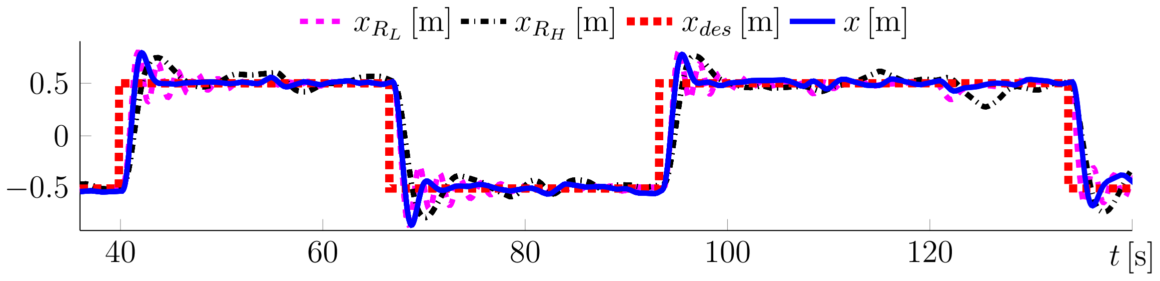
Figure 5.1: *AR.Drone 2.0 NMPC* pose tracking parametrization: 3D trajectories

penalty $\boldsymbol{R}_H$ leads to a slower response, while a lower control penalty $\boldsymbol{R}_L$ results in a faster but less damped response. The closed-loop trajectory is showing similarities to a second order plant (*PT2*). Interestingly, the response for the chosen nominal control penalty $\boldsymbol{R}$ is oscillating the least. All three solutions show an overshoot of $\Delta^{\mathcal{G}}x \approx 0.3\,\mathrm{m}$ for a step height of $\Delta^{\mathcal{G}}x \approx 1.0\,\mathrm{m}$, which can be further reduced by making the closed-loop system slow via higher control penalties. However, this has not been further investigated, as the objective in most *UAV* scenarios is a fast response in ${}^{\mathcal{G}}x, {}^{\mathcal{G}}y$-direction. The same behavior can be also observed for the ${}^{\mathcal{G}}y$-axis in Figure 5.2b. Figure 5.2c is showing the *UAV*'s ${}^{\mathcal{G}}z$-axis response in $TC4$. The altitude response is significantly slower than in ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-direction and shows a highly damped asymptotic convergence towards the target value. However, this converging trajectory is superposed with disturbance introduced by the internal altitude controller of the *AR.Drone 2.0*. The high variance of the *AR.Drone 2.0*'s altitude sensing solution can lead to sudden undesired displacements, as e.g. at time instance $t \approx 442\,\mathrm{s}$ in $\boldsymbol{R}_L$. Finally, the ${}^{\mathcal{V}}\Psi$ behavior is given in 5.2d. Due to the direction vector description in the *MPC* prediction model, the controller is able to stabilize the *AR.Drone 2.0* in full 360°. In contrast to a simple single angle error limitation to $-\pi/2 < e_\Psi \leq \pi/2$, the direction vector approach does ensure the uniqueness of the solution throughout the prediction horizon.

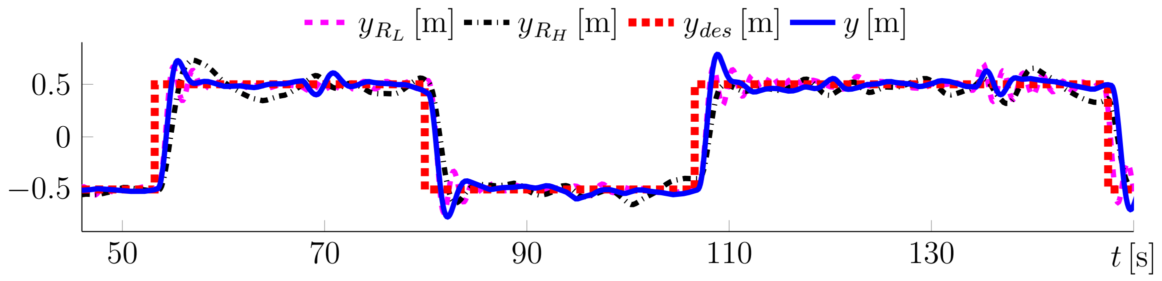
All considered channels ${}^{\mathcal{G}}x, {}^{\mathcal{G}}y, {}^{\mathcal{G}}z, {}^{\mathcal{V}}\Psi$ show the desired asymptotic stability which validates the chosen *NMPC* pose tracking approach.

The direct influence of $\boldsymbol{Q}$ on the controls $\boldsymbol{u}$ can be seen for q in Figure 5.3a, q_H in Figure 5.3b and q_L in Figure 5.3c. As desired, a higher penalty leads to a lower control amplitude. An extract of the resulting position tracking error e_p (5.22) is given in Figure 5.3d and Figure 5.3e shows an extract of the orientation tracking error e_Ψ (5.23). Both errors are converging towards zero after each step in the desired trajectory, while the q_H response is as expected the slowest. This states the effectiveness and performance of the applied *NMPC* pose tracking approach.

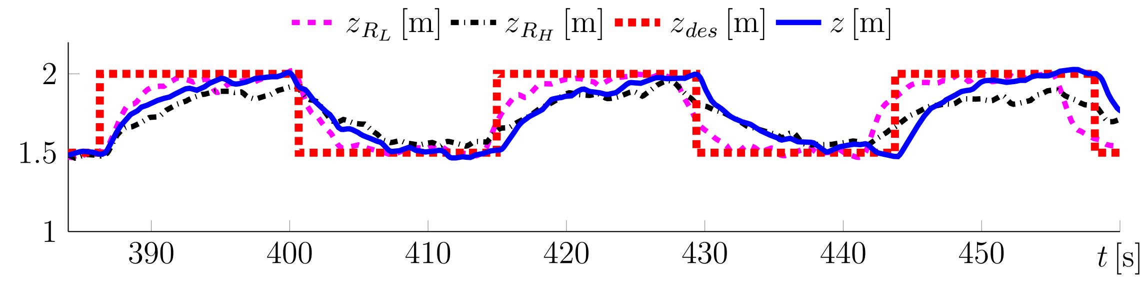
(a) *AR.Drone 2.0* ${}^{\mathcal{G}}x$-Position response in $TC1$



(b) *AR.Drone 2.0* ${}^{\mathcal{G}}y$-Position response in $TC1$



(c) *AR.Drone 2.0* ${}^{\mathcal{G}}z$-Position response in $TC4$



(d) *AR.Drone 2.0* ${}^{\mathcal{V}}\Psi_{\mathcal{G}}$-Orientation response in $TC3$

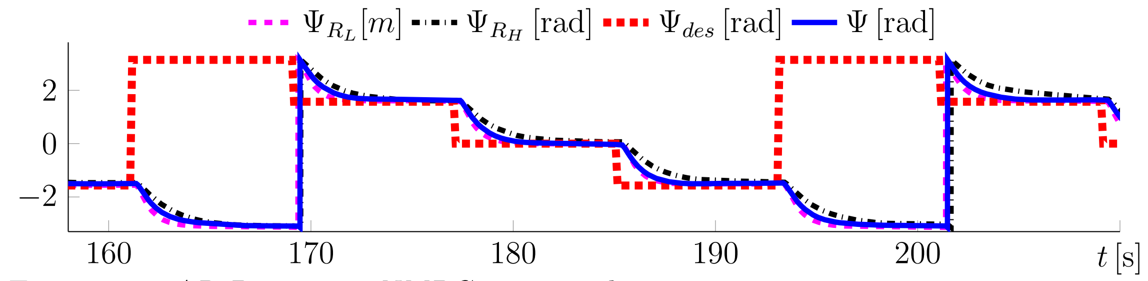


Figure 5.2: *AR.Drone 2.0 NMPC* pose tracking parametrization: state trajectories

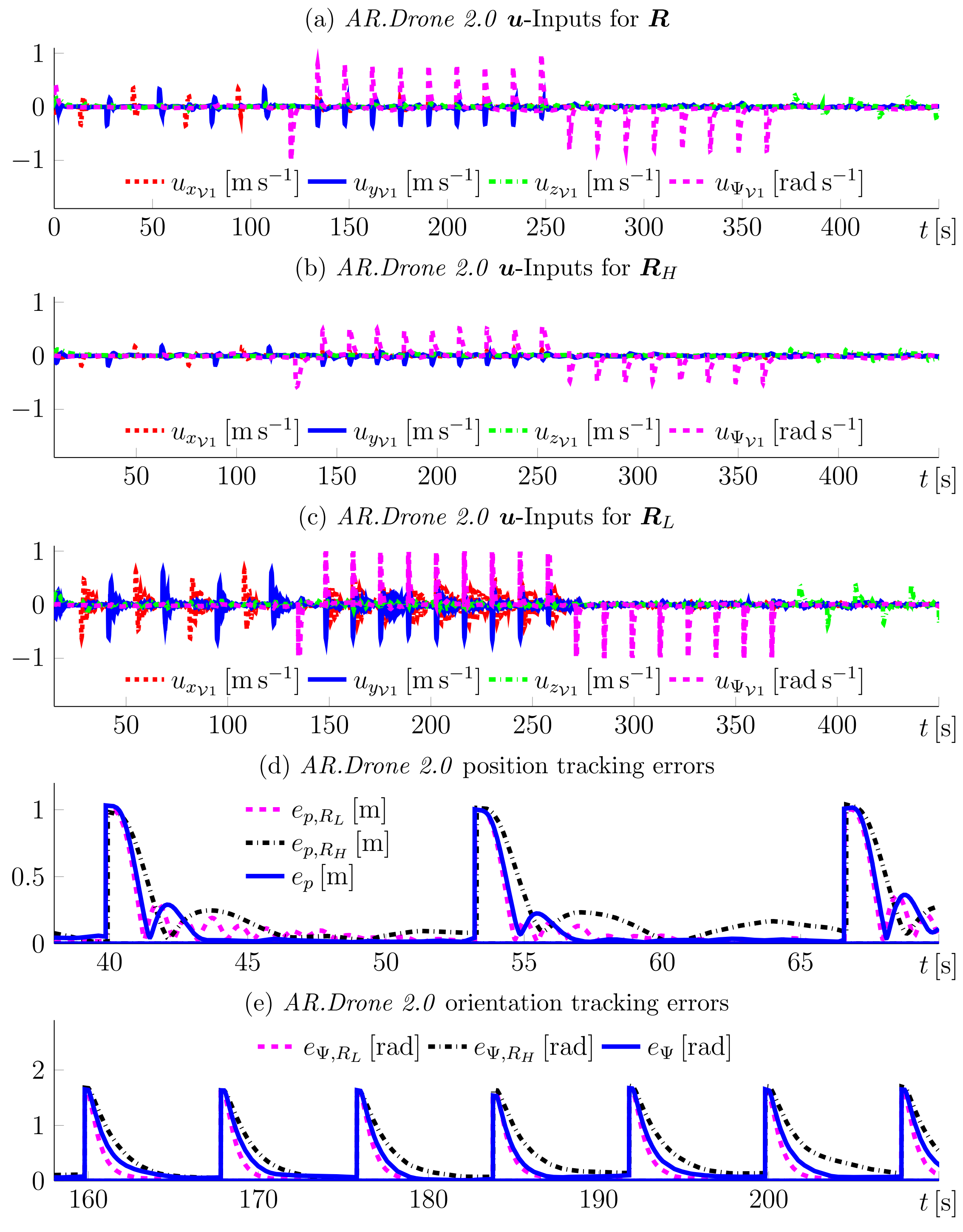


Figure 5.3: *AR.Drone 2.0 NMPC* pose tracking parametrization: tracking error, controls and computation time

AR.Drone 2.0 NMPC pose tracking

In order to provide a simulative control development framework, the developed discretized *UAV* models (3.45) are implemented in *V-REP*. As the utilized *CMSC/GMRES* solver is using an explicit Euler-integrator scheme in the context of this work, these simulation models are equivalent to the prediction model (3.49). This allows to use the closed-loop behavior in order to evaluate the model performance and the control performance. The idea is, that the more the real *AR.Drone 2.0* and simulator closed-loop trajectory are congruent, the higher the quality of the utilized prediction model is. Furthermore, as the simulator and the *NMPC* model are equivalent, the control performance can be evaluated in an environment where just delay and numerical errors appear.

For the evaluation, the previous four TCs of §5.3.1 are used. Figure 5.4a shows, that the simulated *AR.Drone 2.0* response in $TC1$ is following the square pattern between the desired poses as desired. The disturbance of the real *AR.Drone 2.0* system is particularly visible in the translation phases. Furthermore, the overshoot in the corner regions visualizes the model mismatch. This difference is further exacerbated by a combined position and orientation change in $TC2$, as visualized in Figure 5.4b. Here, the trajectory pattern between the real *AR.Drone 2.0* and the simulator is clearly distinguishable, which appears to indicate a significant model mismatch. However, it is important to consider that the pose changes are imposed stepwise. Therefore, the trajectory in the translation periods is directly determined by the solver, wherefore even small disturbance, respectively model-mismatch is able to influence the trajectory pattern. Due to their congruency, the simulator ${}^{\mathcal{V}}\Psi$ response in $TC3$ (Figure 5.4c) and ${}^{\mathcal{G}}z$ in $TC4$ (Figure 5.4d) is covered by the real systems trajectory. For a more detailed analysis of the closed-loop behavior, the control channels ${}^{\mathcal{G}}x$, ${}^{\mathcal{G}}y$, ${}^{\mathcal{G}}z$, ${}^{\mathcal{V}}\Psi$ are plotted separately.

The ${}^{\mathcal{G}}x$ trajectory in Figure 5.5a as well as the ${}^{\mathcal{G}}y$ trajectory in Figure 5.5b show that the controlled real *AR.Drone 2.0* is responding faster than the simulator model with a higher initial overshoot. Figure 5.5c shows, that the real *AR.Drone 2.0* is responding slower in ${}^{\mathcal{G}}z$-direction. The ${}^{\mathcal{V}}\Psi$-response is finally shown in Figure 5.5d, where simulated and real trajectory are congruent. All trajectories in Figure 5.5a-5.5d show the desired performance of a closed-loop system and state the quality of the model by the low trajectory difference.

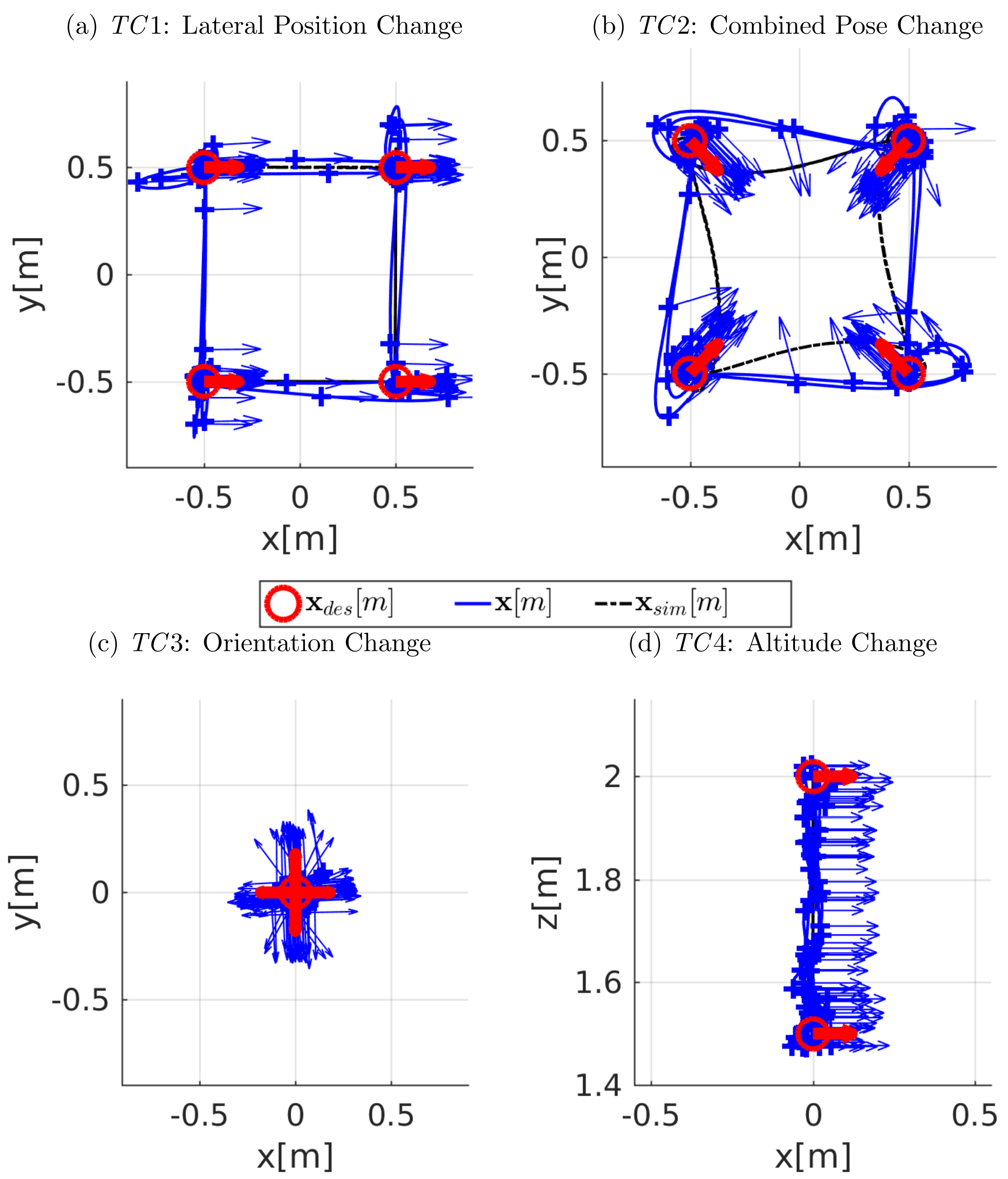
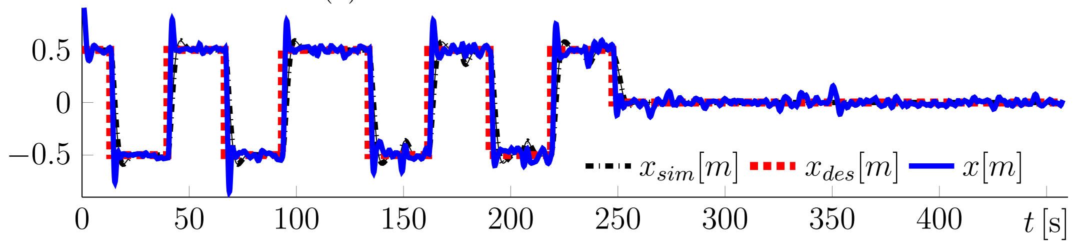
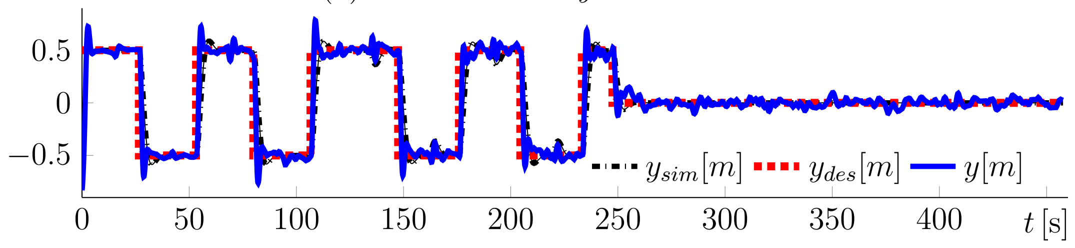
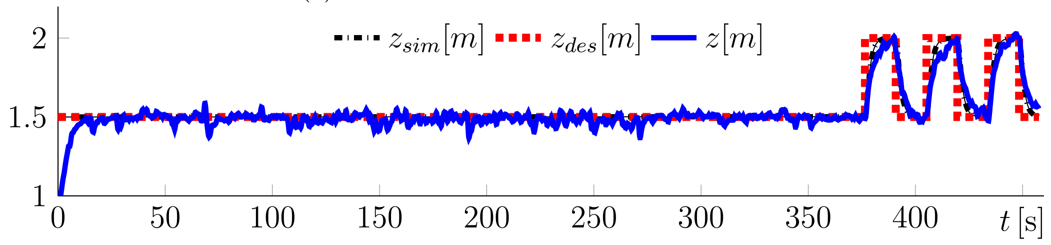


Figure 5.4: *AR.Drone 2.0 NMPC* pose tracking: 3D trajectories


(a) *AR.Drone 2.0* ${}^{\mathcal{G}}x$-Position


(b) *AR.Drone 2.0* ${}^{\mathcal{G}}y$-Position


(c) *AR.Drone 2.0* ${}^{\mathcal{G}}z$-Position

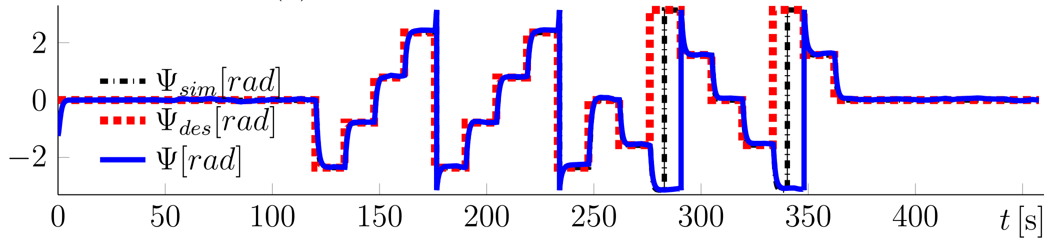

(d) *AR.Drone 2.0* ${}^{\mathcal{V}}\Psi$-Orientation

Figure 5.5: *AR.Drone 2.0 NMPC* pose tracking: channel trajectories

This is also stated by similar position e_p (5.22) and orientation error e_Ψ (5.23), as shown in Figure 5.6a. The resulting real drone controls are given in Figure 5.6b. Finally, the computational efficiency of the proposed *NMPC* is validated by an average computation time of $\bar{t_c} = 0.37\,\text{ms}$ for the real *AR.Drone 2.0* system. The related computation time plot is shown in 5.6c.

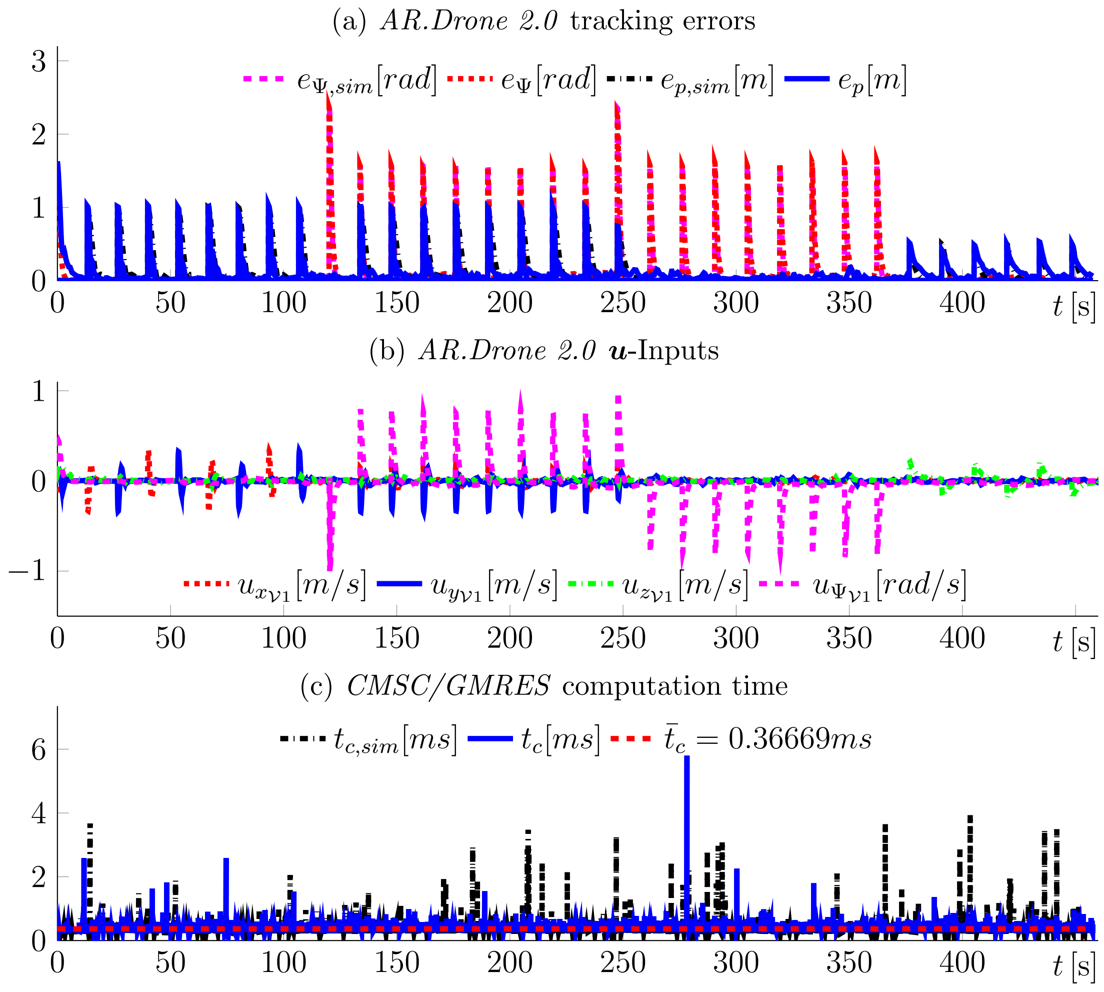


Figure 5.6: *AR.Drone 2.0 NMPC* pose tracking: tracking error, controls and computation time

5.3.2 DJI M100 NMPC pose tracking

Within the context of this work, the *NMPC* is also used to track the pose of a *DJI M100* quadrotor, as for the *AR.Drone 2.0* in §5.3.1. For this purpose, the identified *DJI M100* model (3.55) is used as prediction model within *OCP* (5.16)-(5.19). The corresponding *CMSC/GMRES* parametrization is given in (5.20)-(5.21). The state $\boldsymbol{Q}$ and control penalties $\boldsymbol{R}$ are chosen according to the closed-loop behavior. Detailed results are omitted here, as the procedure and penalizing effects are similar to the *AR.Drone 2.0* in §5.3.1. As a result, the penalties are chosen empirically to

$$\boldsymbol{Q} = \operatorname{diag}\left(\left[1, 1, 1, 1, 1, 0, 0, 0, 0\right]\right) \tag{5.28}$$

$$\boldsymbol{R} = \operatorname{diag}\left(\left[10.0, 10.0, 10.0, 1.0\right]\right). \tag{5.29}$$

As for the *AR.Drone 2.0*, this section is evaluating the quality of the identified *DJI M100 HMDV* by using its discretized form (3.45) as simulation model in *V-REP*. Figure 5.7a-5.7d show similar patterns in all test cases for the real *DJI M100* and the simulator response.

This similarity is also observed when considering the closed-loop behavior of the individual states. The low trajectory difference in Figure 5.8a-5.8d is stating the high quality of the model. In detail, Figure 5.8a shows the ${}^{\mathcal{G}}x$-direction in which *UAV* and simulator show a fast approximation towards the target trajectory. The simulator shows a slightly slower reaction. This is also observed for the ${}^{\mathcal{G}}x$-axis in Figure 5.8b, for ${}^{\mathcal{G}}z$ in Figure 5.8c and ${}^{\mathcal{V}}\Psi$ in Figure 5.8d. As a remark, the angle ${}^{\mathcal{V}}\Psi$ is illustrated here in its typical interval definition $]-\pi, \pi]$, wherefore oscillating steps between ${}^{\mathcal{V}}\Psi = \pm\pi$ can be observed at e.g. $t \approx 170\,\mathrm{s}$, $t \approx 190\,\mathrm{s}$.

The computed controls for the real *DJI M100* are given in Figure 5.9a. These are the normalized inputs $u \in [-1, 1]$ which are translated to the real *DJI M100* according to (3.54). As desired, e_p (5.22) and e_Ψ (5.23) are converging towards zero after each reference change. Therefore, Figure 5.9b is stating the asymptotic stability of the *NMPC* controlled systems. In this context, the simulator and real *DJI M100* show congruent tracking errors which is also stating the quality of the identified model. Finally, the computation time is given in Figure 5.9c. The average computation time of $\bar{t_c} = 0.4295\,\mathrm{ms}$ is validating the real-time applicability. of the presented *NMPC* approach considering a control update interval time of $\Delta t = 10\,\mathrm{ms}$.

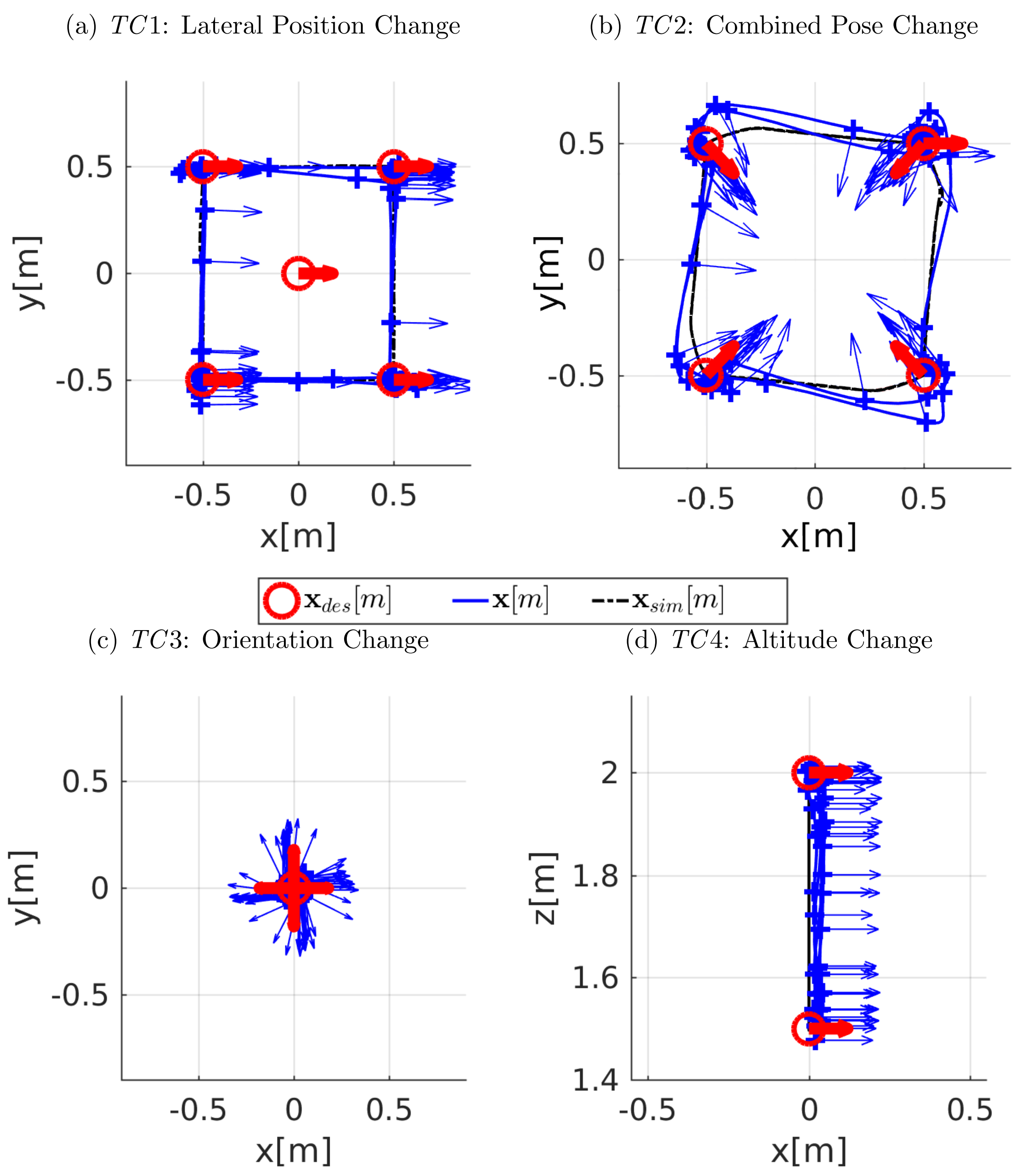


Figure 5.7: *DJI M100 NMPC* pose tracking: 3D trajectory

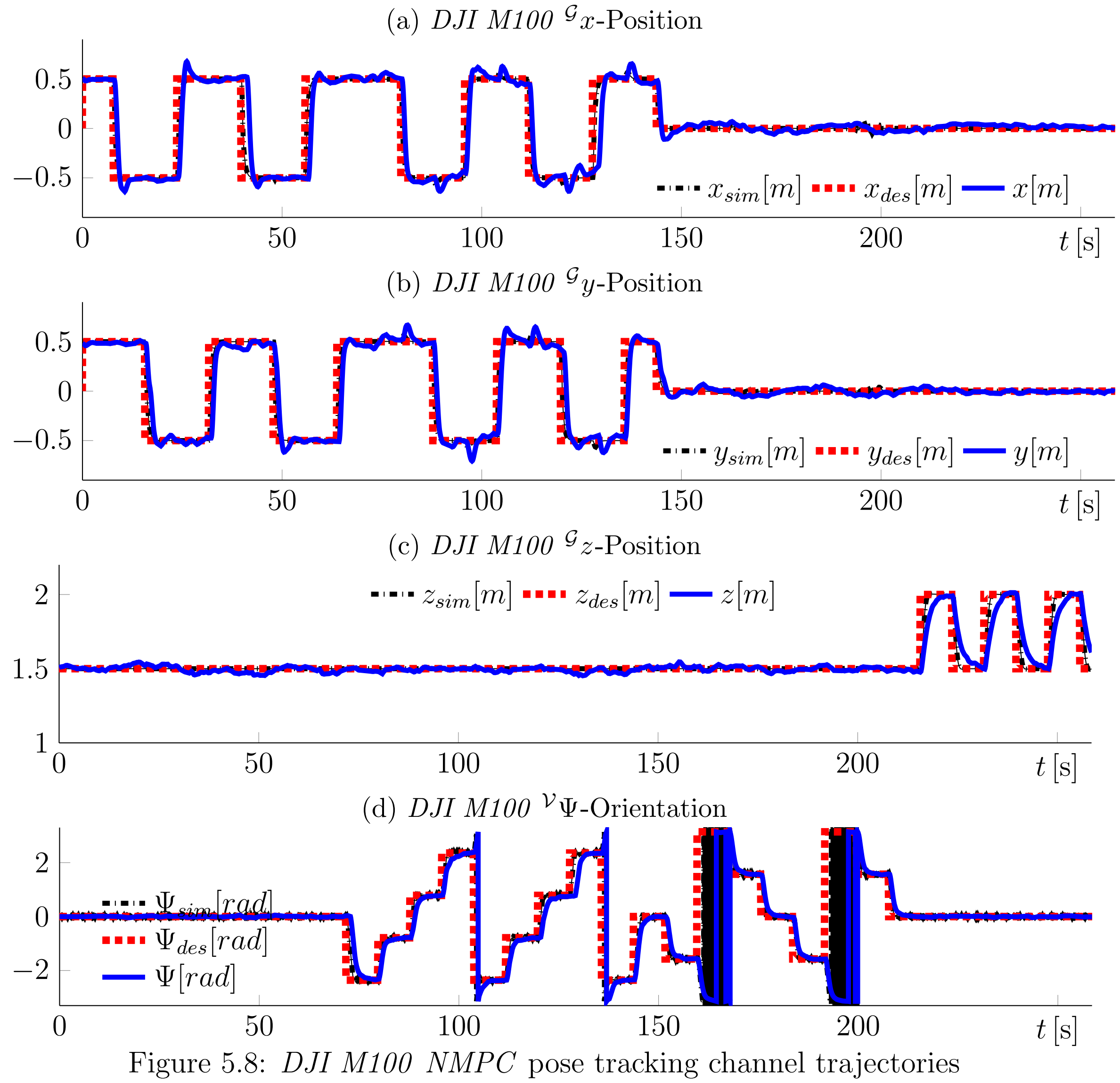


Figure 5.8: *DJI M100 NMPC* pose tracking channel trajectories

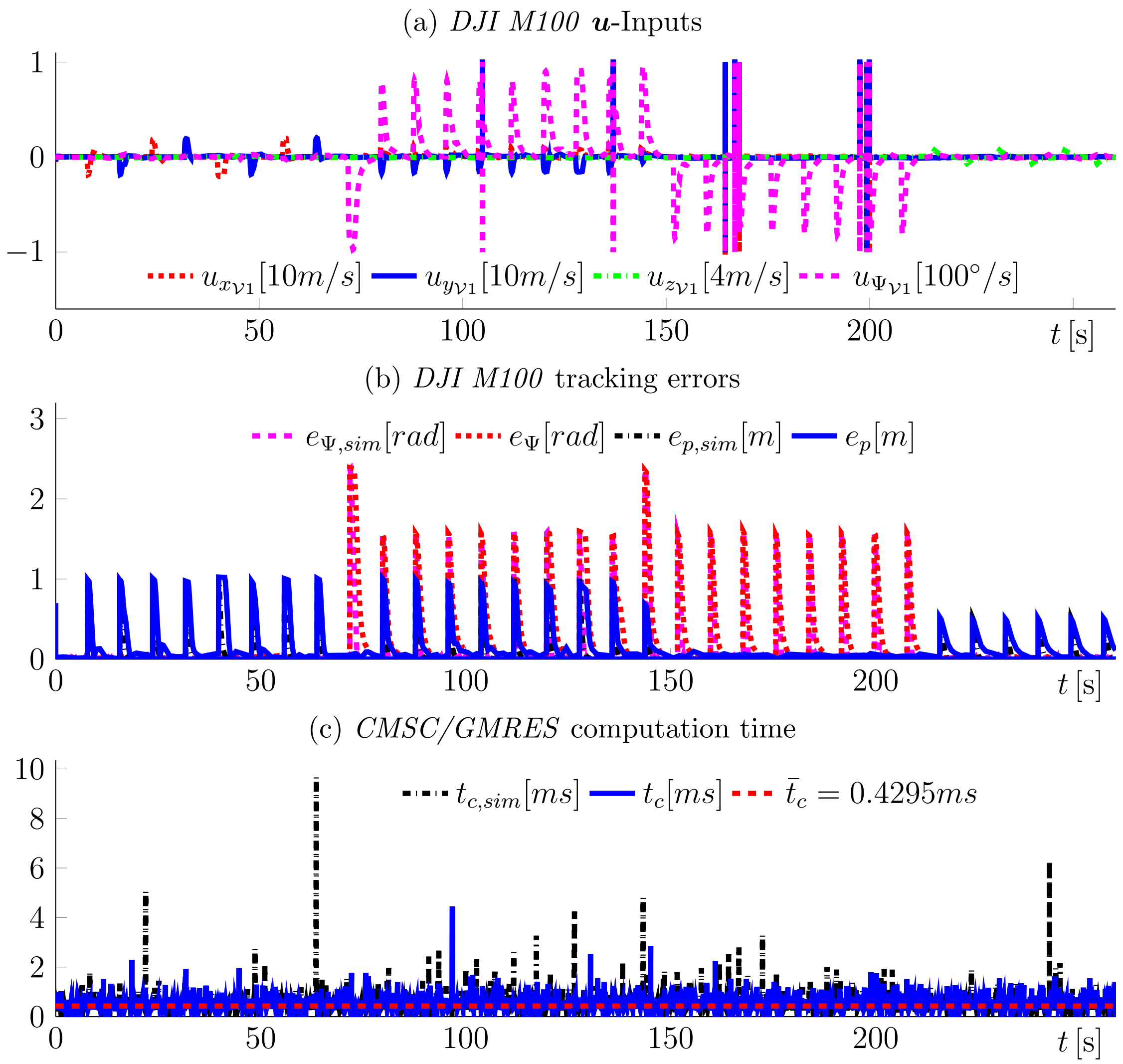


Figure 5.9: *DJI M100 NMPC* pose tracking channel trajectories

5.4 Offset-free MPC

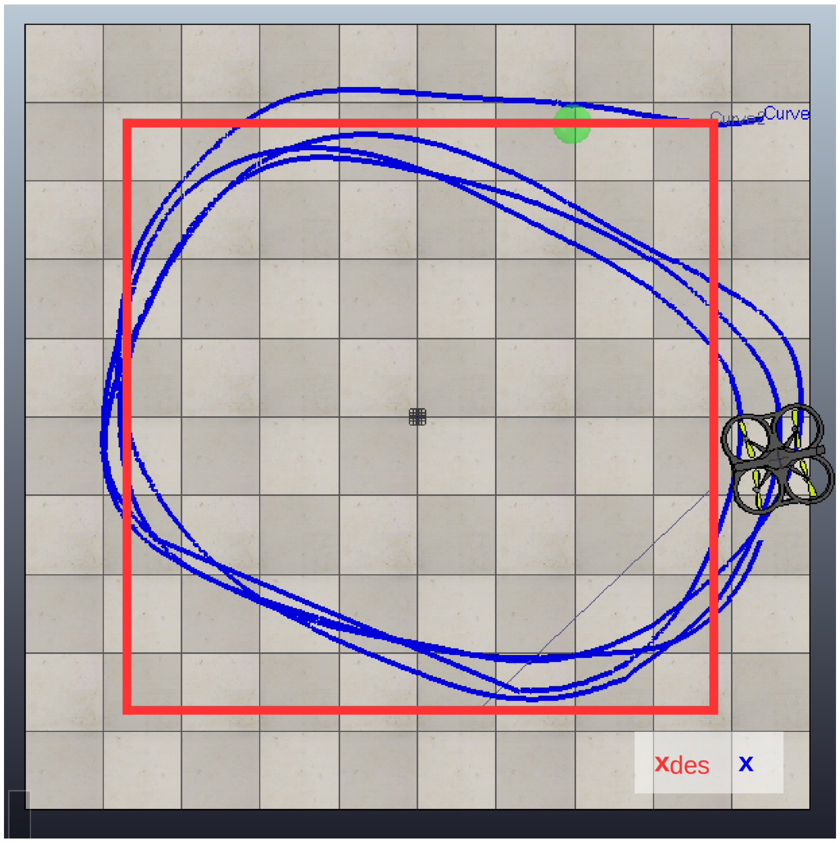


Figure 5.10: Simulation of *NMPC* tracking of constantly moving target

In many applications as for example area exploration, *UAV*s follow a given path with a defined velocity. However, only static pose changes have been addressed in §5.3. Considering constantly changing target states, the phenomena shown in Figure 5.10 can be observed. In this scenario the target position (green sphere) is moving with a constant velocity on the square in the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane (red trajectory). The *UAV* is tracking the constantly moving target position which results in the ellipsoidal pattern (blue trajectory). The closed-loop *UAV* system acts as a low-pass filter, as the corners of the square are not reached. This systematic protraction of the closed-loop system is caused by:

1 Delay between state measurements and control output: In the time that the optimal controls are computed, the target has already moved on

2 Trade-off in control tracking versus state tracking The given *UAV*'s target velocity and target position trajectory do not fit together

Apart from the pattern shown in Figure 5.10, the protraction leads to a static tracking error

$$\lim_{t\to\infty} e_p \to \text{const.}, \tag{5.30}$$

for a target which is moving with constant velocity. This can pose a safety problem. In the context of the publication [DKMV16c], this effect has been analyzed using a deprecated version of the *RHMY* (5.32). The test *OCP* is given as

$$\min_{\underline{\boldsymbol{u}}} \; J = \int_{t_0}^{t_f} \left(\underline{\boldsymbol{x}}_{\boldsymbol{des}} - \underline{\boldsymbol{x}}\right)^\top \boldsymbol{Q} \left(\underline{\boldsymbol{x}}_{\boldsymbol{des}} - \underline{\boldsymbol{x}}\right) + \underline{\boldsymbol{u}}^\top \boldsymbol{R} \underline{\boldsymbol{u}} \;\, \mathrm{d}\tau \tag{5.31}$$

s. t.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} {}^{\mathcal{V}1}\dot{\underline{x}}(t)\cos\left({}^{\mathcal{V}}\underline{\Psi}(t)\right) - {}^{\mathcal{V}1}\dot{\underline{y}}(t)\sin\left({}^{\mathcal{V}}\underline{\Psi}(t)\right) \\ {}^{\mathcal{V}1}\dot{\underline{x}}(t)\sin\left({}^{\mathcal{V}}\underline{\Psi}(t)\right) + {}^{\mathcal{V}1}\dot{\underline{y}}(t)\cos\left({}^{\mathcal{V}}\underline{\Psi}(t)\right) \\ 0.8827\underline{u}_{\mathcal{V}1_z}(t) \\ 1.265\underline{u}_{\omega\mathcal{V}1}(t) \\ -0.8799 \cdot {}^{\mathcal{V}1}\dot{\underline{x}}(t) + 3.273 \cdot \underline{u}_{\mathcal{V}1_x}(t) \\ -0.5092 \cdot {}^{\mathcal{V}1}\dot{\underline{y}}(t) + 1.458 \cdot \underline{u}_{\mathcal{V}1_y}(t) \end{bmatrix} \tag{5.32}$$

with input limits: $|u| < 1$:

$$\mathbf{0} \le \boldsymbol{c_{in}} = \left[\underline{u}_{\mathcal{V}1_x}{}^2 - 1, \underline{u}_{\mathcal{V}1_y}{}^2 - 1, \underline{u}_{\mathcal{V}1_z}{}^2 - 1, \underline{u}_{\mathcal{V}1_\omega}{}^2 - 1\right]^{\mathrm{T}} \tag{5.33}$$

$$\underline{\boldsymbol{x}}(0) = \left[0, 0, 0, 0, 0, 0\right] \tag{5.34}$$

$$\boldsymbol{Q} = \operatorname{diag}\left(\left[1, 1, 8, 3, 1.5, 1.5\right]\right), \quad \boldsymbol{R} = \operatorname{diag}\left(\left[1.5, 1.5, 3.0, 3.1\right]\right). \tag{5.35}$$

using the *CMSC/GMRES* parameters

$$\xi = 10,\; \upsilon \;=\; 1,\; \Delta t \;=\; 0.1\,\mathrm{s}, T \;=\; 1\,\mathrm{s}, \tag{5.36}$$

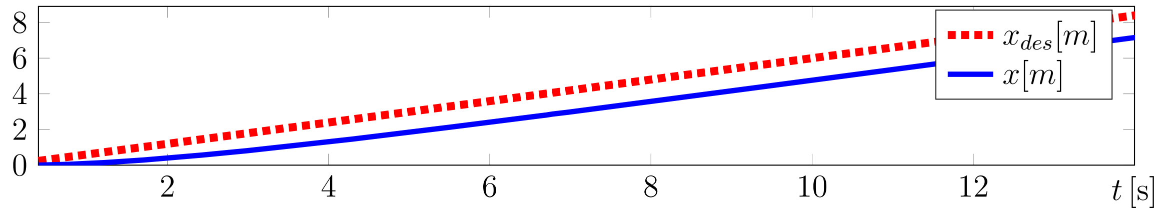
$$N = 10,\; h \;=\; 0.001\,\mathrm{s},\; h \;=\; 10^{-8},\; i_{max} = 10. \tag{5.37}$$

To analyze the tracking behavior, *OCP* (5.31)-(5.35) is used to track a target which is constantly moving in ${}^{\mathcal{G}}x$-direction

$$\boldsymbol{x_{des}}(t) = \begin{bmatrix} 0.2\,\mathrm{m\,s^{-1}} \cdot t & 0 & 1.5 & 0 & 0 \end{bmatrix}. \tag{5.38}$$

In order to exacerbate the protraction effect, the penalty values (5.35) are chosen to track zero velocity $\boldsymbol{v} = \mathbf{0}$ (5.38). The penalty matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are chosen to limit the drone movement to the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane by highly penalizing deviations from ${}^{\mathcal{G}}z_{des}$. The simulative evaluation using system model (5.32) is shown in Figure 5.11. The x-channel in Figure 5.11a is showing the expected position offset. The resulting Euclidean position tracking error e_p (5.22) in Figure 5.11b is stating the convergence towards a constant value. From a control perspective, this proportional offset can be treated with an integral part, as shown in the following.

(a) *AR.Drone 2.0* ${}^{\mathcal{G}}x$-Position



(b) *AR.Drone 2.0* position tracking error

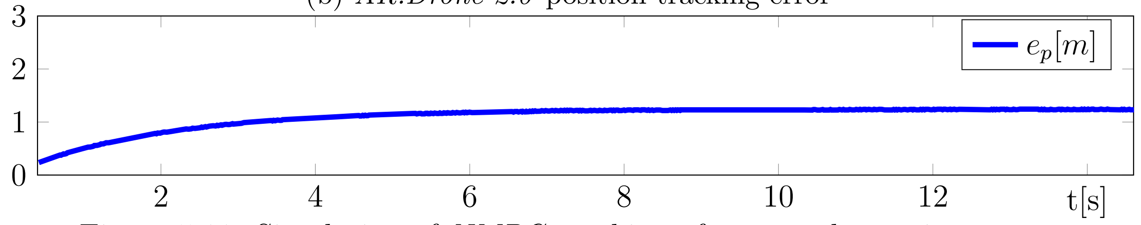


Figure 5.11: Simulation of *NMPC* tracking of constantly moving target

Target position control

In order to achieve an offset-free tracking, a target position control (*TPC*) is developed. The first step is hereby to model the closed-loop system. For this purpose, each *UAV* direction is considered independently in this context. Furthermore, the control and velocity penalty values are assumed to cause a well-damped closed-loop behavior as shown in Figure 5.11a. This allows the approximation of the *UAV* trajectory with a *PT1*. In the Laplace domain, this is expressed by system function

$$x\left(t\right) = (1 - e^{-\frac{t}{t_s}})x_{des}\left(t\right) \;\circ\!\!-\!\!\bullet\; x\left(s\right) = \frac{1}{1 + t_s s}x_{des}\left(s\right) \tag{5.39}$$

Variables of the Laplace domain are marked by the Laplace operator as argument s. Under use of the constant velocity v_0 step $\epsilon\left(t\right)$, the position trajectory can be described as a ramp

$$x_{des}\left(t\right) = v_0 \cdot t \cdot \epsilon\left(t\right) \;\circ\!\!-\!\!\bullet\; x_{des}\left(s\right) = \frac{v_0}{s^2}. \tag{5.40}$$

Inserting (5.40) into (5.39) leads to the system response

$$x(s) = \frac{\frac{1}{t_s}}{(\frac{1}{t_s} + s)s^2} \tag{5.41}$$

The inverse Laplace transformation of (5.41) results in the system response

$$x(t) = v_0 t_s \left(e^{-\frac{t}{t_s}} + \frac{t}{t_s} - 1 \right) \epsilon(t) = v_0 t + v_0 t_s \left(e^{-\frac{t}{t_s}} - 1 \right) \tag{5.42}$$

This system response is illustrated in Figure 5.12 for arbitrarily chosen $t_s = 2\,\mathrm{s}$ and $v_0 = 0.2\,\mathrm{m\,s^{-1}}$. As it shows the same behavior as the simulation of the ${}^{\mathcal{G}}x$-position in Figure 5.11, this does not prove the consistency of the model, but justifies the modeling approach.

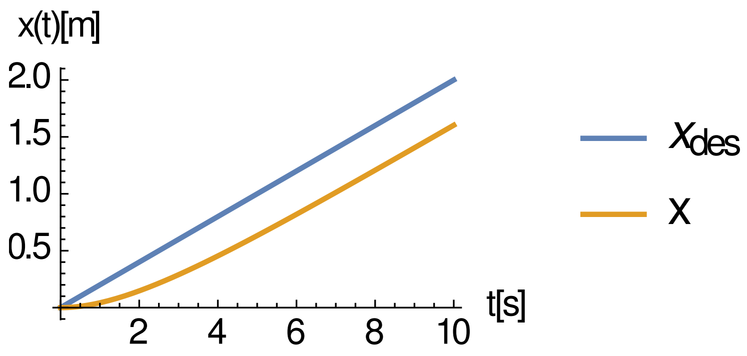


Figure 5.12: Analytically determined system response for $t_s = 2\,\mathrm{s}$, $v_0 = 0.2\,\mathrm{m\,s^{-1}}$

To get rid of this static tracking error, the basic concept is an adaptation of the target position x_{des}. For this reason the system is extended with an outer control loop according to Figure 5.13. For this purpose the new input w and error $e = w - x$ is introduced. The trajectory reference signal (5.38) is directed to the new input w. Figure 5.13 illustrates the idea for a displacement in x-direction. Instead of

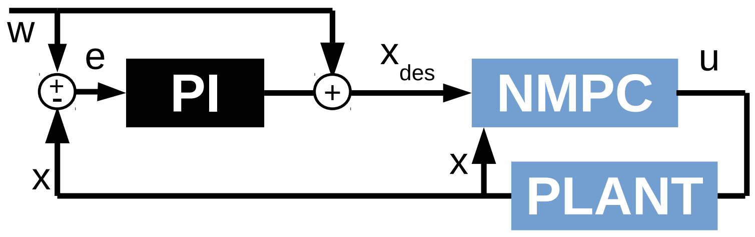


Figure 5.13: *TPC* control structure

the actual target w, an altered target x_{des} is given to the *NMPC*. This alteration

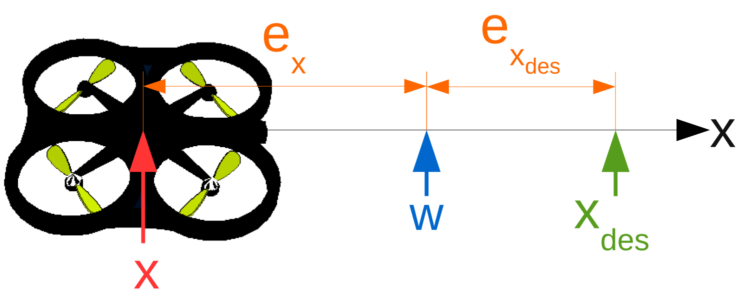


Figure 5.14: *TPC* idea

x_{des} is increasing with the tracking error e_x The resulting system response is higher, which leads to a convergence towards w. A linear interpolation of the target position (P-part) and an integral (I-part) leads to the control law

$$x_{des}(t) = w(t) + k_p \cdot e(t) + k_i \cdot \int_0^t e(t)\, dt \tag{5.43}$$

$$\circ\!\!-\!\!\bullet\; x_{des}(s) = w(s) + k_p \cdot e(s) + k_i \cdot \frac{e(s)}{s}. \tag{5.44}$$

Based on the system control can be composed as

$$x(s) = \frac{1}{1+t_s s}\left(w(s) + k_p \cdot (w(s) - x(s)) + k_i \cdot \frac{(w(s) - x(s))}{s}\right) \tag{5.45}$$

$$\tag{5.46}$$

which yields

$$\frac{x(s)}{w(s)} = \frac{k_i + (1+k_p)s}{k_i + (1+k_p)s + s^2 t_s}. \tag{5.47}$$

For the input signal of

$$w(t) = v_0 \cdot t \;\circ\!\!-\!\!\bullet\; \frac{v_0}{s^2}, \tag{5.48}$$

the system response results in

$$x(s) = \frac{v_0(k_i + k_p s + s)}{s^2(k_i + s(k_p + s t_s + 1))} \tag{5.49}$$

which represents the time domain signal

$$x(t) = \frac{v}{r}\left(\left(t_s - t_s e^{\frac{rt}{t_s}}\right) e^{-\frac{t(k_p+r+1)}{2t_s}} + rt\right) \tag{5.50}$$

$$\text{with } r = \sqrt{(k_p+1)^2 - 4k_i t_s}. \tag{5.51}$$

This signal shows the desired convergence, as shown in Figure 5.12. Furthermore,

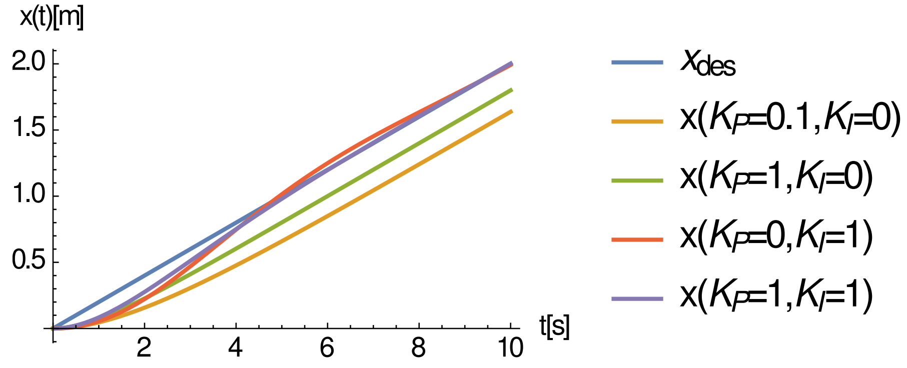


Figure 5.15: *TPC* system responses for different parametrization for $t_s = 2\,\mathrm{s}$ and $v_0 = 0.2\,\mathrm{m\,s^{-1}}$

Figure 5.12 shows that higher k_p leads to smaller tracking errors. However, convergence for $k_p < \infty$ can be only achieved with an integral part $k_i > 0$. Due to the underlying *MPC* and the three dimensional problem, where ${}^{\mathcal{G}}x$, ${}^{\mathcal{G}}y$, ${}^{\mathcal{G}}z$ are coupled, the controller cannot be parametrized by the typical root locus analysis. One reason is, that the integral part leads to overshooting in the quadrotor trajectory at direction changes. Accordingly, the integral part k_i is chosen to be very small, while the convergence is accelerated by using high proportional gain k_p. Due to the stability properties of the internal *MPC* controller, k_p cannot be chosen arbitrarily high. For the considered *UAV*/*MPC* the parameters have been empirically chosen for the ${}^{\mathcal{G}}x$-channel to $k_p = 2$ and $k_i = 0.001$. The simulation with the *TPC* is shown in Figure 5.16. The resulting x-trajectory in Figure 5.16a shows a smaller offset than without *TPC* (see Figure 5.11a). As expected, the tracking error e_p is converging towards zero. Hence, the next step is the validation with a real *UAV*.

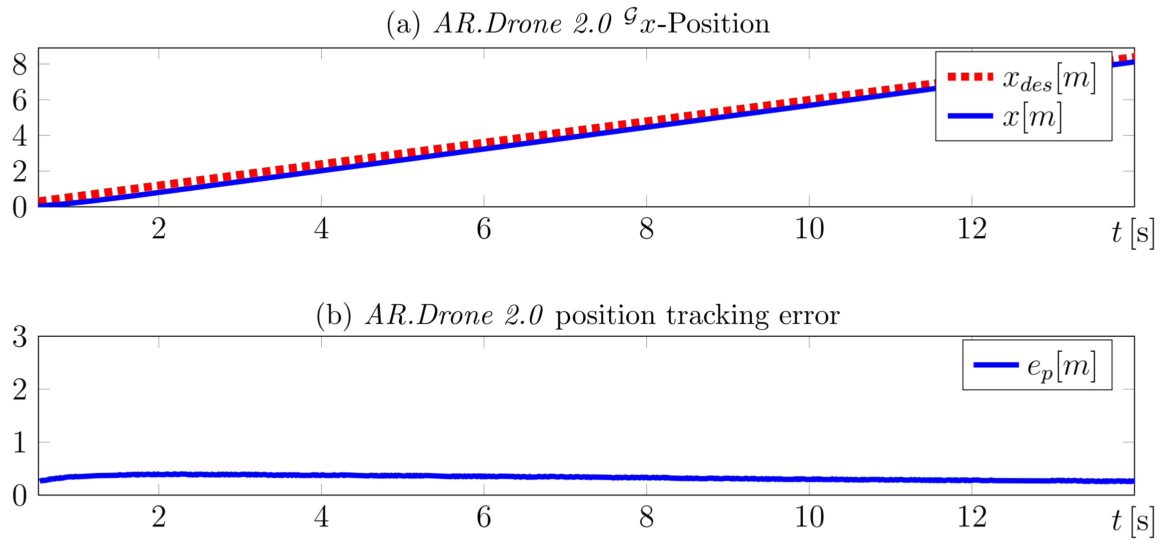


Figure 5.16: Simulation of tracking a constantly moving target with *TPC*

Experimental validation

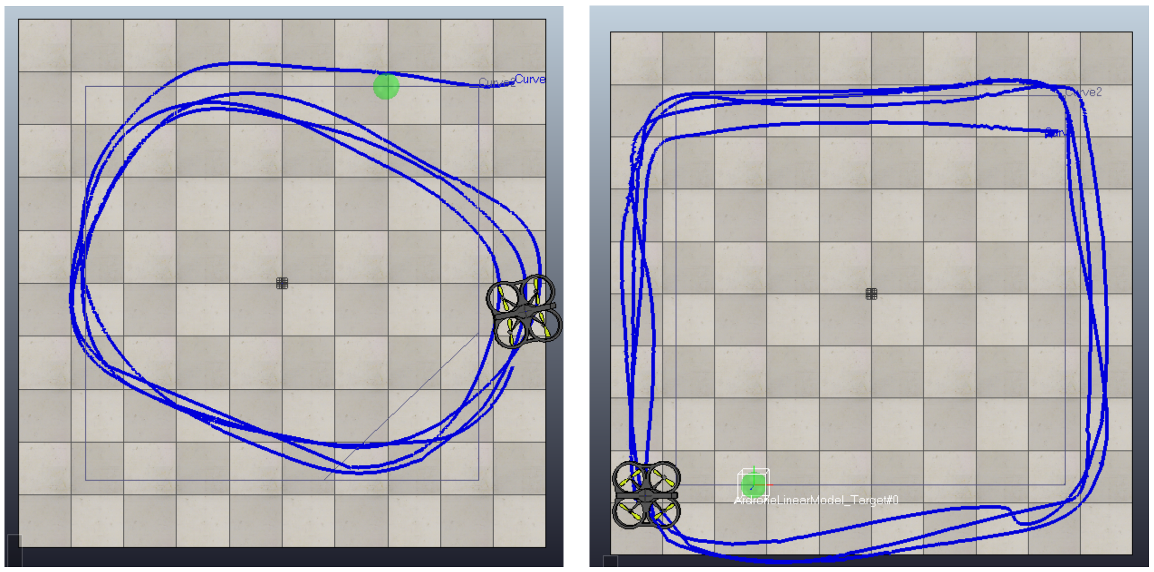


Figure 5.17: Visualization of the experimental data of tracking of a target, moving in a square with a constant velocity

The validation experiment consists of an *AR.Drone 2.0* controlled by the inner *MPC* control of §5.3.1 and the outer *TPC* of §5.4. The position of the quadrotor is measured with the *OPTITRACK* motion capture system, which limits the experimental space. For this reason, the desired trajectory is chosen as square with a side length of 1.5 m in an altitude of ${}^{\mathcal{G}}z_{des} = 1.5\,\mathrm{m}$. The target position is changed with a constant velocity of $v_0 = 0.2\,\mathrm{m\,s^{-1}}$. Figure 5.17 is showing on the left side the resulting ellipsoidal trajectory without the proposed *TPC*. The corresponding system trajectories are shown in Figure 5.18. As expected, ${}^{\mathcal{G}}x$ and ${}^{\mathcal{G}}y$ position are responding delayed to ${}^{\mathcal{G}}x_{des}$ and ${}^{\mathcal{G}}y_{des}$. The resulting tracking error of $e_p \approx 2\,\mathrm{m}$ shows a ripple which is caused by the direction changes in the square corners.

To reduce the tracking error e_p (5.22), the proposed *TPC* is implemented. As the prediction model (5.32) shows asymmetric model parameters for the ${}^{\mathcal{G}}x$ and ${}^{\mathcal{G}}y$ channel, the *TPC* is adapted to each channel individually. The parameters for the ${}^{\mathcal{G}}x$-channel are empirically chosen to $k_{px} = 2$, $k_{ix} = 0.001$ and for the ${}^{\mathcal{G}}y$-channel $k_{py} = 3$, $k_{iy} = 0.001$. The resulting trajectory in Figure 5.17 (right) states the effectiveness of the *TPC* by a better tracking of the square trajectory. In the corners of the square, the integral part leads to overshooting. After the corner, the quadrotor converges towards the square edges. The corresponding trajectory shows a much lower tracking error e_p (5.22), as given in Figure 5.19. This is caused by the smaller delay in the ${}^{\mathcal{G}}x$ and ${}^{\mathcal{G}}y$ response. At each corner the tracking error is increased due to the direction change of the target trajectory. After the corner, it is converging against zero as desired. Hence, the experiment is validating the desired *TPC* for constantly moving targets.

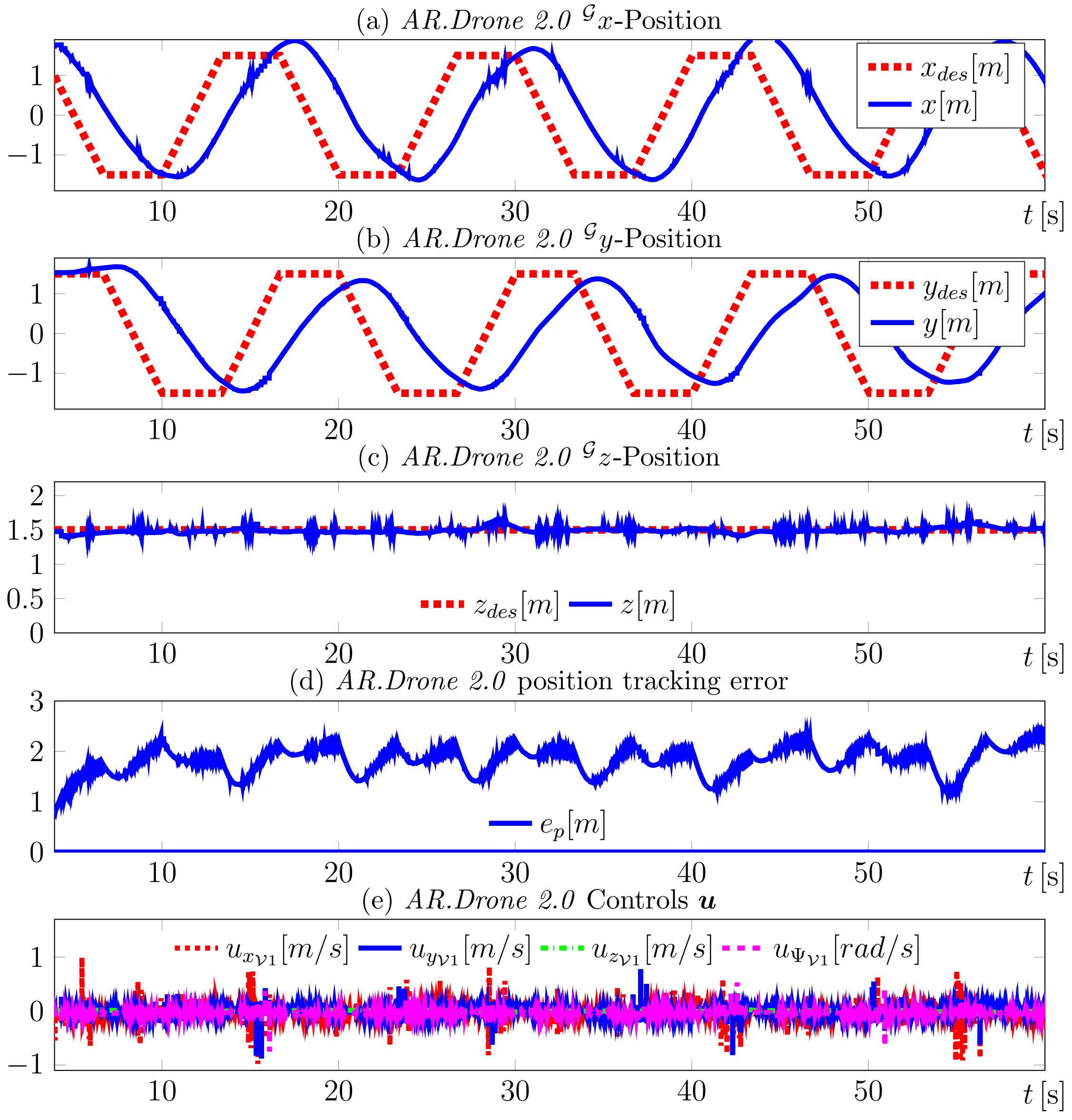


Figure 5.18: Real *AR.Drone 2.0* square tracking without *TPC*

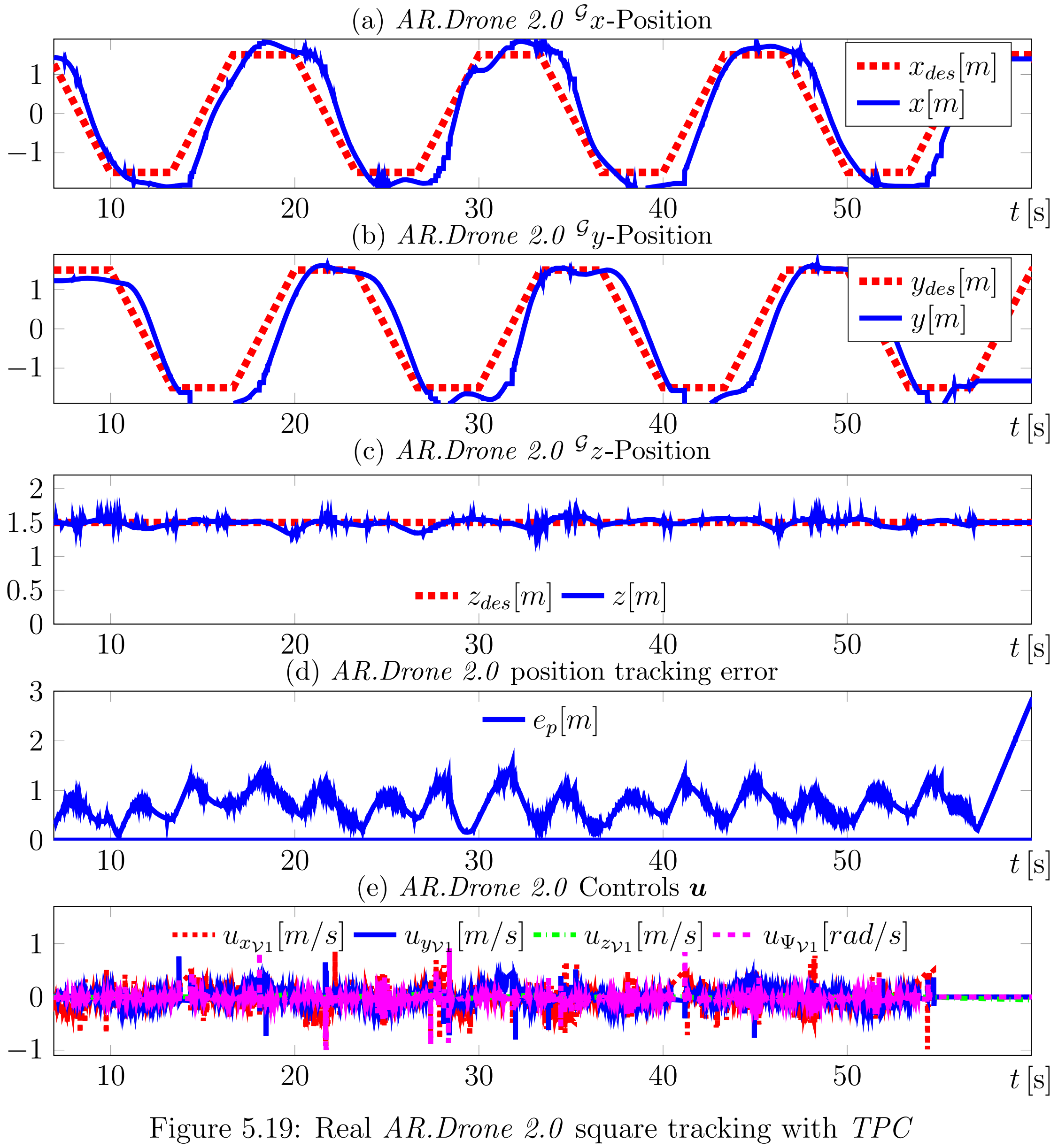


Figure 5.19: Real *AR.Drone 2.0* square tracking with *TPC*

5.5 Model predictive collision avoidance

In order to deploy *UAV*s in cluttered environments, the capability to avoid collisions is essential for safety reasons. A typical *CA* approach is to keep a minimal Euclidean distance d_{des} between an obstacle ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o$ and the *UAV*'s position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}$. This problem can be formulated as inequality constraint (5.52) which can be translated with $0 \leq c_{in}$ into a constraint (5.53). To avoid the expensive square root computation in (5.53), it is advantageous to use the quadratic form (5.54) instead. This is possible as d_{des} is always positive, wherefore (5.54) and (5.53) have equivalent roots .

$$d_{des} \leq \sqrt{\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\mathrm{T}} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)} \tag{5.52}$$

$$c_{in} \leq d_{des} - \sqrt{\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\mathrm{T}} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)} \tag{5.53}$$

$$c_{in} \leq d_{des}^2 - \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\mathrm{T}} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) \tag{5.54}$$

If the safety distance d_{des} is chosen large enough, minor violations are acceptable. This is advantageous, as a soft constraint can be used, as shown in §4.5.3. The soft constraint is hereby necessary to avoid an infeasible *OCP* solutions if d_{des} is violated due to initial conditions or disturbance. According to the *sig* approximation in §4.5.3 the constraint (5.54) can be transformed to the additional stage cost term

$$L_{CA} = \frac{\kappa_H}{1 + e^{-\kappa_A \left(d_{des}^2 - \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\mathrm{T}} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)\right)}} \tag{5.55}$$

The parameters κ_H, L_{CA} can be adapted to the system costs J. Accordingly κ_H has to be chosen big enough to have dominating costs L_{CA} to ensure that CA is prioritized in relation to trajectory tracking. Figure 5.20 is showing the influence of the parameter

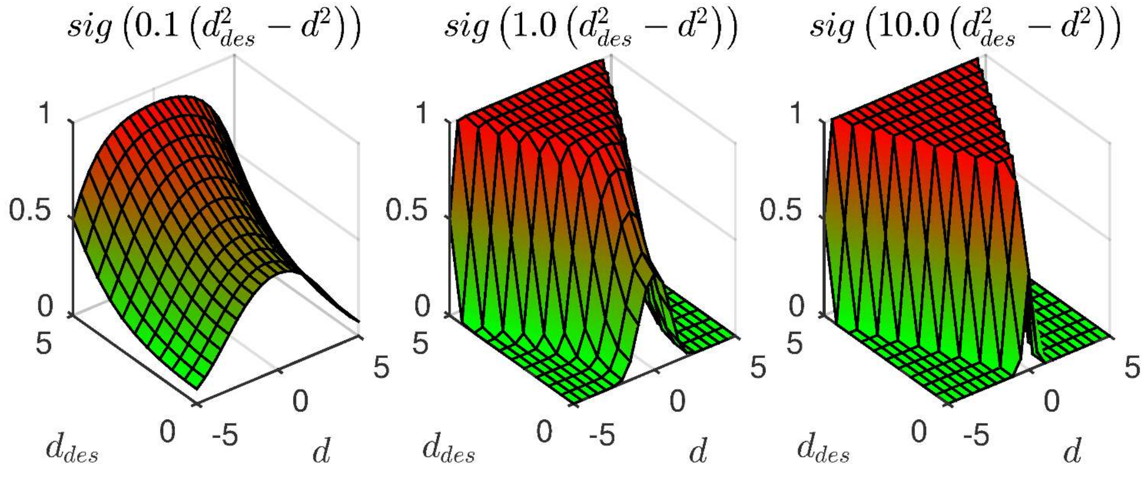


Figure 5.20: Sigmoid distance cost function L_{CA} with $\kappa_H = 1$

κ_A which is affecting the sharpness of the switching behavior. For increasing κ_A values (Figure 5.20 from left to right), the switching behavior is approximated better, but the system becomes more ill-conditioned and therefore more difficult to solve. d is representing the distance $|{}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}|$ and d_{des} the desired distance that should be kept. If the UAV is in the prohibited area $d < d_{des}$ (top area in Figure 5.20), the system function is dominated by (5.55). As a consequence, the solver preferably tries to minimize L_{CA} (5.55) and therefore increases the distance d to the obstacle. The validation of the proposed sigmoid CA is shown in the following §5.5.

Experimental validation of NMPC collision avoidance

For the experimental validation a static obstacle is introduced at position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o = [0, 0, 1]^{\mathrm{T}}$. The UAV is tracking position changes between ${}^{\mathcal{G}}p_{des} = [0, -1, 1]^{\mathrm{T}}$ and ${}^{\mathcal{G}}p_{des} = [0, 1, 1]^{\mathrm{T}}$. According to the CA, the UAV has to avoid the obstacle in the center in order to reach the tracked position. The result is a collision evasion curve around the obstacle as shown in Figure 5.21.

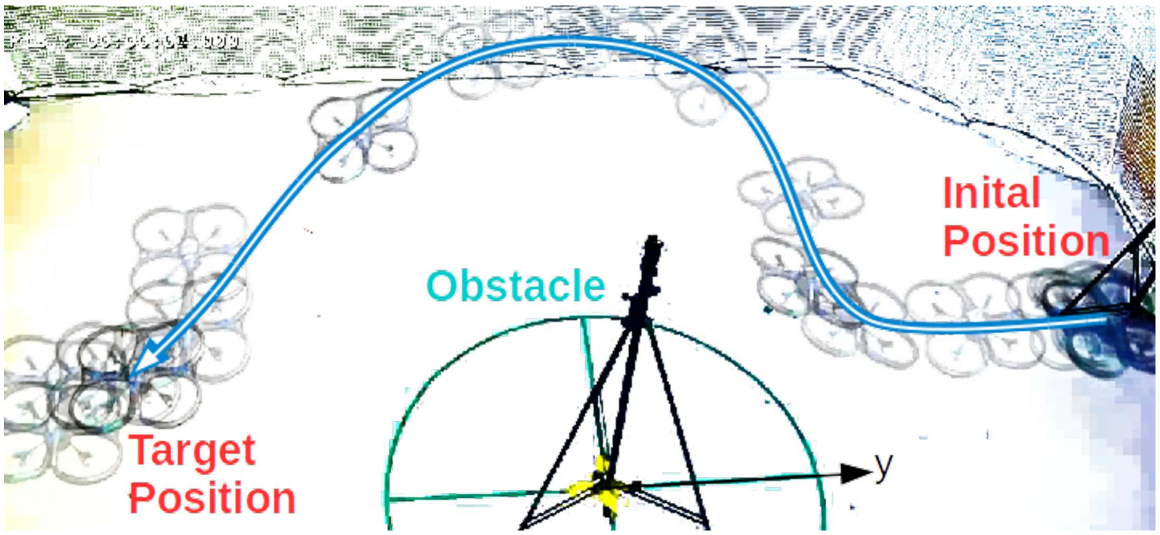


Figure 5.21: *AR.Drone 2.0 CA*: the trajectory of the quadrotor is deviated by an obstacle, depicted as stand in the center point of the circle. As desired, the drone is avoiding the circular keep out area with radius $r = 1\,\mathrm{m}$

The experimental validation has been conducted in the context of [DKMV16b] with a deprecated version of the $RHMY$ and parametrization (3.29). Under use of

L_{CA} (5.55), the related *OCP* with the *CA* extension is given as

$$\min_{\boldsymbol{u}} \; J = \int\limits_{t}^{t+T} \left(\underline{\boldsymbol{x}_{\boldsymbol{des}}} - \underline{\boldsymbol{x}}\right)^{\mathrm{T}} \boldsymbol{Q} \left(\underline{\boldsymbol{x}_{\boldsymbol{des}}} - \underline{\boldsymbol{x}}\right) + \underline{\boldsymbol{u}}^{\mathrm{T}} \boldsymbol{R} \underline{\boldsymbol{u}}$$

$$+ \frac{\kappa_H}{1 + e^{-\kappa_A \left(d_{des}^2 - \left({}^{\mathcal{G}}\vec{\boldsymbol{p}}_o - {}^{\mathcal{G}}\underline{\vec{\boldsymbol{p}}}_{UAV}\right)^{\mathrm{T}} \left({}^{\mathcal{G}}\vec{\boldsymbol{p}}_o - {}^{\mathcal{G}}\underline{\vec{\boldsymbol{p}}}_{UAV}\right)\right)}} \; d\tau$$

$$s.\ t. \tag{5.56}$$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} {}^{\mathcal{V}1}\underline{\dot{x}}\,(t)\cos\left({}^{\mathcal{V}}\underline{\Psi}\,(t)\right) - {}^{\mathcal{V}1}\underline{\dot{y}}\,(t)\sin\left({}^{\mathcal{V}}\underline{\Psi}\,(t)\right) \\ {}^{\mathcal{V}1}\underline{\dot{x}}\,(t)\sin\left({}^{\mathcal{V}}\underline{\Psi}\,(t)\right) + {}^{\mathcal{V}1}\underline{\dot{y}}\,(t)\cos\left({}^{\mathcal{V}}\underline{\Psi}\,(t)\right) \\ \underline{u}_{\mathcal{V}1_z}\,(\tau) \\ 1.6\underline{u}_{\omega_{\mathcal{V}1}}\,(\tau) \\ -0.5092 \cdot {}^{\mathcal{V}1}\underline{\dot{x}}\,(t) + 1.458 \cdot \underline{u}_{\mathcal{V}1_x}\,(t) \\ -0.5092 \cdot {}^{\mathcal{V}1}\underline{\dot{y}}\,(t) + 1.458 \cdot \underline{u}_{\mathcal{V}1_y}\,(\tau) \end{bmatrix} \tag{5.57}$$

input limits handled with auxiliary variably constraint (§4.5.2): $|u| < 1$:

$$\boldsymbol{0} \leq \boldsymbol{c_{in}} = \left[\underline{u}_{\mathcal{V}1_x}{}^2 - 1, \underline{u}_{\mathcal{V}1_y}{}^2 - 1, \underline{u}_{\mathcal{V}1_z}{}^2 - 1, \underline{u}_{\mathcal{V}1_\omega}{}^2 - 1\right]^{\mathrm{T}} \tag{5.58}$$

$$\underline{\boldsymbol{x}}_0\,(t) = \boldsymbol{x}\,(t)\,, \qquad \boldsymbol{x}\,(0) = \left[0, 0, 0, 0, 0, 0\right], \tag{5.59}$$

$$\boldsymbol{Q} = \mathrm{diag}\left(\left[1, 1, 1, 10, 0, 0\right]\right), \qquad \boldsymbol{R} = \mathrm{diag}\left(\left[1.3, 1.3, 3.0, 1.1\right]\right), \tag{5.60}$$

$$\boldsymbol{r}_\nu = \left[0.001, 0.001, 0.001, 0.001\right] \tag{5.61}$$

$$\xi = 1,\ \upsilon = 2,\ \Delta t = 0.01\,\mathrm{s}, T = 1\,\mathrm{s}, \tag{5.62}$$

$$N = 10,\ h = 0.001\,\mathrm{s},\ \epsilon = 10^{-8},\ i_{max} = 10 \tag{5.63}$$

For the experimental validation L_{CA} (5.55) is parametrized with $\kappa_A = 6$ and $\kappa_H = 3$ which have been chosen empirically. Furthermore, to limit collision evasion maneuvers to take place in the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane, the ${}^{\mathcal{G}}z$-axis tracking penalty $r_z = 3$ is set higher in comparison to the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-tracking penalties.

The resulting trajectories are given in Figure 5.22-5.23. In each figure three sections are separated by vertical lines. Each of these sections is representing a change in the target position. Figures 5.22a-5.22c show that the main evasion takes place in the ${}^{\mathcal{G}}y$ direction. The ${}^{\mathcal{V}}\Psi$-axis is stabilized in ${}^{\mathcal{V}}\Psi = 0$ as shown in Figure 5.22d. At first, the *UAV* is just moving in ${}^{\mathcal{G}}y$-direction until it approaches the *CA* sphere. Then the ${}^{\mathcal{G}}x$-axis is deployed to initiate the *CA* curve. When reaching the obstacle avoidance sphere with $d < d_{min} = 1\,\mathrm{m}$, the repulsive behavior of the *CA* constraint pushes the quadrotor away from this sphere. Until a feasible path is found, this leads to position

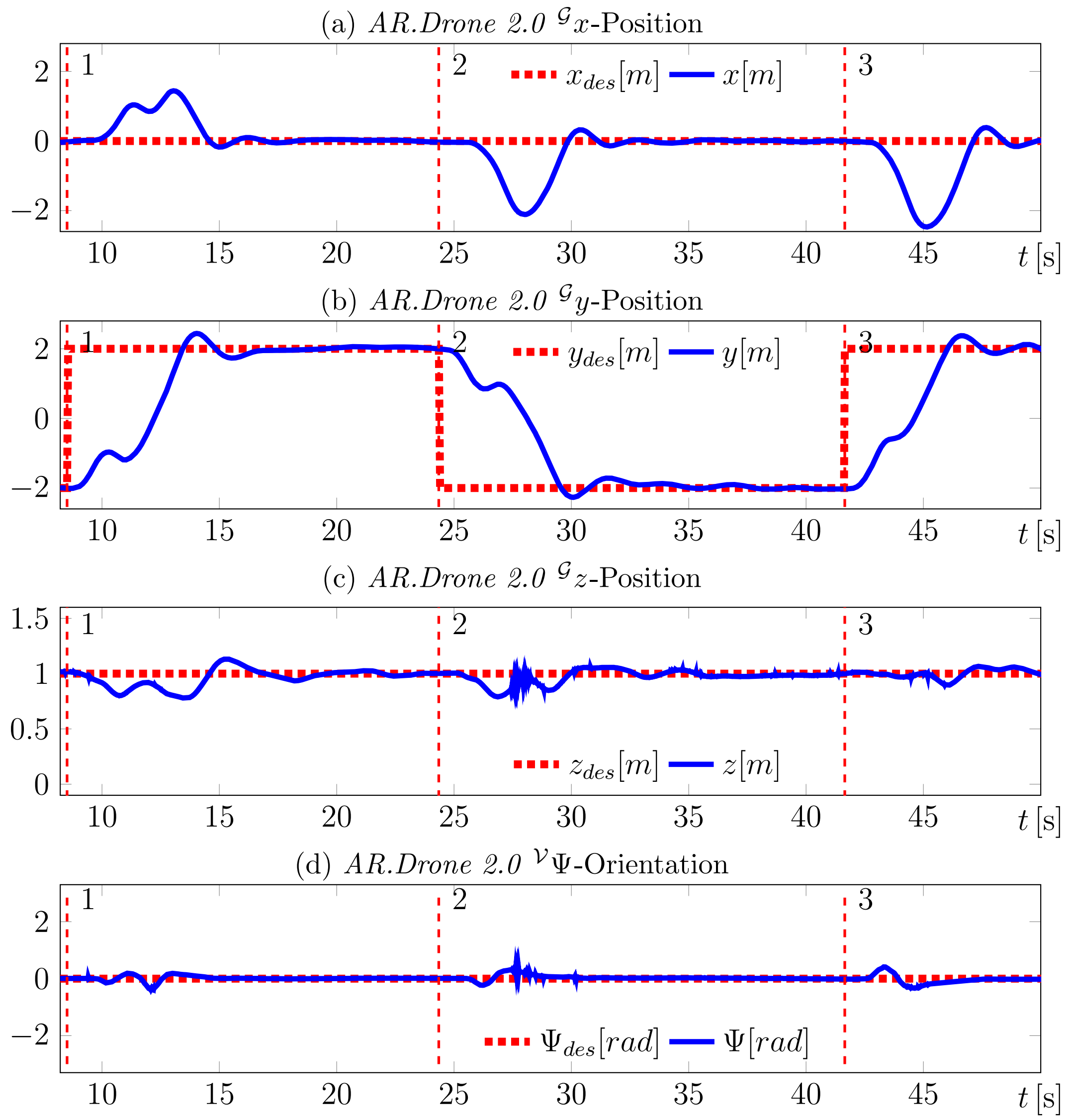


Figure 5.22: *AR.Drone 2.0 CA*: pose trajectory

oscillations which are caused by a wrong prediction due to the model error. These oscillations can be decreased by using a more precise model, further smoothening of the sigmoid approximation or a reduction of the control action.

The distance to the obstacle (Figure 5.23a) states a slight violation of the soft constraint $d_{min} = 1\,\text{m}$. The stabilization of the *NMPC* is shown in the tracking errors which are converging towards zero as shown in Figure 5.23b. The corresponding inputs in Figure 5.23c state the effectiveness of the input interval limitation with the auxiliary variable constraint handling. Figure 5.23d is showing the computation time t_c. The maximum $max(t_c) = 1.6\,\text{ms}$ and average computation time of $\bar{t_c} = 0.7709\,\text{s}$ are stating the real-time capability, effectiveness and efficiency of the proposed *CMSC/GMRES* + sigmoid-*CA* combination.

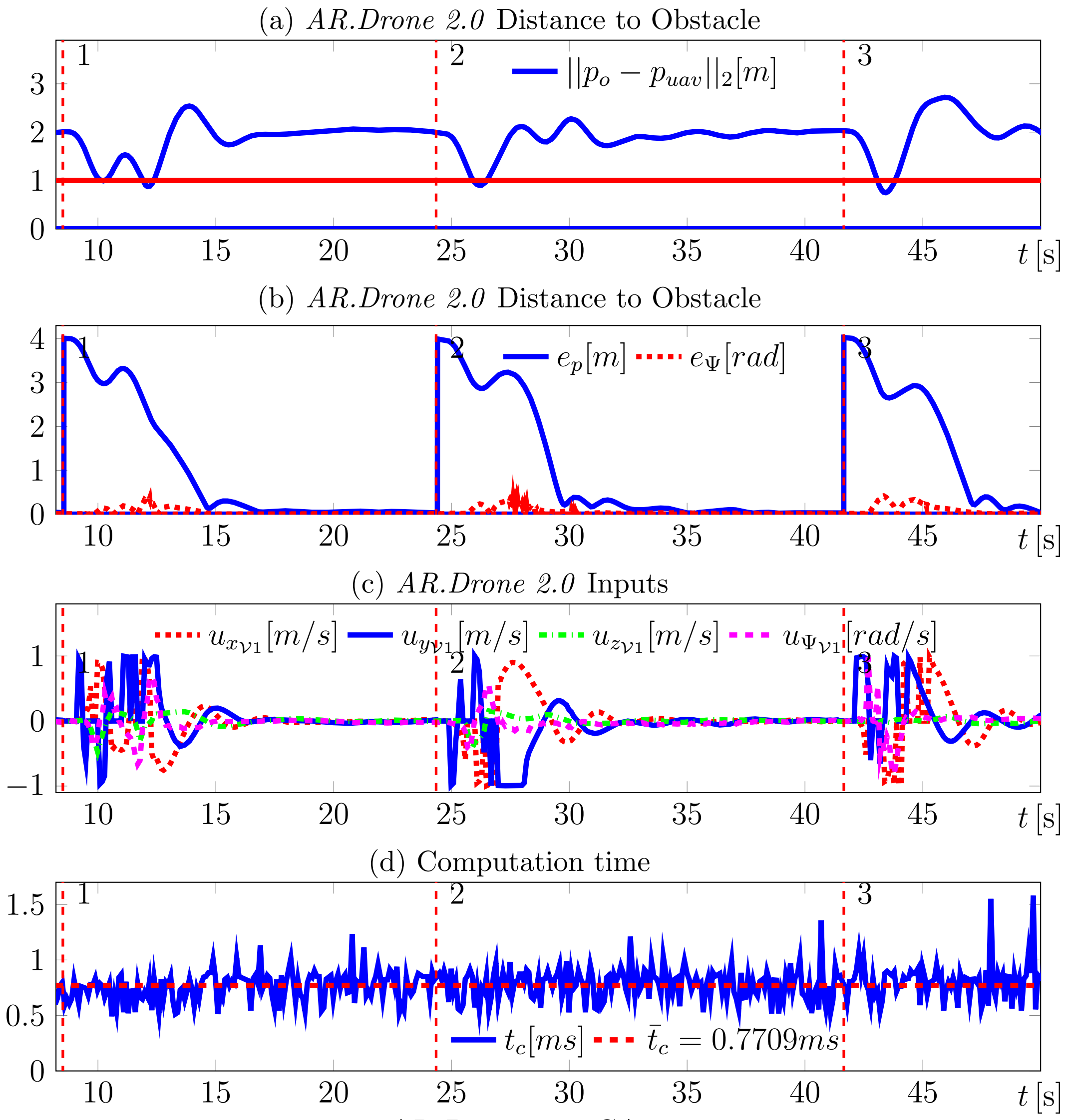


Figure 5.23: *AR.Drone 2.0 CA*: trajectories

5.6 Task-based control using potential functions

This section is extending the use of *NMPC* for pose tracking of §5.3 and the soft *CA* constraint formulation of §5.5 to more complex tasks as e.g. sensor limitations. One major difficulty to control complex tasks autonomously is their mathematical formulation. *MPC* is offering the possibility to formulate such tasks with the help of constraints as shown in §4.5. The consideration of inequality constraints in *MPC* requires computational overhead which threatens the real-time capability of *MPC* for fast systems. For this reason, this section is presenting a constraint design workflow to describe tasks in terms of soft constraints according to §4.5.3. In practice, this refers to the substitution of hard inequality constraints by a cost function that imposes a repulsive behavior from the violation of the constraint. This implementation in the cost function equals a potential function. In this section, a generic procedure to create such potential functions is presented.

As use case for the constraint design process, the *UAV* scenario shown in Figure 5.24 is considered. In this use case, the control goal is to keep an object within the limited sensor perception space of a quadrotor. In contrast to the track a certain pose (see §5.3), the objective here is to track a certain pose region. The sensor perception space is assumed to be convex and shaped like a cone (e.g. ultrasonic distance sensor). An *AR.Drone 2.0* quadrotor is used as *UAV* and represented with the reduced direction vector dynamics (3.37).

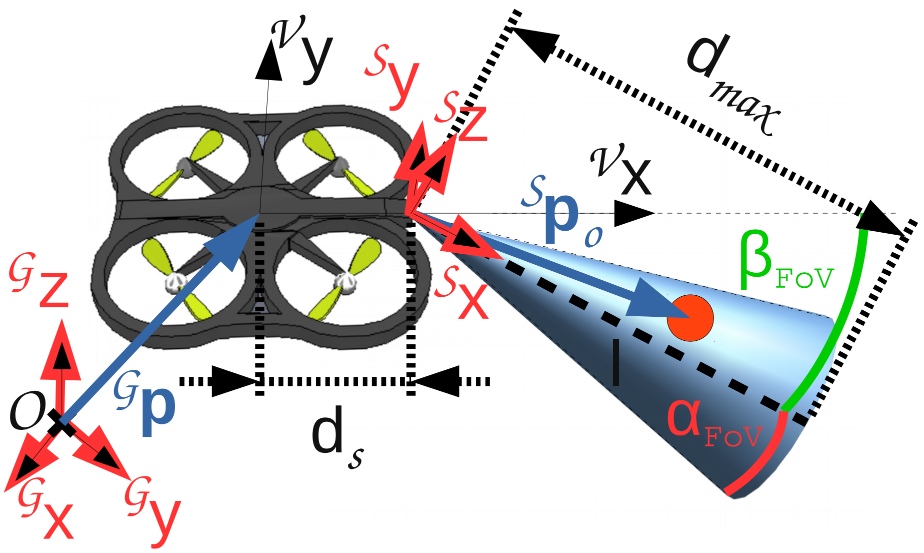


Figure 5.24: Use case scenario: tracking with cone-shaped sensor perception of a quadrotor

The task formulation can be separated into following steps:

1 Task formulation in the task frame with inequality constraints (§5.6.1)

2 Composition of the cost function using step functions to approximate the switching behavior of inequality constraint (§5.6.2)

3 Introduction of artificial gradients to improve convergence to desired cost function region (§5.6.3)

4 Approximation the step function with analytical functions (§5.6.4)

5 Transformation from the task frame into the global frame perspective (§5.6.5)

6 Adding safety measures (§5.6.6)

which are detailed in the following sections.

5.6.1 Formulating task with inequality constraints

The first step to implement the sensor-based control is to formulate the task in terms of inequality constraints. For the use case, the object has to be tracked within the given perception space limits. The task frame $\mathcal{S}$ is here accordingly the sensor frame. Accordingly, a target object is defined in the sensor coordinate frame $\mathcal{S}$ with the target position vector ${}^{\mathcal{S}}\vec{\boldsymbol{p}}_o \in \mathcal{S}$

$$
{}^{\mathcal{S}}\vec{\boldsymbol{p}}_o = \begin{bmatrix} {}^{\mathcal{S}}x_o \\ {}^{\mathcal{S}}y_o \\ {}^{\mathcal{S}}z_o \end{bmatrix}^\top \in \mathcal{R}^3. \tag{5.64}
$$

Subsequently, the field of view of the sensor can be expressed by the inequality constraints

$$
0 \geq c_{in1}\left({}^{\mathcal{S}}\vec{\boldsymbol{p}}_o, \alpha_{FoV}\right) = {}^{\mathcal{S}}y_o^2 + {}^{\mathcal{S}}z_o^2 - \left({}^{\mathcal{S}}x_o \sin(\alpha_{FoV})\right)^2 \tag{5.65}
$$

$$
0 \geq c_{in2}\left({}^{\mathcal{S}}\vec{\boldsymbol{p}}_o\right) = -{}^{\mathcal{S}}x_o. \tag{5.66}
$$

In this example (5.65) is representing a double cone with the sensor beam width angle α_{FoV}. To receive a single cone perception space, constraint (5.66) is limiting the cone to the positive half plane of ${}^{\mathcal{S}}x_o$. In contrast to just pointing the quadrotor into the target direction, the formulation of the perception space tracking offers more flexibility to the *MPC* to optimize the energy consumption and to adapt the scenario to other constraints e.g. obstacles. The task constraints (5.65)-(5.66) are considered in the *OCP* by means of potential functions.

5.6.2 Inequality transformation to potential function

Considering the constraints (5.65)-(5.66) as hard constraints is problematic, because their violation would lead to an infeasible optimization problem. In the sensor-based tracking scenario, this would be the case if the object is outside the sensor perception space. Such a violation is possible due to disturbance or an infeasible initial pose. For this reason, the sensor constraints are designed as soft constraints with the saturation function constraint handling discussed in §4.5.3. However, for validation purposes it is advantageous to do this step-wise by first using step functions to approximate the inequality constraint switching behavior. For the task constraints (5.65)-(5.66) this leads to

$$\begin{aligned} &L_{c_{in}}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}, \alpha_{FoV}\right) = \epsilon\left(c_{in2}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}\right)\right) + \epsilon\left(c_{in1}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}, \alpha_{FoV}\right)\right)\epsilon\left(-c_{in2}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}\right)\right) \\ &= \epsilon\left(-{}^{\mathcal{S}}x_{o}\right) + \epsilon\left({}^{\mathcal{S}}y_{o}^{2} + {}^{\mathcal{S}}z_{o}^{2} - \left({}^{\mathcal{S}}x_{o}\sin(\alpha_{FoV})\right)^{2}\right)\epsilon\left({}^{\mathcal{S}}x_{o}\right) \end{aligned} \tag{5.67}$$

c_{in2} is used to distinguish between the negative and positive half-space of ${}^{\mathcal{S}}x_{o}$:

$$L_{c_{in}}(c_{in2}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}\right) > 0 \to {}^{\mathcal{S}}x_{o} \in \mathbb{R}^{-}) = 1 \tag{5.68}$$

$$L_{c_{in}}(c_{in2}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}\right) \leq 0 \to {}^{\mathcal{S}}x_{o} \in \mathbb{R}^{+}) = c_{in1}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}, \alpha_{FoV}\right) \tag{5.69}$$

Figure 5.25 is validating the soft cone constraint (5.67), where the area in the cone has the potential value $L_{c_{in}} = 0$ and areas outside are penalized with $L_{c_{in}} = 1$. This facilitates the validation of the desired behavior, but poses the problem of zero gradient descent.

5.6.3 Adding cost gradient

The left plot in Figure 5.25 is showing the costs $L_{c_{in}}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}, \alpha_{FoV}\right)$ (5.67) in the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane. As expected, the gradient satisfies

$$\nabla L_{c_{in}}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}, \alpha_{FoV}\right) = \mathbf{0} \qquad \forall c_{in1}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_{o}, \alpha_{FoV}\right) \neq 0. \tag{5.70}$$

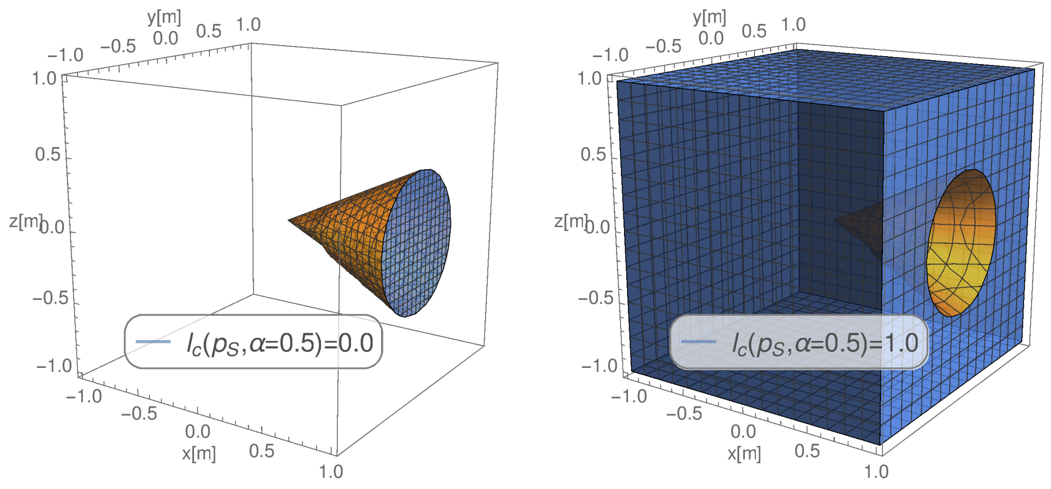


Figure 5.25: Unit step penalty function values of cone constraint

Most *MPC* solvers are exploiting the gradient to find the optimal solution. Hence, this property becomes problematic, as the convergence of the search algorithm cannot be guaranteed with zero gradient descent. To address this issue the next step is to add a cost gradient, which is indicating the direction of the compliant region.

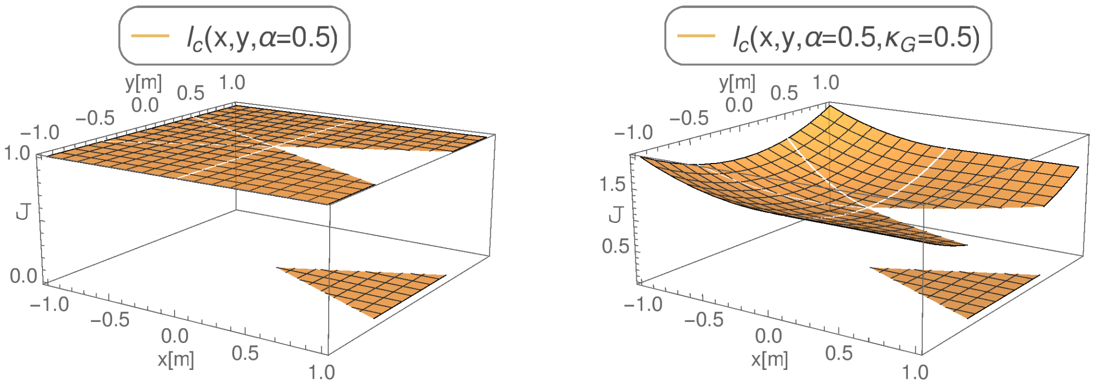


Figure 5.26: Unit step penalty function of cone constraint in the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane

To impose a gradient for the use case, a simple quadratic penalty can be added to the non-compliant regions around the cone. Hence, (5.67) is extended to

$$\begin{aligned} L_{c_{in}}\left({}^{\mathcal{S}}\vec{\boldsymbol{p}}_o, \alpha_{FoV}, \kappa_G\right) &= \epsilon\left(-{}^{\mathcal{S}}x_o\right)\left(1+\kappa_G\left({}^{\mathcal{S}}x_o^2+{}^{\mathcal{S}}y_o^2+{}^{\mathcal{S}}z_o^2\right)\right) \\ &+ \epsilon\left({}^{\mathcal{S}}y_o^2+{}^{\mathcal{S}}z_o^2-\left({}^{\mathcal{S}}x_o\sin(\alpha_{FoV})\right)^2\right)\cdot\epsilon\left({}^{\mathcal{S}}x_o\right)\left(1+\kappa_G\left({}^{\mathcal{S}}y_o^2+{}^{\mathcal{S}}z_o^2\right)\right), \end{aligned} \tag{5.71}$$

whereby parameter κ_G is controlling the steepness of the gradient derivative. The result on the right side in Figure 5.26 shows the desired convergence towards the compliant triangular (cone) area. As a remark, gradient-based solvers typically require convex *OCP* problems to ensure convergence. Naturally, most sensor perception spaces are convex. However, this should be kept in mind for the development of more complex constraints or the combination of multiple constraints.

5.6.4 Addressing differentiability of the potential function

Finally, a sigmoid function is used to approximate the inequality constraint switching behavior. This results in the saturation function constraint handling, as discussed in §4.5.3. The transformation of the extended soft cone constraint (5.71) by means of a sigmoid function (4.137), finally yields to

$$
\begin{aligned}
&L_{c_{in}}\left({}^{\mathcal{S}}\vec{\boldsymbol{p}}_o, \alpha_{FoV}, \kappa_G, \kappa_A\right) = sig\left(-{}^{\mathcal{S}}x_o, \kappa_A\right)\left(1+\kappa_G\left({}^{\mathcal{S}}x_o^2 + {}^{\mathcal{S}}y_o^2 + {}^{\mathcal{S}}z_o^2\right)\right) \\
&+ sig\left({}^{\mathcal{S}}y_o^2 + {}^{\mathcal{S}}z_o^2 - \left({}^{\mathcal{S}}x_o \sin(\alpha_{FoV})\right)^2, \kappa_A\right) \cdot sig\left({}^{\mathcal{S}}x_o, \kappa_A\right)\left(1+\kappa_G\left({}^{\mathcal{S}}y_o^2 + {}^{\mathcal{S}}z_o^2\right)\right).
\end{aligned}
$$

The resulting cost function is shown in Figure 5.27 for $\kappa_A = 10$. To be able to track an object with known position in the global coordinate system, ${}^{\mathcal{S}}\vec{\boldsymbol{p}}_o$ in (5.67) has to be determined by its counterpart ${}^{\mathcal{G}}\vec{\boldsymbol{p}}_o$ in the global coordinate system. The required coordinate transformation is explained in the following section.

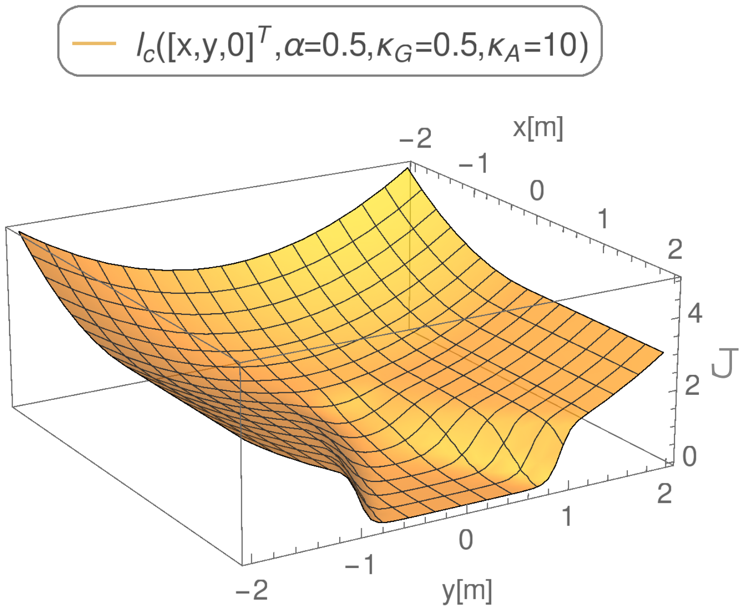


Figure 5.27: Extended soft cone constraint transformed with sigmoid

5.6.5 Coordinate relation

As shown in §2.2, the coordinate transformation from $\mathcal{G}$ to $\mathcal{S}$ can be described as a sequential transformation using homogeneous transformation matrices. Considering the hovering condition, the

$$\begin{aligned} &\text{robot position} && {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} = ({}^{\mathcal{G}}x, {}^{\mathcal{G}}y, {}^{\mathcal{G}}y)^\top \text{ and} \\ &\text{robot orientation} && {}^{\mathcal{G}}\boldsymbol{o} = ({}^{\mathcal{V}}o_x, {}^{\mathcal{V}}o_y)^\top \end{aligned}$$

in direction vector form are defining the UAV's pose. The kinematic chain from the global frame $\mathcal{G}$ into the sensor frame $\mathcal{V}1$ is given by (2.6), respectively (3.34). This results to

$$\begin{aligned} {}^{\mathcal{V}1}\boldsymbol{T}_{\mathcal{G}}\left({}^{\mathcal{G}}\boldsymbol{o}, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) &= \begin{bmatrix} o_x & o_y & 0 & 0 \\ -o_y & o_x & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -{}^{\mathcal{G}}x \\ 0 & 1 & 0 & -{}^{\mathcal{G}}y \\ 0 & 0 & 1 & -{}^{\mathcal{G}}z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} {}^{\mathcal{V}}o_x & {}^{\mathcal{V}}o_y & 0 & -{}^{\mathcal{V}}o_x{}^{\mathcal{G}}x - {}^{\mathcal{V}}o_y{}^{\mathcal{G}}y \\ -{}^{\mathcal{V}}o_y & {}^{\mathcal{V}}o_x & 0 & {}^{\mathcal{V}}o_y{}^{\mathcal{G}}x - {}^{\mathcal{V}}o_x{}^{\mathcal{G}}y \\ 0 & 0 & 1 & -{}^{\mathcal{G}}z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{5.72}$$

To further transform from $\mathcal{V}1$ into $\mathcal{S}$, the coordinates are translated translating (2.6) by distance d_s along ${}^{\mathcal{V}1}x$ and finally rotated (2.8) by the sensor inclination angle β_{FoV} into the sensor frame $\mathcal{S}$

$$\begin{aligned} {}^{\mathcal{S}}\boldsymbol{T}_{\mathcal{V}1}\left(\beta_{FoV}, d_s\right) &= \begin{bmatrix} \cos(\beta_{FoV}) & 0 & -\sin(\beta_{FoV}) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\beta_{FoV}) & 0 & \cos(\beta_{FoV}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -d_s \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos(\beta_{FoV}) & 0 & -\sin(\beta_{FoV}) & -d_s\cos(\beta_{FoV}) \\ 0 & 1 & 0 & 0 \\ \sin(\beta_{FoV}) & 0 & \cos(\beta_{FoV}) & -d_s\sin(\beta_{FoV}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{5.73}$$

The complete kinematic chain equals to

$$
\begin{aligned}
&{}^{\mathcal{S}}\boldsymbol{T}_{\mathcal{G}}\left(\beta_{FoV}, d_s, {}^{\mathcal{G}}\boldsymbol{o}, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}},\right) = {}^{\mathcal{S}}\boldsymbol{T}_{\mathcal{V}1}\left(\beta_{FoV}, d_s\right) \cdot {}^{\mathcal{V}1}\boldsymbol{T}_{\mathcal{G}}\left({}^{\mathcal{G}}\boldsymbol{o}, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) \\
&= \begin{bmatrix}
{}^{\mathcal{V}}o_x \cos(\beta_{FoV}) & {}^{\mathcal{V}}o_y \cos(\beta_{FoV}) & -\sin(\beta_{FoV}) & {}^{\mathcal{G}}z \sin(\beta_{FoV}) - \sigma \cos(\beta_{FoV}) \\
-{}^{\mathcal{V}}o_y & {}^{\mathcal{V}}o_x & 0 & {}^{\mathcal{V}}o_y {}^{\mathcal{G}}x - dx {}^{\mathcal{G}}y \\
{}^{\mathcal{V}}o_x \sin(\beta_{FoV}) & {}^{\mathcal{V}}o_y \sin(\beta_{FoV}) & \cos(\beta_{FoV}) & -{}^{\mathcal{G}}z \cos(\beta_{FoV}) - \sigma \sin(\beta_{FoV}) \\
0 & 0 & 0 & 1
\end{bmatrix} \\
&\text{with } \sigma = d_s + {}^{\mathcal{V}}o_x {}^{\mathcal{G}}x + {}^{\mathcal{V}}o_y {}^{\mathcal{G}}y,
\end{aligned}
$$

and the inverse transformation

$$
\begin{aligned}
&{}^{\mathcal{G}}\boldsymbol{T}_{\mathcal{S}}\left(\beta_{FoV}, d_s, {}^{\mathcal{G}}\boldsymbol{o}, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) = \left({}^{\mathcal{S}}\boldsymbol{T}_{\mathcal{G}}\left(\beta_{FoV}, d_s, {}^{\mathcal{G}}\boldsymbol{o}, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)\right)^{-1} \\
&= \begin{bmatrix}
{}^{\mathcal{V}}o_x \cos(\beta_{FoV}) & -{}^{\mathcal{V}}o_y & {}^{\mathcal{V}}o_x \sin(\beta_{FoV}) & d_s {}^{\mathcal{V}}o_x + {}^{\mathcal{G}}x \\
{}^{\mathcal{V}}o_y \cos(\beta_{FoV}) & {}^{\mathcal{V}}o_x & {}^{\mathcal{V}}o_y \sin(\beta_{FoV}) & d_s {}^{\mathcal{V}}o_y + {}^{\mathcal{G}}y \\
-\sin(\beta_{FoV}) & 0 & \cos(\beta_{FoV}) & {}^{\mathcal{G}}z \\
0 & 0 & 0 & 1
\end{bmatrix}.
\end{aligned}
$$

Accordingly, the global coordinates in the sensor frame yields

$$
\begin{aligned}
&\begin{bmatrix} {}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_o \\ 1 \end{bmatrix} = {}^{\mathcal{S}}\boldsymbol{T}_{\mathcal{G}}\left(\beta_{FoV}, d_s, {}^{\mathcal{G}}\boldsymbol{o}, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) \begin{bmatrix} {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} {}^{\mathcal{S}}x_o \\ {}^{\mathcal{S}}y_o \\ {}^{\mathcal{S}}z_o \\ 1 \end{bmatrix} = \begin{bmatrix}
({}^{\mathcal{G}}z - {}^{\mathcal{G}}z) \sin(\beta_{FoV}) - \sigma \cos(\beta_{FoV}) \\
{}^{\mathcal{V}}o_y({}^{\mathcal{G}}x - {}^{\mathcal{G}}x) + {}^{\mathcal{V}}o_x({}^{\mathcal{G}}y - {}^{\mathcal{G}}y) \\
({}^{\mathcal{G}}z - {}^{\mathcal{G}}z) \cos(\beta_{FoV}) - \sigma \sin(\beta_{FoV}) \\
1
\end{bmatrix} \\
&\text{with } \sigma = d_s + {}^{\mathcal{V}}o_x\left({}^{\mathcal{G}}x - {}^{\mathcal{G}}x\right) + {}^{\mathcal{V}}o_y\left({}^{\mathcal{G}}y - {}^{\mathcal{G}}y\right)
\end{aligned}
\tag{5.74}
$$

Finally, the target position in the sensor coordinate frame can be expressed in global coordinates by (5.74) and applied in $L_{c_{in}}\left({}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_o, \alpha_{FoV}, \kappa_G, \kappa_A\right)$ (5.72). Due to its complexity, the resulting equation is not given here. Figure 5.28 is showing the resulting costs in the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane for a quadrotor at position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} = \left[-1, -1, 0\right]^{\mathrm{T}}$ and orientation ${}^{\mathcal{G}}\boldsymbol{o} = \left[0.71, 0.71\right]^{\mathrm{T}} \equiv {}^{\mathcal{V}}\Psi = 45°$. The triangular base form of the cone is oriented as expected from the *UAV* origin in ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} = \left[-1, -1, 0\right]^{\mathrm{T}}$.

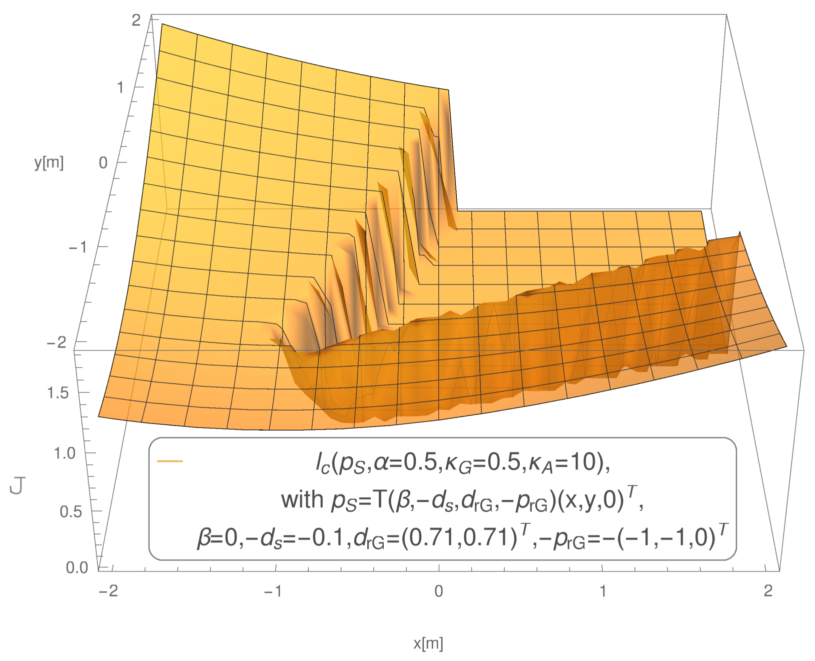


Figure 5.28: Tracking of global position with soft cone constraint

5.6.6 Safety constraints

To validate the derived potential function experimentally, additional safety measures are necessary. The most important safety constraint is treating CA and ensures that a safety distance d_{min} between object and drone is not violated. Second, the maximum distance d_{max} of the sensor has to be considered. This can be accomplished by implementing a cohesion constraint which introduces a repulsive behavior from large distances between object and robot. In general, these safety measures are implemented directly with the inequality constraints in §5.6.2, to ensure the convexity of the derived potential function. However, in this case the safety constraints are treated separately which allows using these also as safety measures for other scenarios. In §5.5 a simple CA constraint has already been introduced, but the workflow presented in §5.6 can be used to derive CA with other convergence properties. This is exemplarily shown for both the cohesion and CA constraint in the presented use case scenario.

Collision avoidance constraint

Considering that ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o$ could also represent an obstacle to avoid, a potential function for obstacle avoidance can be formulated as penalty of a distance below a limit d_{min}:

$$d_{min} \leq \left\| {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right\| . \tag{5.75}$$

The norm can be reformulated by means of quadrature

$$0 \leq c_{inminD} = \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\top} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) - d_{min}^2 \tag{5.76}$$

which is then transformed into a potential function with unit steps

$$\begin{aligned} L_{minD} &= \epsilon\left(c_{inminD}\right) && (5.77)\\ &= \epsilon\left(\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\top} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) - d_{min}^2\right) . && (5.78) \end{aligned}$$

To improve the convergence properties, (5.77) is extended by a quadratic penalty

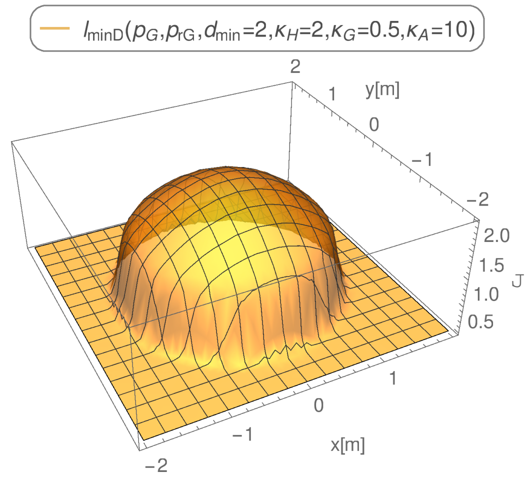
$$\begin{aligned} L_{minD} = {} & \epsilon\left(\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\top} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) - d_{min}^2\right) && (5.79)\\ & \cdot \left(\kappa_H - \kappa_G\left(\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\top} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)\right)\right) \end{aligned}$$

Here, κ_H is defining the maximum height and κ_G the decent of the gradient of the potential function. Finally, (5.79) can be transformed to an analytical function with the help of the sigmoid function (4.138)

$$\begin{aligned} & L_{minD}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}, d_{min}, \kappa_H, \kappa_G, \kappa_A\right) && (5.80)\\ & = sig\left(\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\top} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) - d_{min}^2, \kappa_A\right) && (5.81)\\ & \cdot \left(\kappa_H - \kappa_G\left(\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)^{\top} \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)\right)\right) . \end{aligned}$$

κ_A is describing the quality of the unit step approximation as before. The final result of the potential function is shown in Figure 5.29a which shows a high penalty for the area with distance d_{min} around the origin. The form of the convex top can be adjusted with κ_G. Figure 5.29b shows penalty values $L_{minD} > 0.5$ for an obstacle in position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o = \left[x, y, z\right]^{\mathrm{T}}$ with the *UAV* placed at ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} = \left[-1, -1, 0\right]^{\mathrm{T}}$. As desired, any violation of d_{min} (5.75) is addressed with a high penalty. This reflects the repulsive behavior between robot and object.

(a) Potential function for CA function over ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane



(b) Potential function for CA function in global perspective

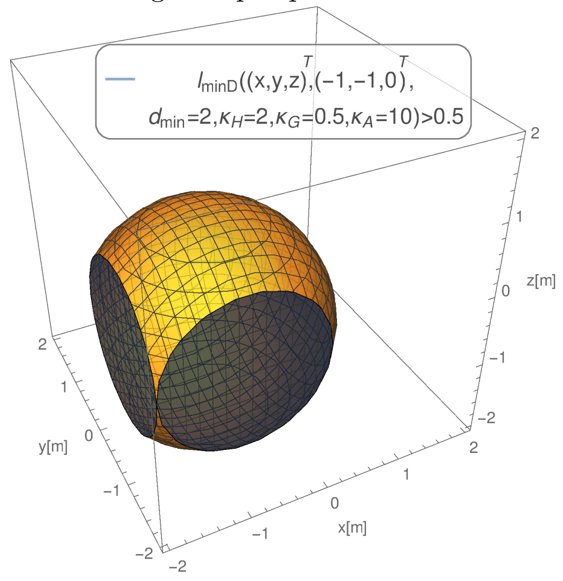


Figure 5.29: Potential function for collision avoidance

Cohesion constraint

Cohesion can be seen as inversion of the *CA* repulsion problem (5.75). This means, distances greater than d_{max} to an object should be avoided. The constraint is therefore formulated as

$$d_{max} \geq \left\| {}^{\mathcal{G}}\vec{\boldsymbol{p}}_o - {}^{\mathcal{G}}\vec{\boldsymbol{p}} \right\|. \tag{5.82}$$

As before, the norm can be expressed with the help of a quadrature

$$0 \leq c_{inmaxD} = d_{max}^2 - \left({}^{\mathcal{G}}\vec{\boldsymbol{p}}_o - {}^{\mathcal{G}}\vec{\boldsymbol{p}}\right)^\top \left({}^{\mathcal{G}}\vec{\boldsymbol{p}}_o - {}^{\mathcal{G}}\vec{\boldsymbol{p}}\right) \tag{5.83}$$

and transformed into a cost function with a unit step function ϵ

$$L_{maxD} = \epsilon\left(c_{inmaxD}\right) = \epsilon\left(d_{max}^2 - \left({}^{\mathcal{G}}\vec{\boldsymbol{p}}_o - {}^{\mathcal{G}}\vec{\boldsymbol{p}}\right)^\top \left({}^{\mathcal{G}}\vec{\boldsymbol{p}}_o - {}^{\mathcal{G}}\vec{\boldsymbol{p}}\right)\right). \tag{5.84}$$

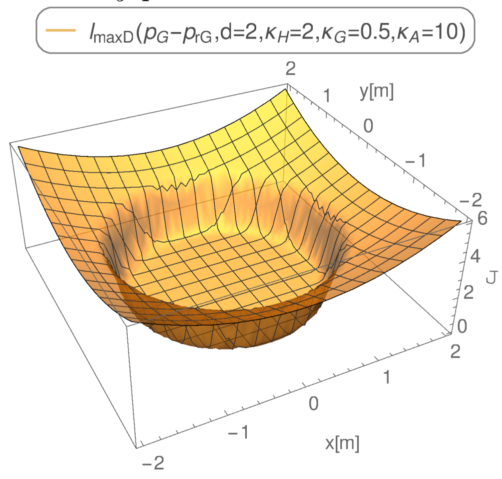
To manipulate the curvature of the given penalty function, κ_H is introduced to define the maximum height. κ_G is describing the ascent of the gradient of the potential function

$$L_{maxD} = \epsilon \left(d_{max^2} - \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right)^\top \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right) \right) \cdot \left(\kappa_H + \kappa_G \left(\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right)^\top \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right) \right) \right). \tag{5.85}$$

The final analytical cost function is gained from an approximation of the unit step ϵ with the sigmoid function (4.138). This results to

$$\begin{aligned} &L_{maxD}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o, d_{max}, \kappa_H, \kappa_G, \kappa_A \right) \\ &= sig\left(d_{max}^2 - \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right)^\top \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right), \kappa_A \right) \ldots \\ &\cdot \left(\kappa_H + \kappa_G \left(\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right)^\top \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right) \right) \right). \end{aligned} \tag{5.86}$$

(a) Potential function for cohesion over ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane



(b) Potential function for cohesion in global perspective

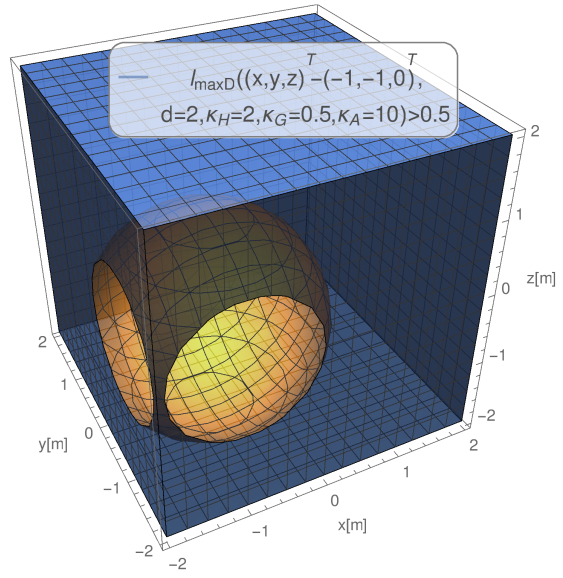


Figure 5.30: Potential function for cohesion

Figure 5.30a is showing the desired attractive behavior.The cohesion constraint leads to a high penalty for distances greater than d_{max} around the origin. Figure 5.29b is showing penalty values $L_{maxD} > 0.5$ for an obstacle in position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o = {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o = \left[x, y, z\right]^{\mathrm{T}}$ with the robot placed at ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} = \left[-1, -1, 0\right]^{\mathrm{T}}$. The figure validates that the whole area of $\left\| {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_o - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}} \right\| \geq d_{max}$ is highly penalized which leads to an attracting behavior between object and robot. With the developed safety constraints (5.80) and (5.86), the cone constraint is validated in the following section.

5.6.7 Experimental validation of task-based MPC

The validation scenario is a visual quadrotor tracking scenario. The underlying task is to keep a target ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_t$ in the sensor perception space, as shown in Figure 5.24. To control this scenario the combination of cone (5.72), *CA* (5.80) and cohesion (5.86) constraints are used in a *NMPC*. In this context the *OCP* considers the coordinate transformation (5.74), quadrotor dynamics (3.27) and the related model parameters (3.29). To validate that the quadrotor keeps the tracked position in its sensor perception space, the scenario is conducted with different initial conditions and disturbance. The disturbance is introduced manually by means of an obstacle with the position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_c$ and a related *CA* constraint (5.79). In order to make the evasive *CA* behavior visible in the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane, the ${}^{\mathcal{G}}z$-axis action of *UAV* is reduced by increasing the input penalty for ${u_{\mathcal{V}1}}_z$. The resulting *OCP* (5.87)-(5.95) does consider the obstacle and target position within the prediction horizon as constant.

$$
\begin{aligned}
\min_{\underline{\boldsymbol{u}}} \; J = & \int_t^{t+T} \underline{\boldsymbol{u}}(\tau)\, \boldsymbol{R}\underline{\boldsymbol{u}}(\tau) && (5.87)\\
& + \kappa_{H0} L_{c_{in}}\left({}^{\mathcal{S}}\underline{\overrightarrow{\boldsymbol{p}}}_t, \alpha_{FoV}, \kappa_{G0}, \kappa_{A0}\right)\\
& + \kappa_{H1} L_{maxD}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_t, {}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}, d_{max}, \kappa_{H1}, \kappa_{G1}, \kappa_{A1}\right)\\
& + \kappa_{H2} L_{minD}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_t, {}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}, d_{min}, \kappa_{H2}, \kappa_{G2}, \kappa_{A2}\right)\\
& + \kappa_{H3} L_{c,minD}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_c, {}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}, d_{c,min}, \kappa_{H3}, \kappa_{G3}, \kappa_{A3}\right) \mathrm{d}\tau\\
s.\ t.\ \; & \dot{\underline{\boldsymbol{x}}} = \boldsymbol{f}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right)\\
& \boldsymbol{0} \leq \boldsymbol{c_{in}} = \left[{u_{\mathcal{V}1}}_x{}^2 - 1, {u_{\mathcal{V}1}}_y{}^2 - 1, {u_{\mathcal{V}1}}_z{}^2 - 1, {u_{\mathcal{V}1}}_\omega{}^2 - 1\right]^{\mathrm{T}} && (5.88)
\end{aligned}
$$

with

$$
\begin{bmatrix} {}^{\mathcal{S}}\underline{\overrightarrow{\boldsymbol{p}}}_t \\ 1 \end{bmatrix} = {}^{\mathcal{S}}\boldsymbol{T}_{\mathcal{G}}\left(\beta_{FoV}, -d_s, {}^{\mathcal{G}}\boldsymbol{o}, -{}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}\right) \begin{bmatrix} {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_t \\ 1 \end{bmatrix}
$$

$$
\boldsymbol{R}_0 = \mathrm{diag}\left(\left[1, 1, 10, 1\right]\right) \qquad (5.89)
$$

$$\varkappa_c : d_s = 0.17, \alpha_{FoV} = 0.5, \beta_{FoV} = 0.5, \tag{5.90}$$
$$\kappa_{H0} = 0.4, \kappa_{G0} = 0.01, \kappa_{A0} = 2.0$$
$$\varkappa_{minD} : d_{min} = 0.7, \kappa_{H1} = 0.4, \tag{5.91}$$
$$\kappa_{H1} = 4.5, \kappa_{G1} = 0.0001, \kappa_{A1} = 3.0$$
$$\varkappa_{maxD} : d_{max} = 2, \kappa_{H2} = 0.4, \tag{5.92}$$
$$\kappa_{H2} = 1.5, \kappa_{G2} = 0.0001, \kappa_{A2} = 3.0$$
$$\varkappa_{c,minD} : d_{c,min} = 1, \kappa_{H3} = 0.6, \tag{5.93}$$
$$\kappa_{H3} = 1.5, \kappa_{G3} = 0.001, \kappa_{A3} = 3.0$$
$$\varkappa_{cgmres} : N = 20, T = 1\,\mathrm{s}, h \;=\; 0.001\,\mathrm{s}, \epsilon = 10^{-8}, \xi = 10 \tag{5.94}$$
$$\Delta t = 0.01\,\mathrm{s}, i_{max} = 30, \upsilon = 2. \tag{5.95}$$

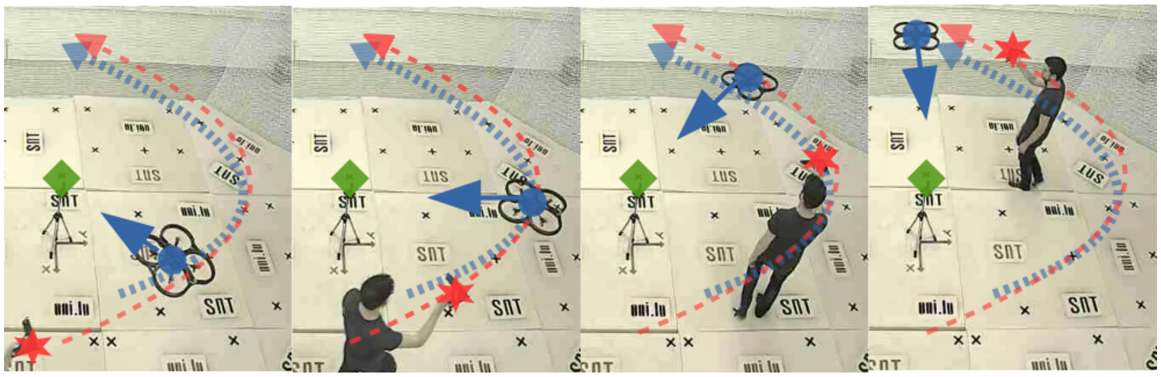


Figure 5.31: Task-based *UAV* control: target tracking while obstacle is avoided

The experimental outline is shown in Figure 5.31. A *OPTITRACK* motion capture system is providing measurements of the *UAV* pose and the obstacle. The constant target position is indicated as green diamond in the center of the pictures. In its initial pose, the *UAV* is not necessarily fulfilling the underlying inequality constraints of the sensor cone constraint (5.72). The activation of the controller therefore initially leads to a convergence towards a compliant pose. As a next step, an obstacle (red star) is introduced manually by means of the motion capture system. The *UAV* (blue circle) is evading the approaching obstacle by moving in the opposite direction. This evasive maneuver shows the form of an arc due to the active target tracking constraints. Manually following the *UAV* with the obstacle leads to a circular trajectory around the tracked target point ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_t$. The corresponding *UAV* orientation is indicated by a blue arrow.

To cope different initial conditions and obstacle patterns, a set of 10 experiments is conducted. Figure 5.32 is showing the resulting ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-trajectory of the UAV (blue line), obstacle (dashed red line) and the fixed target position (green diamond). To visualize the system at different time instances, the position of the obstacle (red star) and the pose of the UAV (blue circle with arrow) is marked every $\Delta t = 4.2\,\mathrm{s}$. The distances between UAV and target is signalized as light grey line at each of these time instances. Accordingly, the distance between the UAV and the obstacle is indicated as light red line. The resulting circular pattern validates the target tracking during the evasion maneuver. Furthermore, the direction of the UAV is pointing to the center which is indicating the tracking of the target. In this context, in plot 7 center left and plot 9 center up the drone orientations which are not pointing towards the target are representing initial conditions. These initial poses have been chosen arbitrarily within the spacial limitations of the laboratory. Its exact values can be exerted from the UAV pose plots in Figure 5.33. In this initial phase, a typical increase in the altitude (${}^{\mathcal{G}}z$) can be observed. This is caused by the inclination angle β_{FoV} of the sensor cone constraint. The steps in the ${}^{\mathcal{V}}\Psi$ values in Figure 5.33 are based on the limited ${}^{\mathcal{V}}\Psi$-angle interval. The sinusoidal trajectories in ${}^{\mathcal{G}}x$ and ${}^{\mathcal{G}}y$ evidence the circular evasion maneuver of UAV. Figure 5.33 is showing measurements of the corresponding obstacle position. Also here the sinusoidal trajectory is visible in order to provoke the UAV evasion maneuver. The steps in the obstacle position at the beginning are caused by the entering into the detection zone of the motion capture system.

To analyze the influence of the obstacle CA constraint L_{minD} (5.80), Figure 5.35 shows the distance between the UAV and the obstacle. The single plots show how the obstacle is moved close to $d_{c,min}$ to provoke an evasion maneuver of UAV. The measured minimal distances are given in Table 5.1. As the CA design is based on soft constraints, violations are feasible. For all 10 experiments, the highest violation of the obstacle distance d_c appears in run 5 with $\min(d_c) = 0.601\,\mathrm{m}$. The violation depends on the UAV and obstacle speed as well as the cost gradient design of the CA constraint. For the considered UAV and obstacle speeds, the cost gradient is chosen less steep (5.93) in order to show a smooth repulsive behavior while enforcing the desired evasion. In reverse conclusion, higher violations are accepted. Its repulsive behavior can be observed as oscillation around the constraint border in Figure 5.35 plots 6, 7 and 9.

The influence of the target CA and cohesion constraints can be evaluated using the Euclidean distance of UAV to the target as shown in Figure 5.36. Both constraints

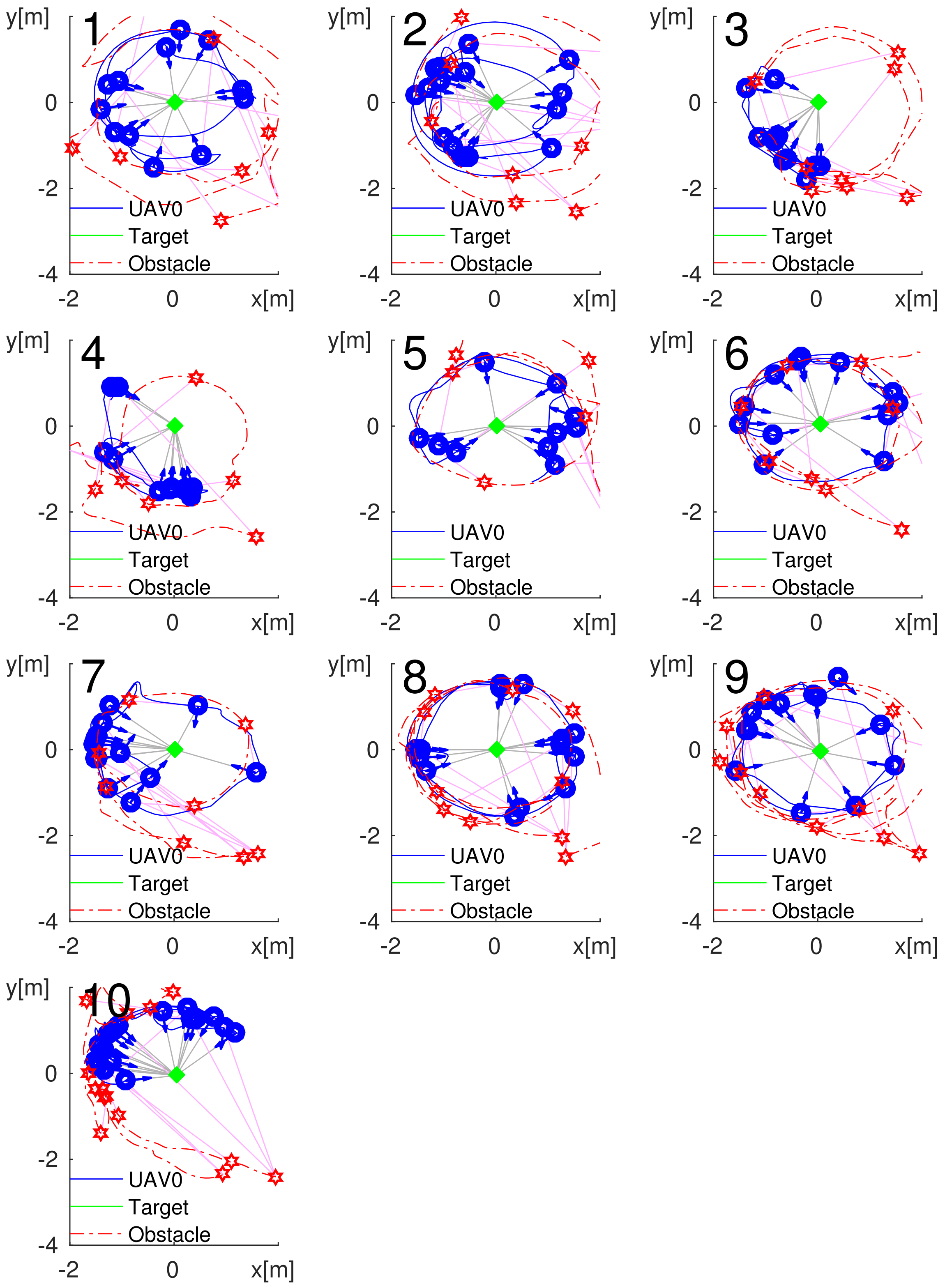


Figure 5.32: Task-based *UAV* control: *UAV* ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-trajectory with pose markers every $\Delta t = 4.2\,\mathrm{s}$ showing circular evasion maneuver pattern

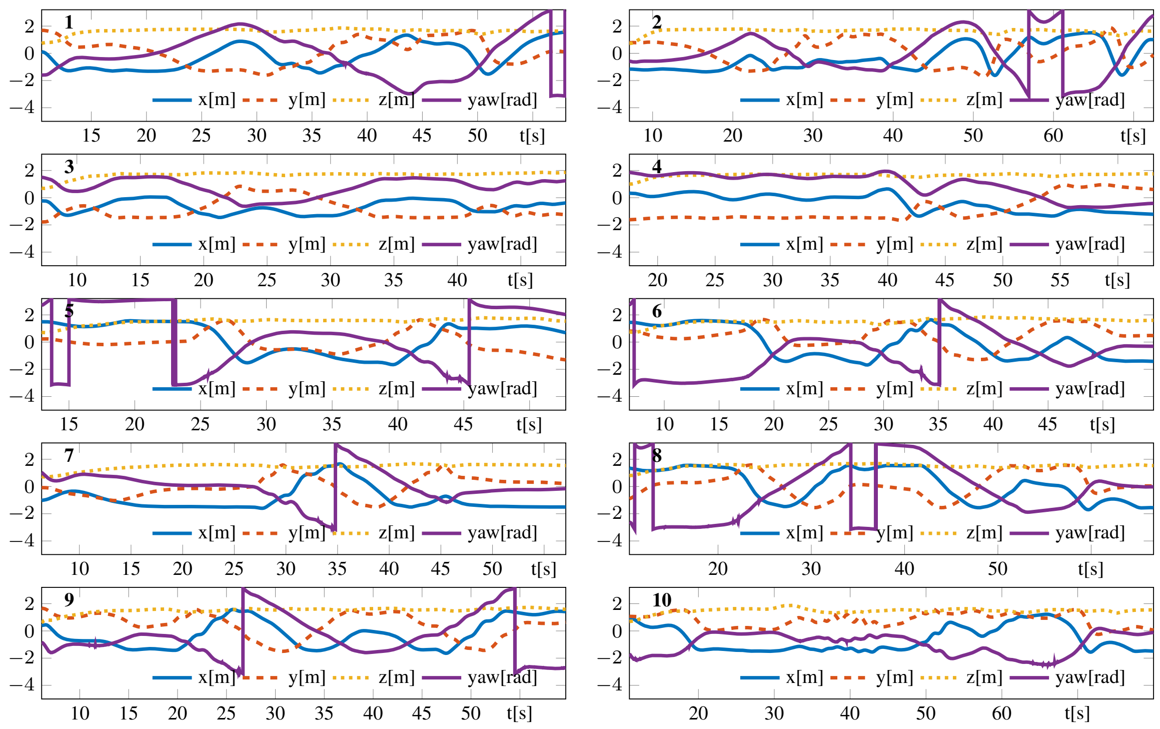


Figure 5.33: Task-based UAV control: UAV pose trajectory with steps in the ${}^{\mathcal{V}}\Psi$-trajectory due to the limited yaw angle interval

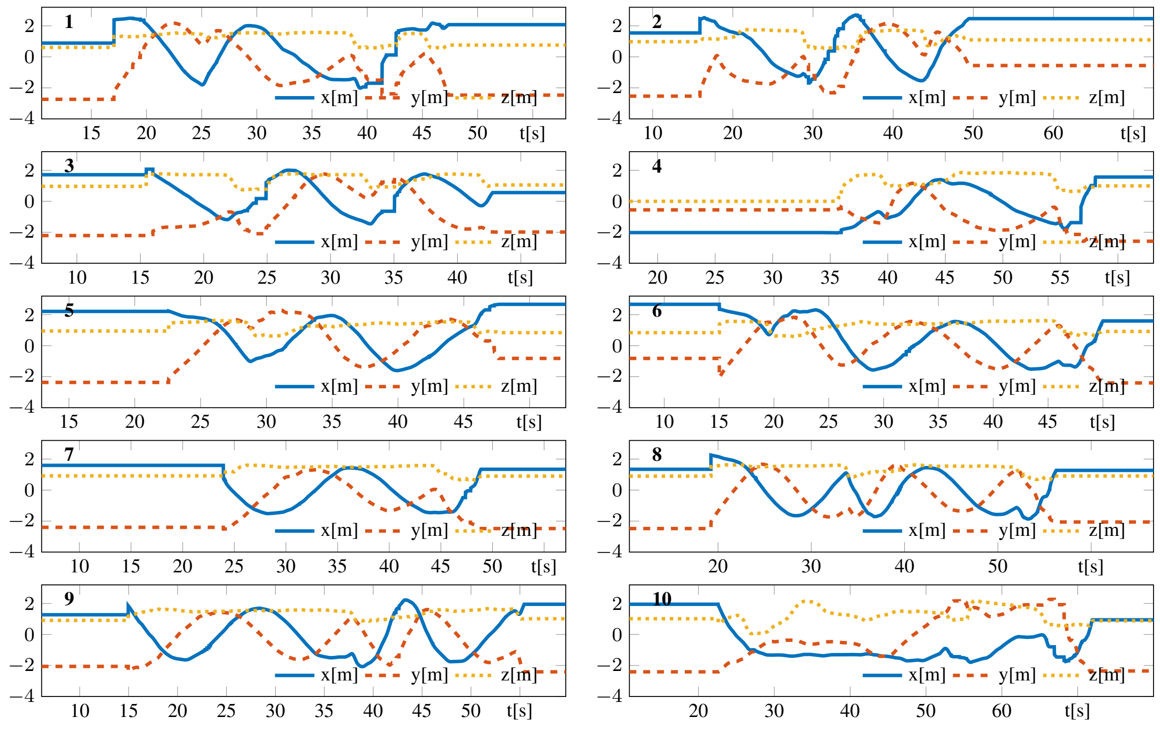


Figure 5.34: Task-based UAV control: manually introduced obstacle position is showing circular pattern in order to follow UAV on its evasion trajectory. The steps are caused by the obstacle entering the field of perception of the motion caption system.

restrict the distance to $d_{min} = 0.7\,\mathrm{m} \leq d \leq 2.0\,\mathrm{m} = d_{max}$. The measured distance d lies in the interval $0.941\,\mathrm{m} \leq d \leq 1.966\,\mathrm{m}$ and therefore complies to the target distance constraints for all 10 experiments. The peak distance values are given in Table 5.1. The minimum tracking distance $\min(d) = 0.941\,\mathrm{m}$ appears in run 2 and its maximum value $\max(d) = 1.966\,\mathrm{m}$ in run 10.

In addition to the distance, also the tracking angle is required in order to evaluate the tracking performance the given sensor cone (5.72). Hence, the absolute tracking angle α_t is defined between the UAV_1 sensor orientation and the distance vector

$$\alpha_t = \| \arccos(\frac{{}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_t \cdot (1,0,0)^\top}{|{}^{\mathcal{S}}\overrightarrow{\boldsymbol{p}}_t|}) \|. \tag{5.96}$$

The result is shown in Figure 5.37. The sensor constraint (5.72) is restricting the absolute tracking angle to $\alpha_t \leq \alpha_{tmax} = 0.5\,\mathrm{rad}$. For the set of experiments, the initial conditions do typically not satisfy this constraint due to a low initial UAV altitude ${}^{\mathcal{G}}z$ and an incompliant orientation ${}^{\mathcal{V}}\Psi$. This is directly visible in the controller counter action as shown in Figure 5.38. In the initial phase, run 1-5 show a direct reaction in the altitude by $|u_{\mathcal{V}1_z}| >> 0$, while a significant adjustment of the orientation by $|u_{\mathcal{V}1_\omega}| >> 0$ is dominating in run 5-10. The resulting convergence towards the constraint compliance $\alpha_t \rightarrow \alpha_{tmax}$ can be seen in all plots of Figure 5.36. This is followed by a period of low action, as all constraints are satisfied and the obstacle is not considered yet. With the approaching obstacle $d_c \rightarrow d_{c,min}$ in Figure 5.35, the control action in Figure 5.38 is increased due to the evasion maneuver. As a result, also the tracking angle α_t in Figure 5.37 is disturbed. To measure the constraint violation, the maximal absolute tracking angle in the nominal state $\max(\alpha_{tns})$ is given in Table 5.1. For this purpose, $\max(\alpha_{tns})$ takes into consideration the absolute tracking angle peak values after the initial convergence phase. $\alpha_t \leq \alpha_{tmax} = 0.5\,\mathrm{rad}$ holds after the initial phase for all the experiments. The measured maximum appears in run 2 with $\max(\alpha_{tns}) = 0.438\,\mathrm{rad}$.

The experimental results validate the desired behavior of task-based tracking also under the influence of disturbance introduced as obstacle. The sensor field of view limitations are respected and collisions are avoided. Furthermore, the set of experiments shows the robust convergence towards a compliant state under differing initial conditions and constraint violations.

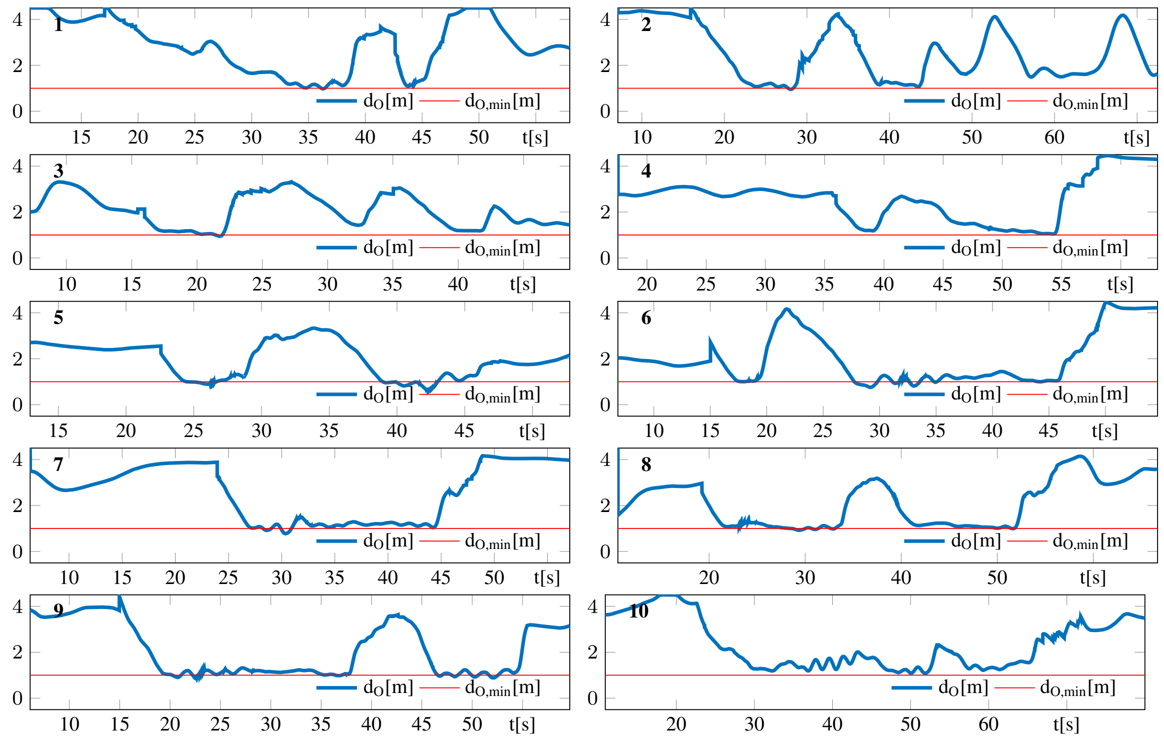


Figure 5.35: Task-based *UAV* control: *UAV* distance to obstacle d_o stays above the defined minimum distance of $d_{c,min} = 1.0\,\mathrm{m}$

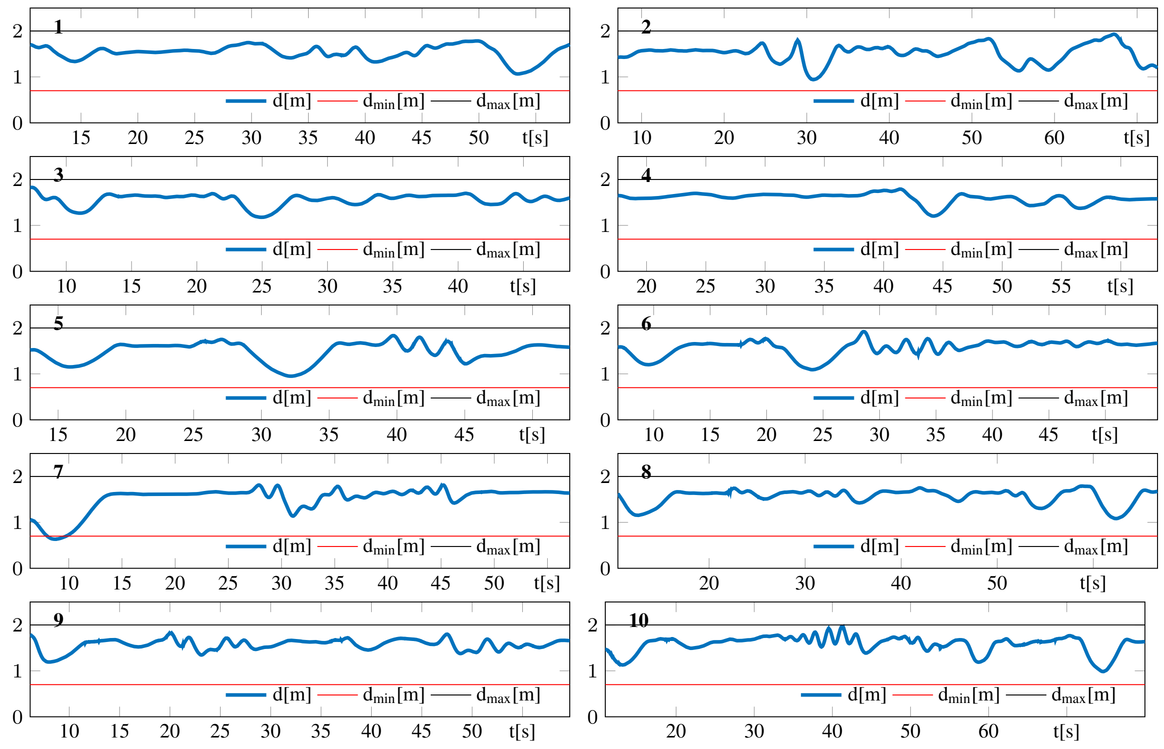


Figure 5.36: Task-based *UAV* control: *UAV* distance to target d stays within the defined minimum distance of $d_{min} = 0.7\,\mathrm{m}$ and maximum distance of $d_{max} = 2.0\,\mathrm{m}$

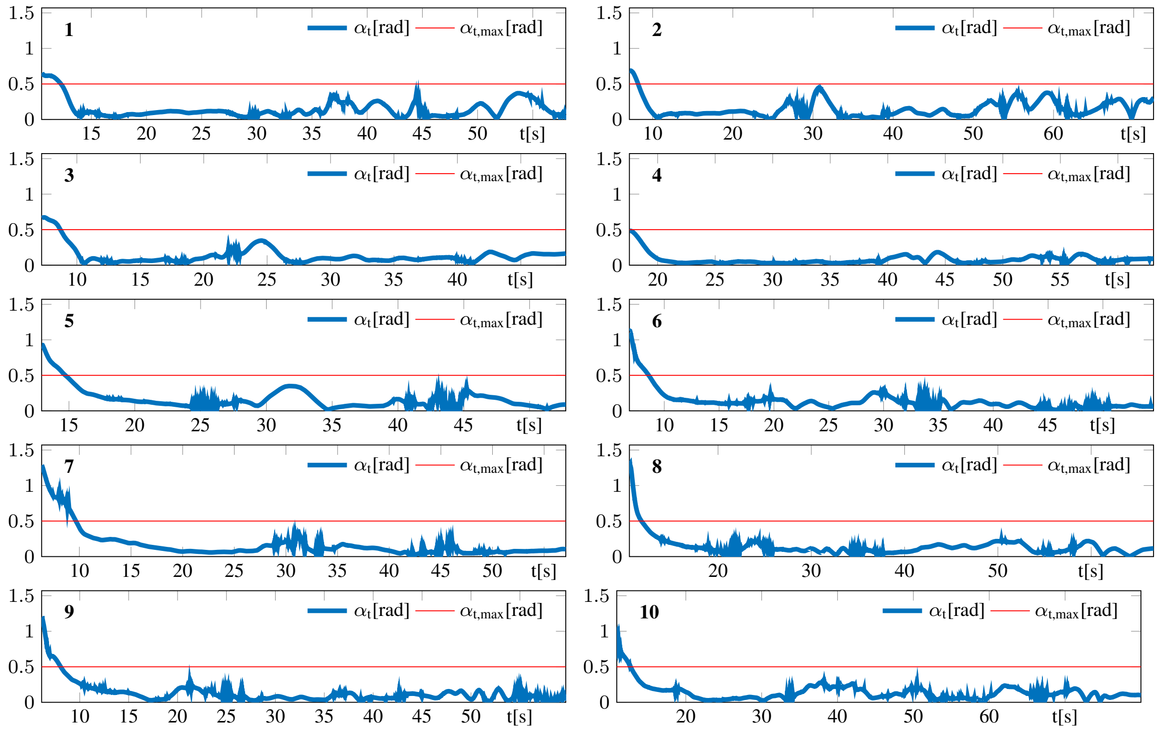


Figure 5.37: Task-based *UAV* control: *UAV* absolute tracking angle α_t stays within the defined maximum of $\alpha_{tmax} = 0.5$ rad of the sensor cone constraint

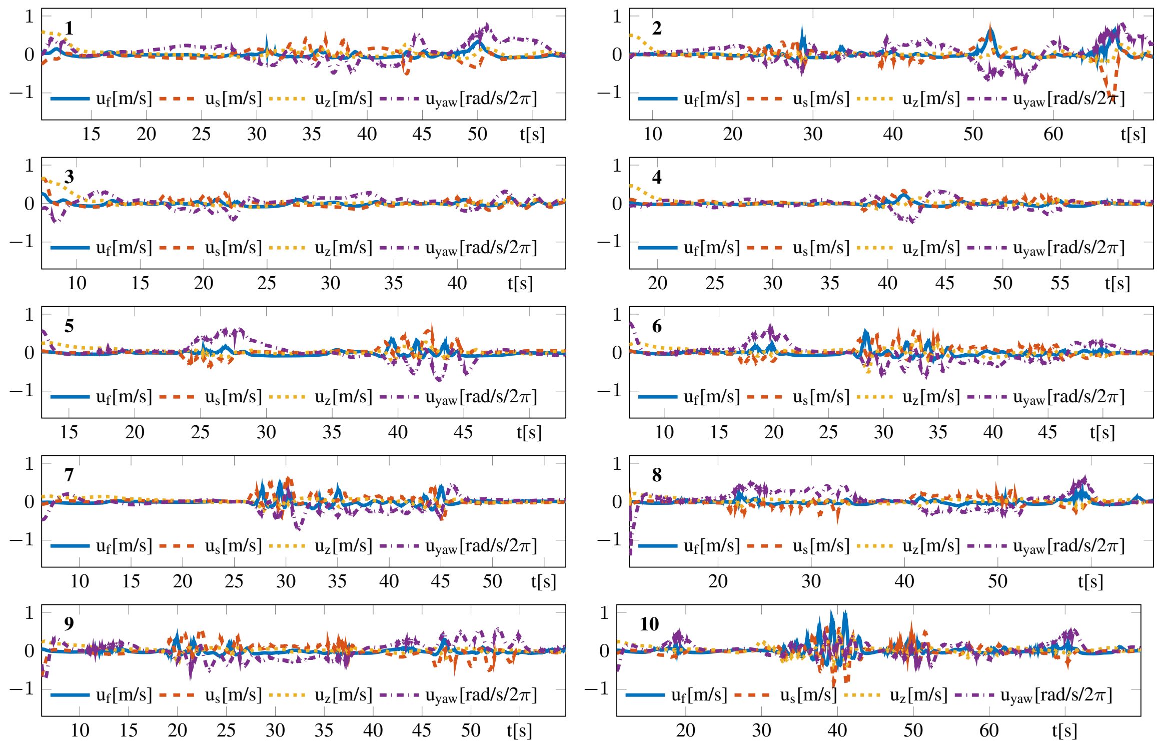


Figure 5.38: Task-based *UAV* control: *UAV* actuation

run	1	2	3	4	5
$\min(d_c)$[m]	0.976	0.950	0.945	1.038	**0.601**
$\min(d)$[m]	1.064	**0.941**	1.176	1.203	0.950
$\max(d)$[m]	1.785	1.931	1.827	1.791	1.834
$\max(\alpha_{tns})$[rad]	0.371	**0.438**	0.345	0.183	0.350
run	6	7	8	9	10
$\min(d_c)$[m]	0.754	0.773	0.915	0.879	1.083
$\min(d)$[m]	1.091	0.633	1.082	1.191	0.984
$\min(d)$[m]	1.922	1.815	1.791	1.823	**1.966**
$\max(\alpha_{tns})$[rad]	0.260	0.307	0.220	0.225	0.271

Table 5.1: Task-based *UAV* control: Peak distance values for the set of 10 conducted experiments

5.7 Conclusion

The chapter has presented an *NMPC* control approach for commercial low-cost multi-rotor systems. The proposed control strategy provides very low computation times combined with the ability to handle constraints. As a first step, the controllability has been analytically validated in section §5.2. In chapter §5.3, a state tracking *NMPC* implementation for *UAV*s has been presented using *CMSC/GMRES*. The proposed control has subsequently been validated on real *AR.Drone 2.0* and *DJI M100* systems. In this context, the *NMPC* parameterization has been discussed and evaluated. In addition, the experimental results have been referenced with simulation results in order to asses the quality of the identified *UAV* models (3.49) and (3.55). An average computation time of $\bar{t_c} = 0.3700\,\text{ms}$ (*AR.Drone 2.0*) and $\bar{t_c} = 0.4295\,\text{ms}$ (*DJI M100*) stated the efficiency of the proposed *CMSC/GMRES* approach. With a control update interval time of $\Delta t = 10\,\text{ms}$, this also validated the real-time applicability.

One problem of standard *NMPC* is the constant offset in pose tracking scenarios of constantly moving targets. An offset-free *MPC* approach by using a *TPC* has been discussed in section §5.4. To address this problem, the closed-loop system has been considered as *PT1*. By means of Laplace transformation it has been shown, that this modeling approach for the closed loop system is justified for the considered problem. To reduce the tracking error, a *TPC* is added as outer loop to the *MPC* controlled system. The *TPC* is providing the *MPC* with targets that are virtually set further away than the actually given. This translation is related via a *PI*-controller with the tracking error. The stability of the proposed algorithm has been validated experimentally and stated its efficiency with a maximum computation time of $max(t_c) \approx 1\,\text{ms}$. The position control has also stabilized the *AR.Drone 2.0* under more complex disturbances (involving ${}^{\mathcal{V}}\Psi$), with the remark that the input limitation constraint handling of $glsu_{\mathcal{V}1_\omega}$ has to be adjusted accordingly. The presented *TPC* requires low computational effort in comparison to other approaches that include disturbance model estimation. Furthermore, the parameter tuning is intuitive, does not require changes in the *MPC* and does not affect the constraint handling of the *MPC*.

Another problem of *UAV* pose tracking in a real environment are obstacles. To be able to avoid obstacles, §5.5 demonstrated how a minimum distance inequality constraints is transformed into a potential function using the saturation function approach from §4.5.3. The corresponding measurements with the real *AR.Drone 2.0* validated this approach [DKMV16b]. Using the reduced memory model, the computational load on a standard computer (see §A.1) was $\frac{max(t_c)}{\Delta t} \approx \frac{1.6\,\mathrm{ms}}{100\,\mathrm{ms}} \approx 1.6\%$.

CA is further generalized to task-based control in §5.6. In this section, a workflow was contributed in order to derive potential functions. This allows the incorperation of control goals, sensor constraints and environmental constraints into the *NMPC*. An *AR.Drone 2.0* quadrotor has served as use case which has been tracking an object with an attached sensor. The first step in the workflow is to formulate the sensor perception space within a sensor coordinate frame by means of inequality constraints. For the use case, the cone shape of the sensor perception space has been described by a combination of two inequality constraints. Following the saturation function approach from §4.5.3, the second step is transforming these constraints into a potential function with the help of step functions. This transformation facilitates the analytical and graphical validation of the resulting cost function without having to deal with the increased complexity of sigmoids. To improve the solvability of the problem for gradient-based solvers, the next step has been the introduction of a gradient in the incompliant regions. The gradient is designed to point away from the compliant region. Finally, the saturation function is used to approximate the step functions which leads to a continuous gradient around the constraint borders. As a result the potential function becomes analytical and therefore solvable by standard real-time *MPC* solvers. As safety measure for the experimental validation, a *CA* and cohesion constraint have been developed with the same workflow. The tracking by means of the sensor constraint has been tested experimentally in simulation and a real-world implementation. In the presented scenarios an *AR.Drone 2.0* is tracking a fixed target using the developed constraints while disturbance is introduced in form of an obstacle. The results show the desired collision evasion maneuver while maintaining sensor tracking for different initial conditions.

A further development will be the analysis of the energy efficiency of the proposed sensor constraint, in comparison to a simple orientation tracking. In this context, a statistical analysis of large numbers of real-world experiments would be of interest.

Chapter 6

Aerial manipulation

The success of *UAV*s in agriculture, security and industry is based on their extended operational space. However, *UAV*s are currently mostly used in sensing and transportation scenarios. The idea of aerial manipulation is thus to extend the usage of *UAV*s by introducing ways to interact with their environment. The importance of this development is stated by dedicated research projects, as for example the European Aerial Robotics Cooperative Assembly System project (*ARCAS*). Examples from the fast-growing field of aerial manipulation applications are the contact inspection of bridges [JCBHO15], canopy sampling of the environment [KSX15], opening of valves [KOO14], and positioning of assembly parts in factories [MLS15].

To fulfill such tasks, a manipulator is attached to the *UAV* which results in a Manipulating Unmanned Aerial Vehicle (*MUAV*). A flexible approach fitting many manipulation scenarios is the use of a robotic arm as manipulator. Typically, for small *UAV* systems (quadrotor, hexarotor, etc.) all rotors are aligned in one plane. Without external influences, such *UAV*s are usually stabilized around their hovering condition with zero pitch and roll. An attached robotic arm has the advantage to extend this operational space, as the orientation of the arm end effector is not limited to the hover condition and thus is able to manipulate objects with different orientations.

Despite the advantages of *MUAV* systems, the number of commercially available solutions is very limited. This is caused by the challenges of precise localization, controlling the complex dynamics of the combined *MUAV*, and the limited payload of the *UAV* which also limits the manipulator size and payload. These challenges are also reflected in the research domain, as the number of publications using real *MUAV* experiments is very limited. Furthermore, these experiments are typically executed with proprietary *MUAV* designs where dynamic properties and internal controller are fully known. However, the difficulty to reproduce proprietary solutions limits the access to this field of application for research and industry.

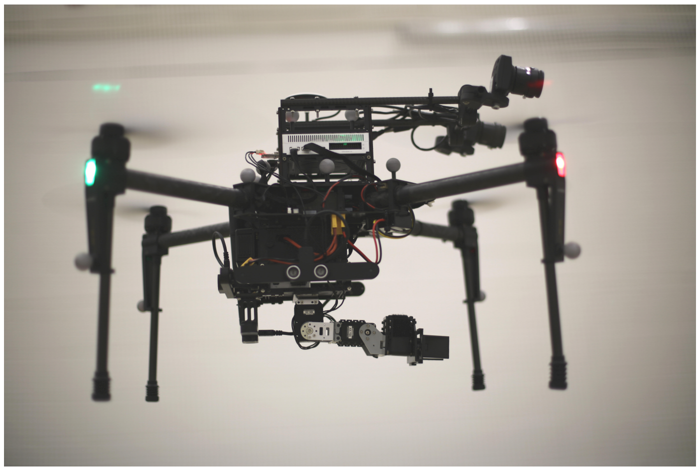


Figure 6.1: Aerial Manipulator consisting of *DJI M100* and two-joint robotic arm (the stereo camera on top is not part of the experimental setup)

For these reasons, §6 is discussing the modeling and control of an under-actuated *MUAV* consisting of a commercially available robotic arm attached to a commercial *UAV*. The *MUAV* system considered in this work is shown in Figure 6.1 consisting of a *DJI M100* quadrotor with an attached 2-joint robotic arm. The utilization of commercially available components hereby is minimizing the development effort and promoting science as well as application development for *MUAV*s. The difficulty of using commercially available products is that very limited information about internal control algorithms and physical model properties is available. As use case for this thesis, a two-joint robotic arm is chosen for its low weight. This results in higher payload as well as longer flight autonomy. The disadvantage of this under-actuation is that *UAV* and manipulator have to be controlled dependently in order to maximize the subspace of reachable poses.

To address commercial *MUAV* control, this chapter is structured as follows. First, the related work is discussed in §6.1. As next step, §6.2 is contributing the forward kinematics (*FK*) of the *MUAV* shown in Figure 6.1. Based on the *FK*, a novel *NMPC* for the *MUAV* end effector pose is developed in §6.3. This approach is based on a separation of the end effector position and orientation tracking in the *NMPC* cost-function. Furthermore, the solution for the Inverse Kinematic (*IK*) problem of the end effector pitch angle is presented. In this context, the *IK* problem is to determine the necessary manipulator joint angles for a given end effector pitch orientation. The combination of the proposed *NMPC* and *IK* control results in a performant, real-time capable *MUAV* end effector control with a large reachable subset of poses. In §6.4, this contribution is extended by a real-world experimental validation of the proposed control approach. Furthermore, its real-time applicability in manipulation tasks is demonstrated in a real *MUAV* bottle grasping scenario.

6.1 Related work in aerial manipulation control

First considering only robotic arms, the literature for control of robotic arms is well established as a result of an extensive industrial use of multi-joint robotic arms. For this reason, a detailed literature review in this perspective is omitted. For further information, a very comprehensive study on robotics and robotic manipulators has been published in [WS16]. The book features fundamentals in robot design, analysis, control and simulation including manipulator kinematics, differential kinematics and dynamics. It also covers *IK* control which is partially applied in this work.

Instead, the focus of this section is the control of full *MUAV* scenarios. Most manipulation tasks require the tracking of the manipulator's end effector pose. The pose of the object to grasp is hereby typically defined in a global coordinate system. In literature, two major approaches can be distinguished. The first approach is to control *UAV* and arm separately. A global convergence is however reached, by injecting their mutually influenced error into the individual manipulator and *UAV* control algorithms. This leads typically to hierarchical control concepts. Examples for such separated approaches are given in [KQGD+16] and [ACMP13]. These publications present hierarchical motion controllers for *MUAV*s. To derive the desired end-effector waypoints for the quadrotors position and orientation control, the manipulator's inverse kinematics (*IK*) are used. In parallel, the manipulator joints are determined by the given desired orientation. Both publications provide a simulative validation scenario. The stability properties of such hierarchical control approaches based on *PID*-controllers is studied in [OKPO13]. Based on the principle of Lyapunov-stability, [KAOMV14] is deriving a hierarchical adaptive *UAV* position control. For this purpose, the horizontal movement and altitude are treated separately. The influence of the manipulator's momentum and force on the *UAV* is thereby rejected by the adaptive control design. In [BZM+14], a two-joint manipulator *MUAV* is controlled with feedback linearization and compensation of the manipulator's influence. It refers to this method as consecutive compensator which is based on the real-time computation of the inertia tensor, center of mass position and reactive torque. The results have been validated in simulation. These types of approaches are typically computationally lightweight, but are less performant than the second type of *MUAV* control.

This second and more intuitive approach is to develop a complete system model, which reflects the combined movement of *UAV* with attached robotic arm and develop a control solution for the combined system dynamics. In contrast to the previous approach, the usage of the full *MUAV* model promises to achieve a better closed-loop *MUAV* performance. One example of a full dynamic model of the combined two-joint *MUAV* system is derived in [KCK13] using the Lagrange-D'Alembert methodology. Subsequently, the paper is presenting a sliding mode controller with Lyapunov stability proof. The developed approach is validated experimentally by grasping a wooden block with a real *MUAV* and inserting it into a box. The combined model of another two-joint *MUAV* system is presented in [ACLV16]. In this context, the full kinematic chain is determined using dual quaternions and the related dynamics according to the Newton-Euler formalism. The dual quaternions are thereby used to express the translational and rotational transformation of coordinate frames at once. The controller is based on a dual quaternion feedback-linearization technique and validated experimentally. A Lyapunov based control-approach for a three-joint *MUAV* is presented in [ÁMMGC$^+$14], providing simulative results. In [ASO15], a controller for a multi-joint *MUAV* is deriving by energy shaping. The paper is proposing potential and kinetic energy shaping to overcome the issue of unstable unactuated coordinates in underactuated *MUAV* systems. The stability properties of the resulting Interconnection and Damping Assignment Passivity-Based Control (IDA-PBC) are derived without and under consideration of external forces. Furthermore, an experimental validation is provided. [BZM$^+$14] is extended by [BKMZ16], presenting a finite-time *MUAV* control with feedback linearization in order to control a two-joint manipulator *MUAV*. The publication is testing the control approach in simulation. A further simulative validation of a *MUAV* controller is given by [ZZZ16]. Furthermore, it contributes a geometric modeling and control approach for an n-joint *MUAV* based on Lie-algebra. The kinematic chain is thereby modeled by means of the exponential map according to Lie-theory.

In the context of the *ARCAS* project, a behavioral approach using the combined differential kinematics of the *MUAV* is given in [BGP$^+$15], [BGP$^+$17]. The idea is here to define a task variable for different *MUAV* elementary behaviors, as e.g. *UAV* tracking, *CA*, etc. Derivatives of these task variables are directly influencing the velocity reference of the *MUAV*. Accordingly, this approach is closely related to an optimal control approach with a task-related cost function. The advantage of the presented approach is the ability to change tasks of the *MUAV*. The publication [MPT+16] is further extending [BGP+15] by a multi *MUAV* coordination strategy. As shown in §5.6, another approach to describe such task-based controllers is *MPC*. In *MPC*, the task is directly formulated with cost functions and constraints in the inherent *OCP*. The result is a simple task definition and high control performance under the drawback of a higher computational burden. For this reason, previous work [DKMV17] presented the framework *DENMPC* in order to provide a computationally efficient *NMPC* that allows switching of tasks and constraints. Detailed information of *DENMPC* is given in §7. However, due to the complex and fast non-linear dynamics of *MUAV*s, real-time *MPC* for such systems is still challenging. A first approach towards this goal has been presented in [GK15] by showing a model predictive trajectory generator for a two-joint *MUAV*. For this purpose, the *MUAV* dynamics are derived and an input constrained *OCP* is formulated. To solve this *OCP*, first, a standard Gauss-Newton solver is used. Second, a linearization technique for optimal control with Lie-Group structure is presented, which subsequently allows the solution with *SQP* and similar. The approach is validated experimentally in a bottle-grasping scenario. For this purpose, the optimal controls are computed as a position and velocity reference for the feedback-linearization control of the *UAV* and the *PID* servo controllers. The authors mentioned that future development will address the real-time applicability of the optimal control approach. Another optimal control approach for a *MUAV* with docking ability is presented in [DABS14]. For this purpose, the model translating between the *MUAV* flight behavior and the behavior during physical interaction is described by piecewise affine systems. For each of these system states, a model for the lateral and longitudinal motion is designed and hybrid predictive controllers are computed.

Besides the computational burden, the major difficulty of a combined *UAV*/manipulator control is the requirement of a combined system model. Most of the previously described approaches are therefore deriving the dynamics based on force and torque acting on the *UAV*. However, for safety reasons commercial *UAV*s typically have an internal controller which stabilizes the *UAV* and tracks velocity commands. These issues are addressed within the context of this work, by presenting a *MUAV NMPC* which is considering the full *MUAV* kinematics based on the *HMDV* (3.55).

Besides the pose tracking problem, force- and torque control and visual servoing shall be mentioned here, but are not in the scope of this thesis. Visual servoing is referring to the usage of onboard image sensors to control the relative pose to the object to grasp. A recent example of visual servoing for aerial manipulation by [SKK17b], is presenting a stochastic *MPC*-based visual servoing algorithm to grasp a cylinder with a *MUAV*. The approach has been validated experimentally. A visual object tracking approach in an aerial manipulation scenario is given in [BCL15]. The control is hereby separately composed of the influence on the end effector position, orientation, center of gravity and a safety constraint to comply with the joint limits. The resulting visual servoing controller is validated in simulation. [DO14a] is simulating the *UAV* with a gantry to consider a visual servoing control under disturbance. In [DO14b], a visual servoing control of a 6-*DOF* manipulator attached to a *UAV* is presented. The servos are thereby controlled according to a camera which is placed on the manipulator end effector. The visual servoing algorithm is experimentally validated in a manipulator stand-alone scenario.

6.2 Aerial manipulator kinematics

Robotic arms are versatile manipulators. They typically consist of a manipulator base and an end effector which are connected via adjoint links. To achieve *MUAV* capabilities, such a robotic arm is attached to the bottom of a *DJI M100* quadrotor. Without loss of generality, this robotic arm consists of multiple neighboring links, which are each interconnected by one joint. Within this work, the robotic arm displayed in Figure 6.2 is utilized which features two revolute joints with parallel rotation axes and a gripper as end effector. This arm configuration has been chosen as compromise between costs, weight and reachable pose set in combination with the *UAV*. The manipulator hereby consists of fully commercially available parts including servo "Dynamixel AX18-A" which has been used for the gripper and joint 2, as well as the "Dynamixel MX28-T" servo for joint 1. The arm links consist of commercially available aluminum and bioloid frames. The servos are interfaced with the *DJI M100* by a "USB2Dynamixel". The angle position can be directly controlled in servo mode or the joint angle velocity in wheel mode. For this setup, the joints are controlled in position control mode ($\theta_1(t)$ for joint 1 and $\theta_2(t)$ for joint 2), as the natural servo interface does not provide angle constraint compliance in wheel mode.

The robotic arm shown in Figure 6.2 is attached to the *DJI M100* which leads to the configuration, shown in Figure 6.3. For the given *MUAV*, the pose of the end effector with attached coordinate system $\mathcal{E}$ is depending on the joint angles $\theta_1(t)$, $\theta_2(t)$ as well as the *UAV*'s orientation ${}^{\mathcal{G}}\boldsymbol{\rho}$ and position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}$. The description of this dependency is generally known as forward or direct kinematic problem under the assumption of rigid arm links and ideal surfaces [WS16, p.19]. The relation to

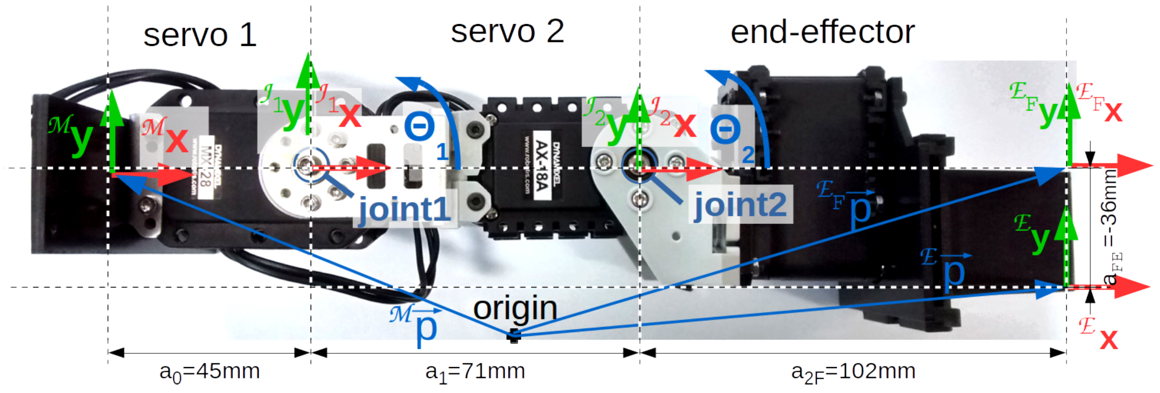


Figure 6.2: Manipulator dimension and coordinate frames for front tracking

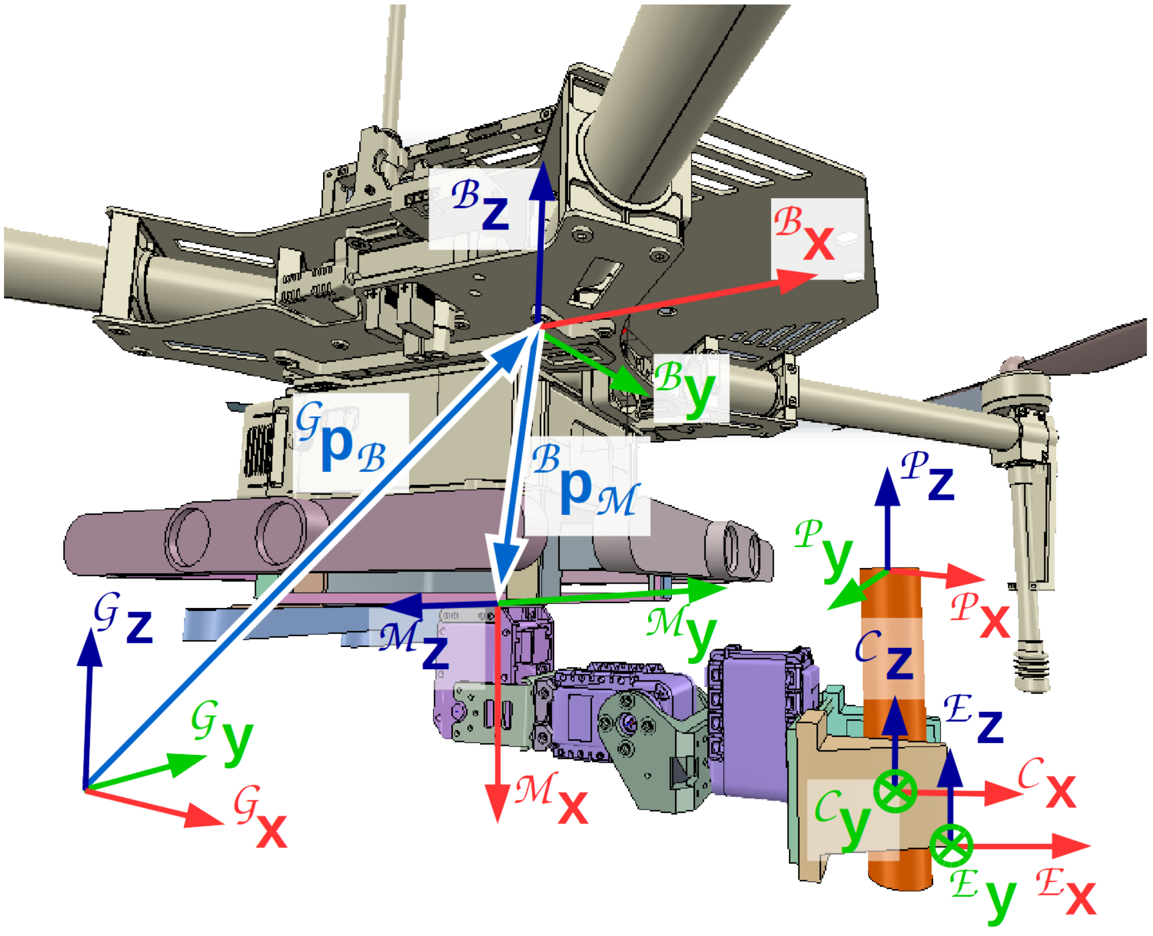


Figure 6.3: *MUAV* with grasped object (orange cylinder): coordinate frames

transform the end effector coordinates into $\mathcal{G}$ can directly expressed by the kinematic chain shown in Figure 6.4.

$$\begin{tikzcd}
\mathcal{G} \arrow[r, bend left, "{}^{\mathcal{V}}\boldsymbol{T}_{\mathcal{G}}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{V}}\right)"] & \mathcal{V} \arrow[l, bend left, "{}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{G}}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{V}}\right)^{-1}"] \arrow[r, bend left, "{}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{V}}\left({}^{\mathcal{G}}\boldsymbol{\rho}\right)"] & \mathcal{B} \arrow[l, bend left, "\left({}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{V}}\left({}^{\mathcal{G}}\boldsymbol{\rho}\right)\right)^{-1}"] \arrow[r, bend left, "{}^{\mathcal{M}}\boldsymbol{T}_{\mathcal{B}}"] & \mathcal{M} \arrow[l, bend left, "\left({}^{\mathcal{M}}\boldsymbol{T}_{\mathcal{B}}\right)^{-1}"] \arrow[dlll, dashed, bend left, "{}^{\mathcal{J}1}\boldsymbol{T}_{\mathcal{M}}\left(\theta_1\right)"] \\
\mathcal{J}1 \arrow[urrr, dashed, bend right, "\left({}^{\mathcal{J}1}\boldsymbol{T}_{\mathcal{M}}\left(\theta_1\right)\right)^{-1}"'] \arrow[r, bend left, "{}^{\mathcal{J}2}\boldsymbol{T}_{\mathcal{J}1}\left(\theta_2\right)"] & \mathcal{J}2 \arrow[l, bend left, "\left({}^{\mathcal{J}2}\boldsymbol{T}_{\mathcal{J}1}\left(\theta_2\right)\right)^{-1}"] \arrow[r, bend left, "{}^{\mathcal{E}_F}\boldsymbol{T}_{\mathcal{J}2}"] & \mathcal{E}_F \arrow[l, bend left, "\left({}^{\mathcal{E}_F}\boldsymbol{T}_{\mathcal{J}2}\right)^{-1}"] \arrow[r, bend left, "{}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{E}_F}"] & \mathcal{E} \arrow[l, bend left, "\left({}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{E}_F}\right)^{-1}"]
\end{tikzcd}$$

Figure 6.4: *MUAV* coordinate frame transformation chain

This chain is based on the coordinate conventions of the full *MUAV* as given in Figure 6.3. The first step is thus the consideration of the *UAV* position which is represented as translation to the *UAV* vehicle frame $\mathcal{V}$ with respect to the global coordinates $\mathcal{G}$

$$\left({}^{\mathcal{V}}\boldsymbol{T}_{\mathcal{G}}\left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\,(t)\right)\right)^{-1} = \begin{bmatrix} 1 & 0 & 0 & {}^{\mathcal{G}}x \\ 0 & 1 & 0 & {}^{\mathcal{G}}y \\ 0 & 0 & 1 & {}^{\mathcal{G}}z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.1}$$

In contrast to the *UAV* position, pitch Θ and roll Φ angle do directly affect the end effector position. Therefore the simple direction vector model is not appropriate for the end effector control and the full quaternion orientation description introduced in (2.23) is used. Accordingly, the transformation from the *DJI M100* body frame $\mathcal{B}$ into the vehicle frame $\mathcal{V}$ is given by the transformation matrix $\boldsymbol{T}_{\mathcal{B}}^{\mathcal{V}}\left({}^{\mathcal{G}}\boldsymbol{\rho}\,(t)\right)$ which consists of a rotation matrix (2.23)) and uses the *UAV* orientation in form of the quaternion ${}^{\mathcal{G}}\boldsymbol{\rho}\,(t)$. To map the coordinates of the manipulator base frame $\mathcal{M}$ onto the *UAV* body coordinates $\mathcal{B}$, the coordinates have to be rotated, as the *DH* for the manipulator requires the coordinate alignment according to Figure 2.3. The rotation is reflecting that the $x_{\mathcal{M}}$-axis is mapped onto the $-{}^{\mathcal{B}}z$-axis, the $y_{\mathcal{M}}$-axis onto the $x_{\mathcal{B}}$-axis and the $z_{\mathcal{M}}$-axis onto the $-{}^{\mathcal{B}}y$-axis. In addition, the manipulator base is not directly installed in the center of gravity of the *UAV*, wherefore an additional translation ${}^{\mathcal{B}}\overrightarrow{\boldsymbol{p}}_{\mathcal{M}}$ between $\mathcal{M}$ and $\mathcal{B}$ is required. The combined transformation matrix is respectively given as

$$\left({}^{\mathcal{M}}\boldsymbol{T}_{\mathcal{B}}\right)^{-1} = \begin{bmatrix} 0 & 1 & 0 & {}^{\mathcal{B}}x_{\mathcal{M}} \\ 0 & 0 & -1 & {}^{\mathcal{B}}y_{\mathcal{M}} \\ -1 & 0 & 0 & {}^{\mathcal{B}}z_{\mathcal{M}} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.2}$$

The next steps are the joint transformations (2.26) according to the *DH* convention. The first manipulator link is only describing the translation of joint 1 $\mathcal{J}_1$ with respect to the base frame $\mathcal{M}$. As the manipulator joints are aligned with the manipulator mounting center, This translation can be described in *DH* parameters

$$\left({}^{\mathcal{J}1}\boldsymbol{T}_{\mathcal{M}}\right)^{-1} = \begin{bmatrix} 1 & 0 & 0 & a_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.3}$$

Due to the manipulator configurations, the joints do not have a twist, wherefore the twist angle $\alpha_1 = \alpha_2 = 0$. The alignment of joints and manipulator base yields $d_1 = d_2 = 0$, wherefore (2.26) results for joint 1 (with $d = d_1$) and joint 2 (with $d = d_2$) to

$$\left({}^{\mathcal{J}2}\boldsymbol{T}_{\mathcal{J}1}\left(\theta\left(t\right)\right)\right)^{-1} = \left({}^{\mathcal{E}_F}\boldsymbol{T}_{\mathcal{J}2}\left(\theta\left(t\right)\right)\right)^{-1} = \begin{bmatrix} \cos\left(\theta\right) & -\sin\left(\theta\right) & 0 & a\cos\left(\theta\right) \\ \sin\left(\theta\right) & \cos\left(\theta\right) & 0 & a\sin\left(\theta\right) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.4}$$

Finally, only the coordinate transformation from the front end effector frame $\mathcal{E}_F$ (see Figure 6.2) to $\mathcal{E}$ is missing. This transformation is depending on the task of the *MUAV*. Examples for such tasks are the grasping of an object, the manipulation with a grasped object, etc. These examples can be illustrated with use case scenarios. In order to grasp e.g. a bottle, the bottleneck has to be between the gripper tongs. Therefore, it makes sense to define $\mathcal{E}$ in the center of the gripper as illustrated in Figure 6.3 to coincide with $\mathcal{E}_C$. To manipulate with the bottle (pouring out liquid), the position of the bottle opening is more important. Thus, $\mathcal{E}$ is chosen to coincide with $\mathcal{P}$ in this case. According to the task $\mathcal{E}$ is therefore translated with respect to $\mathcal{E}_F$. In Figure 6.3 $\mathcal{E}$ is shown to represent the center between the front bottom corners of the gripper. In order to generalize this definition, the end effector coordinate frame $\mathcal{E}$

is defined with axes parallel to the $\mathcal{B}$-axes for $\theta_1 = -\pi/2$ and $\theta_2 = 0$. As the coordinate axes $\mathcal{E}_F$ are defined in DH-convention, they have to be rotated accordingly. Combined with the translation ${}^{\tilde{\mathcal{E}}_F}\vec{\boldsymbol{p}}_{\mathcal{E}}$ between frame $\mathcal{E}_F$ and $\mathcal{E}_F$, the complete transformation yields

$$\left({}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{E}_F}\right)^{-1} = \begin{bmatrix} 1 & 0 & 0 & {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}} \\ 0 & 0 & 1 & {}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}} \\ 0 & -1 & 0 & -{}^{\tilde{\mathcal{E}}_F}y_{\mathcal{E}} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.5}$$

The $\tilde{}$ is hereby indicating that ${}^{\tilde{\mathcal{E}}_F}\vec{\boldsymbol{p}}_{\mathcal{E}}$ is given in already rotated coordinates (aligned with $\mathcal{E}$).

Following the transformation chain in Figure 6.4 using (6.1), (2.23), (6.2), (6.4) and (6.5), the full transformation from $\mathcal{E}$ to $\mathcal{G}$ finally results to:

$$\begin{aligned} {}^{\mathcal{G}}\boldsymbol{T}_{\mathcal{E}}\left(\theta_1, \theta_2, {}^{\mathcal{G}}\boldsymbol{\rho}, {}^{\mathcal{G}}\vec{\boldsymbol{p}}\right) &= \left[\begin{array}{ccc|c} & {}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{E}} & & {}^{\mathcal{G}}\vec{\boldsymbol{p}}_{\mathcal{E}} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \\ &= \left({}^{\mathcal{V}}\boldsymbol{T}_{\mathcal{G}}\left({}^{\mathcal{G}}\vec{\boldsymbol{p}}\,(t)\right)\right)^{-1} \left({}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{V}}\left({}^{\mathcal{G}}\boldsymbol{\rho}\,(t)\right)\right)^{-1} \left({}^{\mathcal{M}}\boldsymbol{T}_{\mathcal{B}}\right)^{-1} \left({}^{\mathcal{J}1}\boldsymbol{T}_{\mathcal{M}}\right)^{-1} \dots \\ &\dots \cdot \left({}^{\mathcal{J}2}\boldsymbol{T}_{\mathcal{J}1}\left(\theta_1\,(t)\right)\right)^{-1} \left({}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{J}2}\left(\theta_1\,(t)\right)\right)^{-1} \left({}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{E}_F}\right)^{-1} \end{aligned} \tag{6.6}$$

The end effector orientation is thereby defined by the rotation matrix

$$\begin{gathered} {}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{E}}\left(\theta_1, \theta_2, {}^{\mathcal{G}}\boldsymbol{\rho}, {}^{\mathcal{G}}\vec{\boldsymbol{p}}\right) = \\ \begin{bmatrix} \begin{bmatrix} -2(\rho_w\rho_y + \rho_x\rho_z)\cos(\theta_1+\theta_2) + \left(1 - 2\rho_y^2 - 2\rho_z^2\right)\sin(\theta_1+\theta_2) \\ 2\rho_x\rho_y - 2\rho_w\rho_z \\ \left(-2\rho_y^2 - 2\rho_z^2 + 1\right)\cos(\theta_1+\theta_2) + 2(\rho_w\rho_y + \rho_x\rho_z)\sin(\theta_1+\theta_2) \end{bmatrix}^{\mathrm{T}} \\ \begin{bmatrix} 2((\rho_w\rho_x - \rho_y\rho_z)\cos(\theta_1+\theta_2) + (\rho_x\rho_y + \rho_w\rho_z)\sin(\theta_1+\theta_2)) \\ 1 - 2\left(\rho_x^2 + \rho_z^2\right) \\ 2((\rho_x\rho_y + \rho_w\rho_z)\cos(\theta_1+\theta_2) + (\rho_y\rho_z - \rho_w\rho_x)\sin(\theta_1+\theta_2)) \end{bmatrix}^{\mathrm{T}} \\ \begin{bmatrix} \left(2\rho_x^2 + 2\rho_y^2 - 1\right)\cos(\theta_1+\theta_2) + 2(\rho_x\rho_z - \rho_w\rho_y)\sin(\theta_1+\theta_2) \\ 2(\rho_w\rho_x + \rho_y\rho_z) \\ (2\rho_x\rho_z - 2\rho_w\rho_y)\cos(\theta_1+\theta_2) + \left(1 - 2\rho_x^2 - 2\rho_y^2\right)\sin(\theta_1+\theta_2 \end{bmatrix}^{\mathrm{T}} \end{bmatrix}, \end{gathered} \tag{6.7}$$

while the end effector position is expressed by the translation

$$
{}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}\left(\theta_1, \theta_2, {}^{\mathcal{G}}\boldsymbol{\rho}, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right) = \tag{6.8}
$$

$$
\begin{bmatrix}
-2{}^{\mathcal{B}}x_{\mathcal{M}}\rho_y^2 + 2\left(-a_0\rho_w + {}^{\mathcal{B}}z_{\mathcal{M}}\rho_w + \left({}^{\mathcal{B}}y_{\mathcal{M}} + {}^{\tilde{\mathcal{E}}_F}y_{\mathcal{E}}\right)\rho_x\right)\rho_y - 2{}^{\mathcal{B}}x_{\mathcal{M}}\rho_z^2 + {}^{\mathcal{B}}x_{\mathcal{M}} + {}^{\mathcal{G}}x \\
\ldots -2\left(\left({}^{\mathcal{B}}y_{\mathcal{M}} + {}^{\tilde{\mathcal{E}}_F}y_{\mathcal{E}}\right)\rho_w + \left(a_0 - {}^{\mathcal{B}}z_{\mathcal{M}}\right)\rho_x\right)\rho_z - 2a_1\left(\rho_w\rho_y + \rho_x\rho_z\right)\cos\left(\theta_1\right) \\
\ldots - \left(2\left(a_2 + {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\right)\left(\rho_w\rho_y + \rho_x\rho_z\right) + {}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}}\left(2\rho_y^2 + 2\rho_z^2 - 1\right)\right)\cos\left(\theta_1 + \theta_2\right) + a_1\left(-2\rho_y^2 - 2\rho_z^2 + 1\right)\sin\left(\theta_1\right) \\
\ldots + \left(a_2 + {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\right)\sin\left(\theta_1 + \theta_2\right) - 2\left(\left(a_2 + {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\right)\left(\rho_y^2 + \rho_z^2\right) - {}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}}\left(\rho_w\rho_y + \rho_x\rho_z\right)\right)\sin\left(\theta_1 + \theta_2\right) \\
\\
-2{}^{\mathcal{B}}y_{\mathcal{M}}\rho_x^2 - 2{}^{\tilde{\mathcal{E}}_F}y_{\mathcal{E}}\rho_x^2 + 2\left(a_0 - {}^{\mathcal{B}}z_{\mathcal{M}}\right)\rho_w\rho_x + 2{}^{\mathcal{B}}x_{\mathcal{M}}\rho_y\rho_x - 2\left({}^{\mathcal{B}}y_{\mathcal{M}} + {}^{\tilde{\mathcal{E}}_F}y_{\mathcal{E}}\right)\rho_z^2 + {}^{\mathcal{B}}y_{\mathcal{M}} + {}^{\tilde{\mathcal{E}}_F}y_{\mathcal{E}} + {}^{\mathcal{G}}y \\
\ldots +2\left({}^{\mathcal{B}}x_{\mathcal{M}}\rho_w + \left({}^{\mathcal{B}}z_{\mathcal{M}} - a_0\right)\rho_y\right)\rho_z + 2\left(a_1\left(\rho_w\rho_x - \rho_y\rho_z\right)\cos\left(\theta_1\right)\right. \\
\ldots + \left(a_2\rho_w\rho_x + {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\rho_w\rho_x + {}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}}\rho_y\rho_x + {}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}}\rho_w\rho_z - a_2\rho_y\rho_z - {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\rho_y\rho_z\right)\cos\left(\theta_1 + \theta_2\right) \\
\left.\ldots +a_1\left(\rho_x\rho_y + \rho_w\rho_z\ \right)\sin\left(\theta_1\right) + \left(\left(a_2 + {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\right)\left(\rho_x\rho_y + \rho_w\rho_z\right) + {}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}}\left(\rho_y\rho_z - \rho_w\rho_x\right)\right)\sin\left(\theta_1 + \theta_2\right)\right) \\
\\
-2\left(\rho_x^2 + \rho_y^2\right){}^{\mathcal{B}}z_{\mathcal{M}} + {}^{\mathcal{B}}z_{\mathcal{M}} + {}^{\mathcal{G}}z + 2\left({}^{\mathcal{B}}y_{\mathcal{M}} + {}^{\tilde{\mathcal{E}}_F}y_{\mathcal{E}}\right)\rho_w\rho_x - 2{}^{\mathcal{B}}x_{\mathcal{M}}\rho_w\rho_y + a_0\left(2\rho_x^2 + 2\rho_y^2 - 1\right) \\
\ldots +2\left({}^{\mathcal{B}}x_{\mathcal{M}}\rho_x + \left({}^{\mathcal{B}}y_{\mathcal{M}} + {}^{\tilde{\mathcal{E}}_F}y_{\mathcal{E}}\right)\rho_y\right)\rho_z + a_1\left(2\rho_x^2 + 2\rho_y^2 - 1\right)\cos\left(\theta_1\right) - \left(a_2 + {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\right)\cos\left(\theta_1 + \theta_2\right) \\
\ldots +2\left(-{}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}}\rho_w\rho_y + \left(a_2 + {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\right)\left(\rho_x^2 + \rho_y^2\right) + {}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}}\rho_x\rho_z\right)\cos\left(\theta_1 + \theta_2\right) + 2a_1\left(\rho_x\rho_z - \rho_w\rho_y\right)\sin\left(\theta_1\right) \\
\ldots + \left(-2{}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}}\rho_x^2 + 2\left(a_2 + {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\right)\rho_z\rho_x - 2{}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}}\rho_y^2 + {}^{\tilde{\mathcal{E}}_F}z_{\mathcal{E}} - 2\left(a_2 + {}^{\tilde{\mathcal{E}}_F}x_{\mathcal{E}}\right)\rho_w\rho_y\right)\sin\left(\theta_1 + \theta_2\right)
\end{bmatrix}.
$$

For given θ_1, θ_2, ${}^{\mathcal{G}}\boldsymbol{\rho}$, ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}$ the end effector frame pose $\mathcal{E}$ with respect to the global frame $\mathcal{G}$ and therefore the *FK*-problem are solved. All other variables are constants defined by the mechanical *MUAV* setup. The task-based definition of ${}^{\tilde{\mathcal{E}}_F}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}$ (see (6.5)) is considered to be constant between changes of the task definition. The parameters for the real *MUAV* setup are given in Table 6.1. Joint length a_0 is thereby already included in the z-component of ${}^{\mathcal{B}}\overrightarrow{\boldsymbol{p}}_{\mathcal{M}}$.

${}^{\mathcal{B}}\overrightarrow{\boldsymbol{p}}_{\mathcal{M}} = [0.02, 0, -0.13]$	${}^{\tilde{\mathcal{E}}_F}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}} = [-0.02, 0, -0.16]$	a_0=0	a_1=0.071	a_2=0.102

Table 6.1: Physical parameters of the *MUAV* setup in m

In the following, the *FK* (6.6) with orientation (6.7) and position (6.8) is utilized in the following to derive a control concept for the *MUAV* system.

6.3 Aerial manipulator control

The proposed *UAV* manipulator combination is an under-actuated *MUAV* which allows position $\overrightarrow{\boldsymbol{p}}$ and yaw Ψ-orientation changes due to the *UAV* and pitch Θ orientation changes according to the robotic arm. In contrast to hyper-redundant manipulators, this requires the *UAV* to move according to the tracked end effector pose. The contribution of this section is an *NMPC* of the *MUAV* platform which takes into consideration the *FK* of the robotic arm. Furthermore, the orientation of the robotic arm is controlled according to the closed-form algebraic solution of the *IK* using artificial angle constraints.

For a grasping scenario as shown in Figure 6.3, the gripper center $\mathcal{E}_C$ has to be steered towards the object frame $\mathcal{P}$. For this purpose, the *NMPC* from §5.3.2 is extended for *MUAV* operation. The *NMPC* design for real-time applications requires to trade off between using a precise computationally expensive model which leads to low control update intervals versus the usage of a simplified compact model allowing high control update intervals. While very precise models provide a more optimal path planning, the controller's capability to reject disturbance suffers from the larger control update intervals. In the case of the two-joint *MUAV*, a consideration of the full end effector pose within the system state vector leads to a high state dimension and complex system dynamics which are computationally expensive.

To address this complexity, the idea is to treat position and orientation tracking separately. For the position tracking, the *UAV* orientation ${}^{\mathcal{G}}\boldsymbol{\rho}$ and manipulator joint angles θ_1, θ_2 are therefore considered to be constant over the whole horizon within one control update interval. This allows end effector tracking by the explicit use of ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}$ (6.8) in the cost function J. Consequently, the *DJI M100 HMDV* (3.55) is used as dynamical constraint with the state vector (3.39). In correspondence, matrix $\boldsymbol{Q}_{ee} \in \mathcal{R}^3$ is introduced in order to penalize the end effector position error.

The control of the orientation itself is also separated into yaw Ψ and pitch Θ tracking. As the end effector joints of the considered *MUAV* are perpendicular to the ${}^{\mathcal{B}}x{}^{\mathcal{B}}z$-plane, the yaw orientation of the end effector ${}^{\mathcal{G}}\Psi_{\mathcal{E}}$ is defined by the *UAV* orientation. Consequently, instead of tracking the end effector Ψ-orientation, the *UAV* Ψ-orientation can be tracked as shown in §5.3.2 using the direction vector states in system function (3.40). However, some industrial applications require the grasping of cylindrical objects with their center axis normal to the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane. Hence, they are ambiguous in their Ψ-orientation (see grasped cylinder in Figure 6.3). As the ${}^{\mathcal{G}}\Psi_{\mathcal{E}}$-orientation can take every desired value, it is advantageous to point the *UAV* towards the object position instead of tracking a specific Ψ angle. For this purpose the vector ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{P}} - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}$ is projected onto the ${}^{\mathcal{G}}x{}^{\mathcal{G}}y$-plane. This projection can then directly be tracked with the direction vector description of the *UAV* orientation

$$L_{\Psi} = 0.5 q_{\Psi} \left(\frac{{}^{\mathcal{G}}x_{\mathcal{P}} - {}^{\mathcal{G}}x}{d_{xy}} - {}^{\mathcal{G}}o_{x} \right)^2 + 0.5 q_{\Psi} \left(\frac{{}^{\mathcal{G}}y_{\mathcal{P}} - {}^{\mathcal{G}}y_{\mathcal{B}}}{d_{xy}} - {}^{\mathcal{G}}o_{y} \right)^2 \tag{6.9}$$
$$\text{with } d_{xy} = \sqrt{\left({}^{\mathcal{G}}x_{\mathcal{P}} - {}^{\mathcal{G}}\underline{x}\right)^2 + \left({}^{\mathcal{G}}y_{\mathcal{P}} - {}^{\mathcal{G}}\underline{y}\right)^2}$$

using penalty factor q_{Ψ}. It might appear more intuitive to use the end effector position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}$ rather than the *UAV* position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}$. However, due to disturbance, it is possible that ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}$ could overshoot with respect to the tracking position. The result would be direction change by $\pm\pi$. As the position tracking is trying to track exactly the point where this switching happens, this might lead to an unstable behavior during the grasping procedure. For this reason, the utilization of the *UAV* position as reference position for the Ψ calculation is advantageous if its projections are distinct from ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{P}}$. This is typically the case for $-\pi/2 \lll {}^{\mathcal{G}}\Theta_{\mathcal{E}} \lll \pi/2$ depending on ${}^{\mathcal{E}_E}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}$. The resulting cost function (6.10) includes the quadratic tracking of the desired end effector position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E},des}$, the direction L_{Ψ} (6.9), the input $\boldsymbol{u}$ and the state trajectory $\boldsymbol{x}$ tracking. The *UAV* and end effector position tracking can be contradicting, therefore it is necessary to "switch off" the undesired tracking according to the task. For this purpose, the opposing tracking objective is "switched off" by setting the corresponding position tracking penalties in $\boldsymbol{Q} \in \mathbb{R}^{9\times 9}$, respectively $\boldsymbol{Q}_{ee} \in \mathbb{R}^{3\times 3}$ to zero. This is also necessary for the orientation tracking versus direction tracking ($\boldsymbol{Q} \in \mathbb{R}^{9\times 9}$, $q_{\Psi} \in \mathbb{R}$).

Using the *HMDV* (3.40) and the end effector position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}$ (6.8), the *NMPC* inherent *OCP* at time instance t results to

$$\begin{aligned}
\min_{\boldsymbol{u}} \ J = \int_{t}^{t+T} & \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{P}}(\tau) - {}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}_{\mathcal{E}}\left(\theta_1(\tau), \theta_2(\tau), {}^{\mathcal{G}}\underline{\boldsymbol{\rho}}(\tau), {}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}(\tau)\right)\right)^{\mathrm{T}} \boldsymbol{Q}_{ee} \\
& \left({}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{P}}(\tau) - {}^{\mathcal{E}}\underline{\overrightarrow{\boldsymbol{p}}}_{\mathcal{G}}\left(\theta_1(\tau), \theta_2(\tau), {}^{\mathcal{G}}\underline{\boldsymbol{\rho}}(\tau), {}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}(\tau)\right)\right) \\
& + 0.5 q_{\Psi} \left(\frac{{}^{\mathcal{G}}\underline{x}_{\mathcal{P}}(t) - {}^{\mathcal{G}}\underline{x}_{\mathcal{B}}(\tau)}{d_{xy}} - {}^{\mathcal{G}}\underline{o_{x}}_{\mathcal{B}} \right)^2 + 0.5 q_{\Psi} \left(\frac{{}^{\mathcal{G}}\underline{y}_{\mathcal{P}}(t) - {}^{\mathcal{G}}\underline{y}_{\mathcal{B}}(\tau)}{d_{xy}} - {}^{\mathcal{G}}\underline{o_{y}}_{\mathcal{B}} \right)^2 \\
& + \left(\underline{\boldsymbol{x}_{\boldsymbol{des}}}(t) - \underline{\boldsymbol{x}}(\tau)\right)^{\mathrm{T}} \boldsymbol{Q} \left(\underline{\boldsymbol{x}_{\boldsymbol{des}}}(t) - \underline{\boldsymbol{x}}(\tau)\right) + \underline{\boldsymbol{u}}(\tau)^{\mathrm{T}} \boldsymbol{R}\underline{\boldsymbol{u}}(\tau) \quad d\tau.
\end{aligned} \tag{6.10}$$
$$\text{with } d_{xy} = \sqrt{\left({}^{\mathcal{G}}x_{\mathcal{P}} - {}^{\mathcal{G}}\underline{x}\right)^2 + \left({}^{\mathcal{G}}y_{\mathcal{P}} - {}^{\mathcal{G}}\underline{y}\right)^2}$$
$$s.\ t.\ \dot{\boldsymbol{x}} = \boldsymbol{f}\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, \tau\right) \tag{6.11}$$
$$\boldsymbol{0} \leq \boldsymbol{c_{in}} = \left[\underline{u}_{\mathcal{V}1_x}{}^2 - 1, \underline{u}_{\mathcal{V}1_y}{}^2 - 1, \underline{u}_{\mathcal{V}1_z}{}^2 - 1, \underline{u}_{\mathcal{V}1_\omega}{}^2 - 1\right]^{\mathrm{T}} \tag{6.12}$$
$$\underline{\boldsymbol{x}}_0(t) = \boldsymbol{x}(t) \tag{6.13}$$

As the kinematic *NMPC* is already controlling ${}^{\mathcal{G}}\Psi_{\mathcal{E}}$ and the *DJI M100* inherent controller is steering $\lim\limits_{t\to\infty} {}^{\mathcal{G}}\Phi_{\mathcal{E}} \to 0$, the only missing *DOF* is the pitching ${}^{\mathcal{G}}\Theta_{\mathcal{E}}$ of the *MUAV*. This problem to determine the joint parameters for a given end effector pose is known as inverse kinematic problem *IK*. However, the solution of *IK* is not trivial as the *FK* of multi-joint robotic arms are not generally a bijective map. In order to determine the separate influences on the end effector ${}^{\mathcal{G}}\Theta_{\mathcal{E}}$, the end effector rotation matrix ${}^{\mathcal{G}}\boldsymbol{R}_{\mathcal{E}}\left(\theta_1, \theta_2, {}^{\mathcal{G}}\boldsymbol{\rho}, {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}\right)$ (6.7) is converted into Euler-angles using (2.16). For this purpose also the *UAV* quaternion description ${}^{\mathcal{G}}\boldsymbol{\rho}$ can be expressed with mapping (2.24) as Euler angles ${}^{\mathcal{G}}\Phi, {}^{\mathcal{G}}\Theta, {}^{\mathcal{G}}\Psi$.

Considering the stationary condition ${}^{\mathcal{G}}\Phi_{\mathcal{E}} = 0$, the end effector pitch can be determined using (2.16) which finally results to

$$\begin{aligned} {}^{\mathcal{G}}\Theta_{\mathcal{E}} &= \operatorname{arctan2}\left(-{}^{\mathcal{G}}R_{\mathcal{E},31}, \sqrt{{}^{\mathcal{G}}R_{\mathcal{E},11}{}^2 + {}^{\mathcal{G}}R_{\mathcal{E},21}{}^2}\right) \\ &\overset{{}^{\mathcal{G}}\Theta_{\mathcal{E}}\to 0}{=} \operatorname{arctan2}\left(\cot\left({}^{\mathcal{G}}\Theta - \theta_1 - \theta_2\right)\right) = \frac{\pi}{2} - \theta_1 - \theta_2 + {}^{\mathcal{G}}\Theta. \end{aligned} \tag{6.14}$$

However, equation (6.14) is under-determined, as the two joint angles θ_1 and θ_2 have to be determined. To address this issue, additional constraining conditions are required. Using the artificial joint angle constraint $\theta_1 = \theta_2$ the pitch angle control law directly follows to

$$\theta_1 = \theta_2 = -\frac{{}^{\mathcal{G}}\Theta_{\mathcal{E},des} - {}^{\mathcal{G}}\Theta - \frac{\pi}{2}}{2}. \tag{6.15}$$

A few remarks about the presented control approach: As the tracking idea is to make $\mathcal{E}_C$ and $\mathcal{P}$ coincide, the *MUAV*'s end effector pose directly relates to the object pose. From the moment when the object is grasped, this condition is always fulfilled $u \to 0$, wherefore the *MUAV* is drifting according to the present disturbance. To address this issue, the control objective of the *MUAV* has to be changed as soon as a static connection with the object is established. It shall also be kept in mind, that for real manipulation tasks, it might be necessary to impose environment constraints, as the *OCP* (6.10)-(6.13) does not consider the physical boundaries of the object to grasp nor its environment.

To conclude this section, an *NMPC* approach has been developed treating end effector position and *UAV* orientation independently to reach a given end effector pose with an under-actuated *MUAV*.

6.4 Experimental validation

To validate the effectiveness of the proposed aerial manipulation control described in §6.3, two experiments are conducted. First, the end effector pose tracking is verified using a real *MUAV*. Second, a bottle grasping scenario is used to demonstrate the applicability of the proposed approach. The *NMPC* feedback of the *DJI M100* pose is thereby provided by a *OPTITRACK* motion capture system. The end effector pose is determined from the *UAV* base via its *FK*.

6.4.1 End-effector pose tracking scenario

The *NMPC OCP* (6.10)-(6.13) of this validation scenario is parametrized according to Table 6.2. Hence, the choice of penalty matrices leads to an *NMPC* tracking of a given end effector position and yaw-orientation. Accordingly, the tracking of the object direction is switched off with $q_\Psi = 0$. Furthermore, a small state penalty is imposed on the lateral *UAV* velocities in order to reduce position overshoot.

$\boldsymbol{Q}$	$\boldsymbol{Q}_{ee}$	$\boldsymbol{R}$	q_Ψ
$\text{diag}\left([0,0,0,1,1,.01,.01,.01,0]\right)$	$\text{diag}\left([1,1,1]\right)$	$\text{diag}\left([10.0,10.0,5.0,1.0]\right)$	0

Table 6.2: End-effector pose tracking scenario: control parametrization

To demonstrate the closed-loop behavior, the desired end effector pose is changed stepwise. The chosen target positions thereby represent the corners of a square. The desired end effector Ψ-orientation is given always opposite to the direction of the center. In order to show the influence of the manipulator, the Θ-orientation is alternated between ${}^{\mathcal{G}}\Theta_{\mathcal{E},des} \in \{-30°, 0°, 30°\}$. The given reference and resulting system path is visualized in Figure 6.5 with the corresponding perspective views. Figure 6.5 confirms that the *NMPC* is able to steer the *UAV* to reach a desired end effector pose. The effect of the *FK* back-propagation by solving *OCP* (6.10)-(6.13) is thereby clearly visible, as distinguishable square *UAV* paths on three different heights are visible. These are a direct result of the three given ${}^{\mathcal{G}}\Theta_{\mathcal{E},des}$-angles.

The time relation of the *MUAV* trajectories is shown in Figure 6.6-6.7. Within these figures, the imposed ${}^{\mathcal{G}}\Theta_{\mathcal{E},des}$-angle changes are indicated by vertical dashed lines. The position plots in ${}^{\mathcal{G}}x$ (Figure 6.6a) and ${}^{\mathcal{G}}y$ (Figure 6.6b) show the expected step-wise square pattern and the convergence of the end effector position ${}^{\mathcal{G}}x_{\mathcal{E}}$ towards ${}^{\mathcal{G}}x_{\mathcal{E},des}$, respectively ${}^{\mathcal{E}}y_{\mathcal{G}}$ towards ${}^{\mathcal{G}}y_{\mathcal{E},des}$. The *UAV* position is hereby given to demonstrate the displacement with respect to the end effector position. Step changes

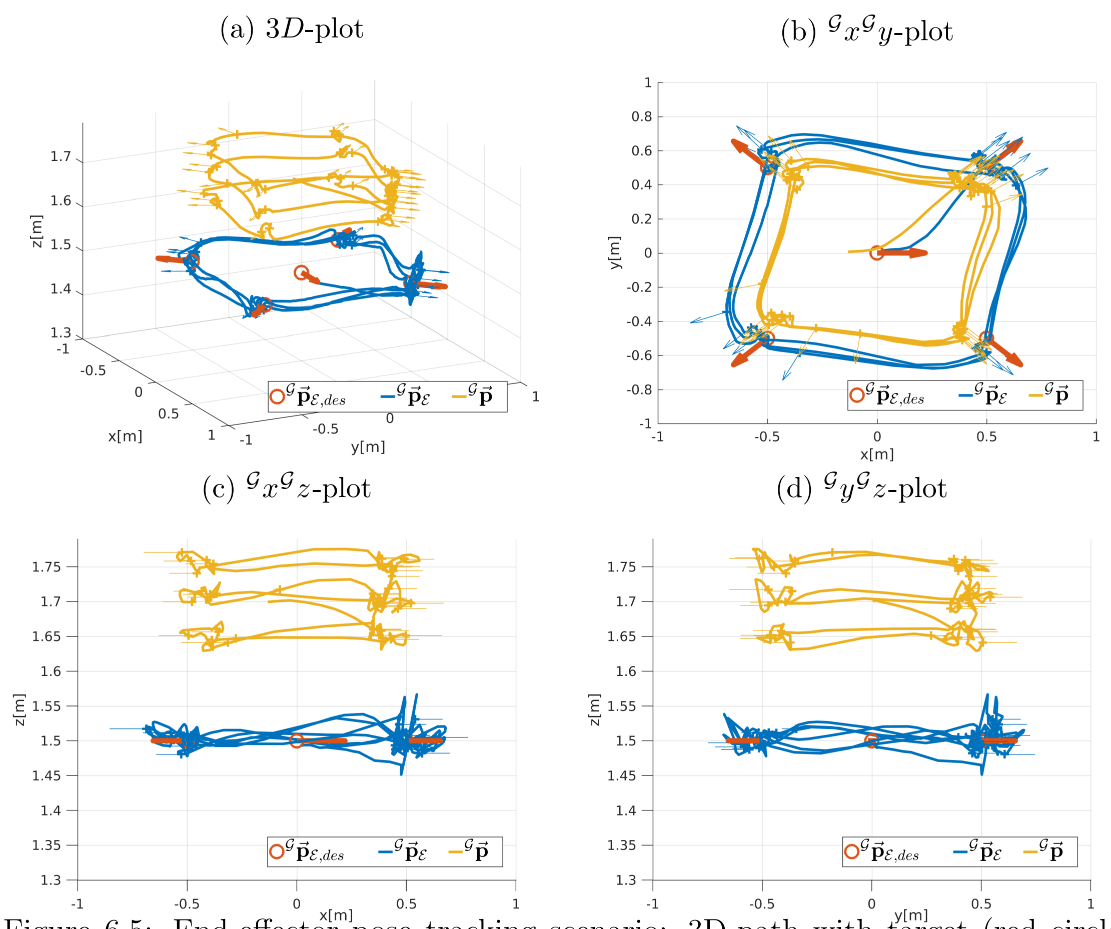


Figure 6.5: End-effector pose tracking scenario: 3D-path with target (red circles), end-effector (blue), *UAV* center (orange)

in ${}^{\mathcal{G}}z_{\mathcal{E},des}$ have been omitted for visibility reasons. However, a change in the manipulator orientation ${}^{\mathcal{G}}\Theta_{\mathcal{E}}$ is physically related to a change of the translation between *UAV* and end effector position. As the end effector position is tracked, the *UAV* position is changing accordingly. Figure 6.6c states, that the altitude ${}^{\mathcal{G}}z_{\mathcal{E},des}$ is tracked as expected. The heading control response in Figure 6.6d is also stating the convergence towards the desired ${}^{\mathcal{G}}\Psi_{\mathcal{E},des}$-orientation.

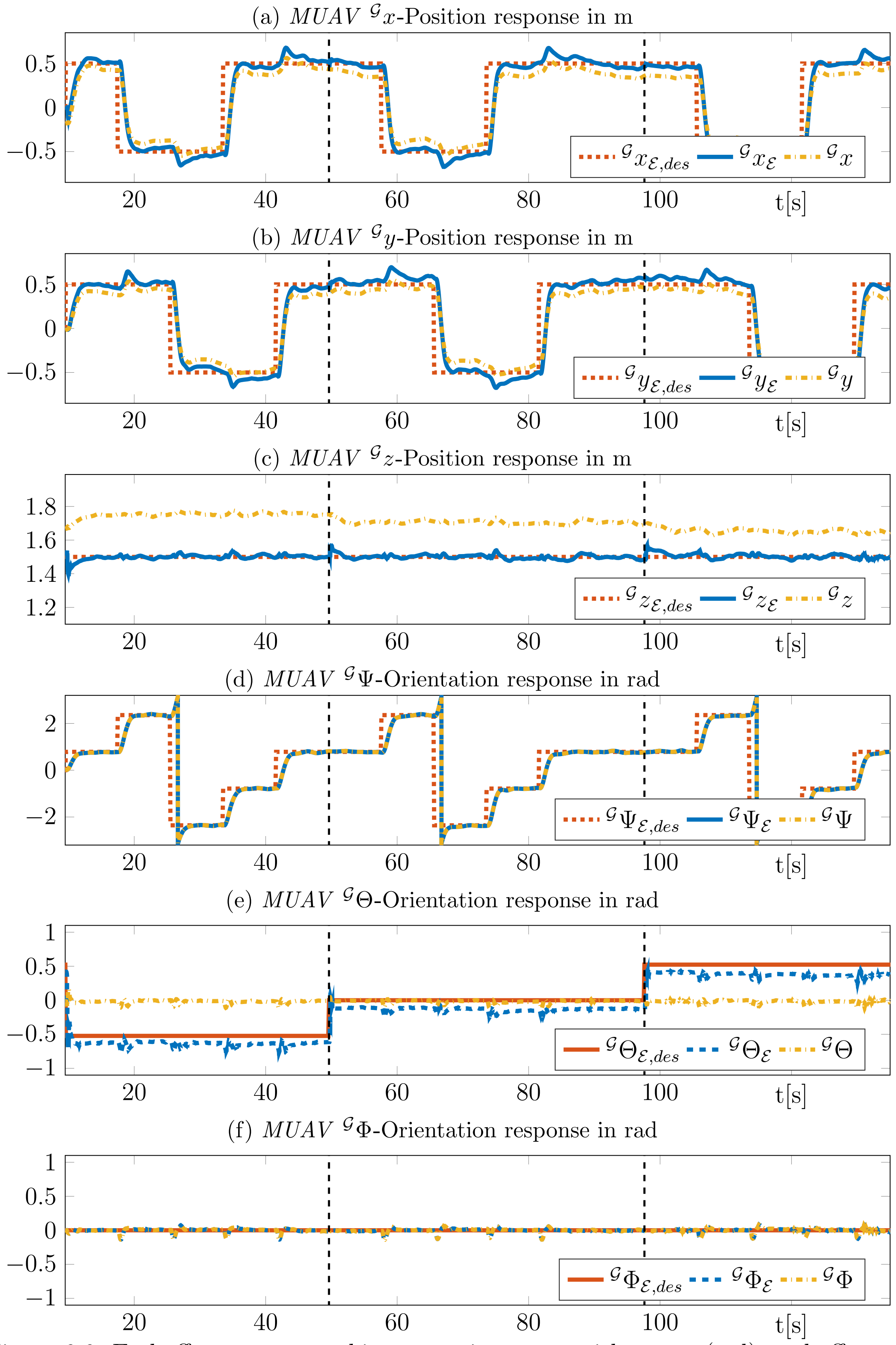


Figure 6.6: End-effector pose tracking scenario: states with target (red), end-effector (blue), *UAV* center (orange)

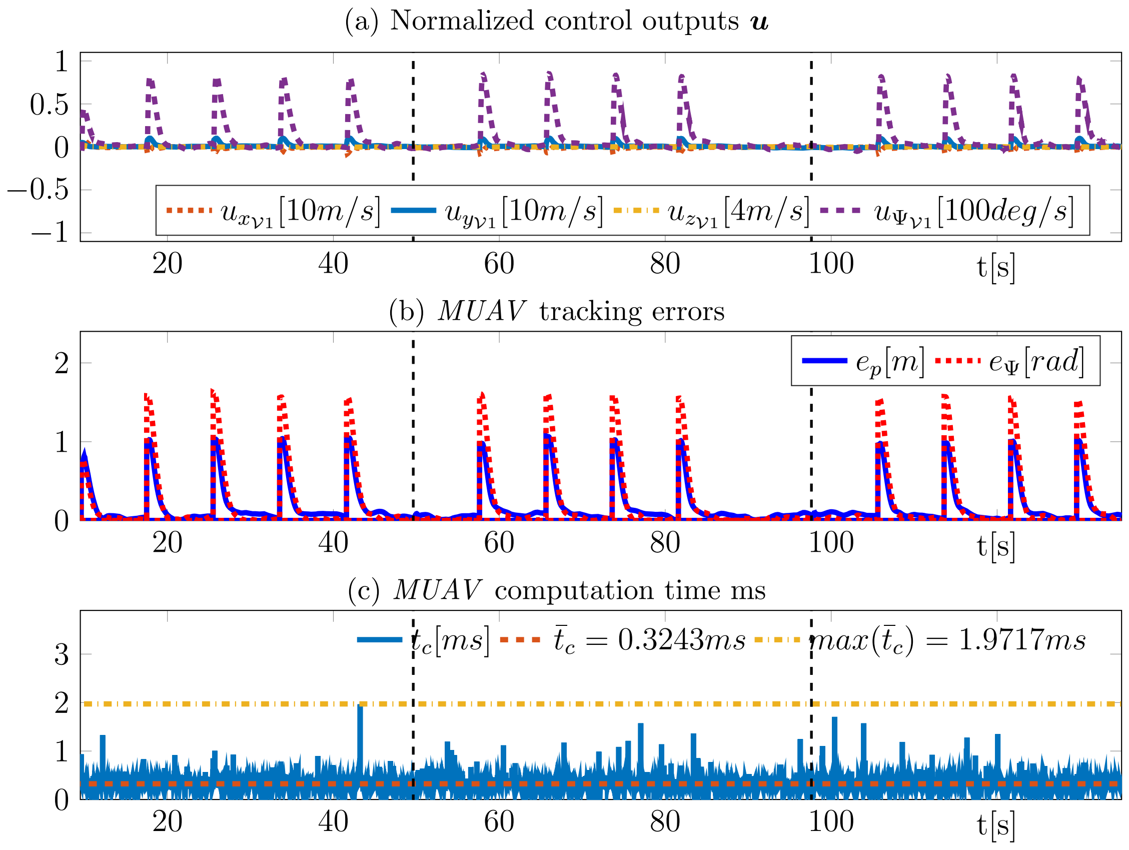


Figure 6.7: End-effector pose tracking scenario: control influence

It shall be mentioned here, that the angle limits of ${}^{\mathcal{G}}\Psi_{\mathcal{E}}$ are leading to steps between $\pm\pi$. The ${}^{\mathcal{G}}\Theta_{\mathcal{E}}$ response is given in Figure 6.6e. As expected a slight offset is visible, as the feed-forward control of the servos is not exact and the manipulator frame not perfectly stiff. Remarkable is here, that disturbance in the UAV pitch-angle ${}^{\mathcal{G}}\Theta$ results in higher ${}^{\mathcal{G}}\Theta_{\mathcal{E}}$ changes. This amplification of small orientation changes is assumed to be caused by the delayed response of the manipulator servos, wherefore it might be advantageous to use ${}^{\mathcal{G}}\Theta_{\mathcal{E}} = 0$, depending on the UAV's internal Θ response time. The high performance of the *DJI M100* attitude controller is visible in Figure 6.6f, by a very low disturbance rejection time. The normalized control outputs are given in Figure 6.7a and show the expected action peaks for trajectory changes, while maintaining the nominal state in between to keep energy consumption low. The related tracking error in position ${}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}$ and yaw ${}^{\mathcal{G}}\Psi_{\mathcal{E}}$-orientation tracking is plotted in Figure 6.7b. Here, the control performance is clearly visible by minimizing the related errors after each trajectory change. Finally, to evaluate the computational performance of the proposed approach, Figure 6.7c is showing the computation time t_c for each control update interval. With an average computation time of $\overline{t_c} = 1.9717\,\mathrm{ms}$

and a control update interval of $\Delta t = 10\,\text{ms}$ on a *Dell Latitude E5440* (see §A.1), the real-time capability of the proposed control approach is evident. To conclude, the experimental results of the end effector pose tracking scenario are validating the desired control behavior and state the performance as well as real-time capability of the control approach introduced in §6.3.

6.4.2 Bottle grasping scenario

To demonstrate the applicability of the proposed *NMPC* for industrial applications, a bottle grasping scenario is demonstrated with the *MUAV*. Like the *UAV* position, the bottle-neck position ${}^{\mathcal{G}}\vec{\boldsymbol{p}}_{\mathcal{P}}$ is localized using a motion capture system. The outline of the scenario can be separated into three consecutive steps which are displayed in Figure 6.8. In the initial state, the *MUAV* is stabilizing in its home position ${}^{\mathcal{G}}\vec{\boldsymbol{p}}_{des} = [-1, 0, 1.4]^{\mathrm{T}}$ with ${}^{\mathcal{G}}o_{x\,des} = 1$, ${}^{\mathcal{G}}o_{y\,des} = 0$ and ${}^{\mathcal{G}}\Theta_{des} = -30° = -0.52\,\text{rad}$. After giving an initial "Start" command, the bottle-neck position is tracked with the end effector. The *MUAV* ${}^{\mathcal{G}}\Psi$-orientation is thereby tracked with (6.9), to point the *MUAV* towards the bottle-neck ${}^{\mathcal{G}}\vec{\boldsymbol{p}}_{\mathcal{P}}$. As result, the *MUAV* is approaching the bottle. The third step is to close the gripper after reaching the bottle-neck position $\left\|{}^{\mathcal{G}}\vec{\boldsymbol{p}}_{\mathcal{P}} - {}^{\mathcal{G}}\vec{\boldsymbol{p}}_{\mathcal{E}}\right\| \leq 0.006\,\text{m}$. The closing procedure is using a timer with $t_{close} = 2\,\text{s}$ after which the system is returning to step one. The resulting path is indicated by the picture sequence in Figure 6.9 and confirms that the bottle is grasped.

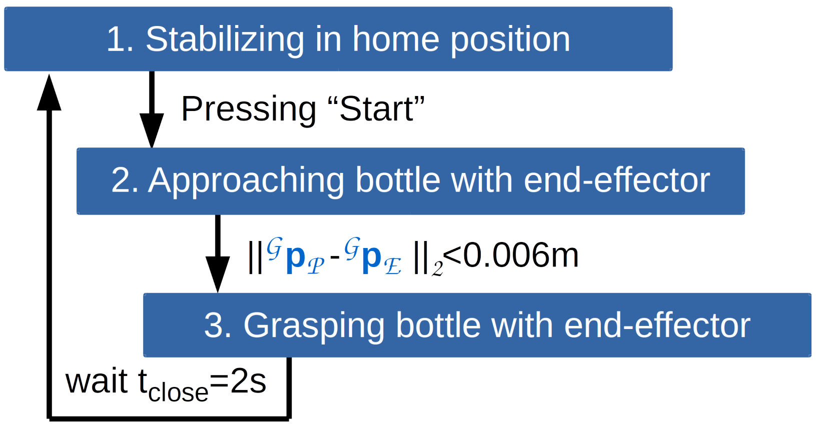

1. Stabilizing in home position
Pressing "Start"
2. Approaching bottle with end-effector
$||{}^{\mathcal{G}}\mathbf{p}_{\mathcal{P}} - {}^{\mathcal{G}}\mathbf{p}_{\mathcal{E}}||_2 < 0.006$m
3. Grasping bottle with end-effector
wait t_{close}=2s

Figure 6.8: *MUAV* grasping scenario: step chain with transition conditions

To switch between the tracking of the home position with the *UAV* center of mass and the tracking of the bottle-neck with the *MUAV* end effector, the control parametrization is changed between steps. The utilized parametrization is given in Table 6.3 and includes the switching of the *IK* pitch control.

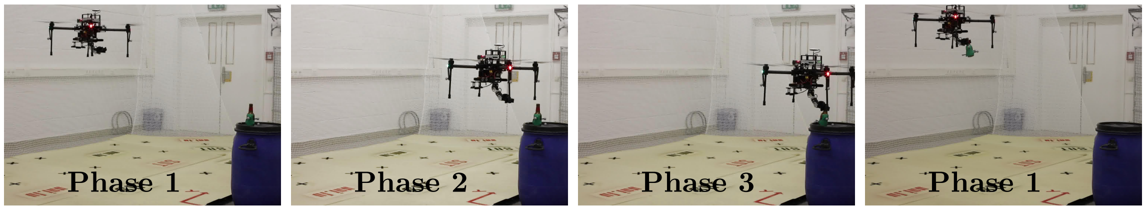


Figure 6.9: *MUAV* grasping scenario: scenario sequence using *DJI M100* with two-joint robotic arm

	Step 1.	Step 2. & 3.
$\boldsymbol{Q}$	$\mathrm{diag}\left([1., 1., 1., 1., 1., .5, .5, .5, 0]\right)$	$\mathrm{diag}\left([0, 0, 0, 0, 0, .01, .01, .01, 0]\right)$
$\boldsymbol{Q}_{ee}$	$\mathrm{diag}\left([0, 0, 0])\right]$	$\mathrm{diag}\left([1, 1, 1])\right]$
$\boldsymbol{R}$	$\mathrm{diag}\left([10.0, 10.0, 10.0, 1.0]\right)$	$\mathrm{diag}\left([10.0, 10.0, 5.0, 10.0]\right)$
q_{Ψ}	0	1
IK tracking	${}^{\mathcal{G}}\Theta_{\mathcal{E},des} = -30°$	${}^{\mathcal{G}}\Theta_{\mathcal{E},des} = {}^{\mathcal{G}}\Theta_{\mathcal{P}}$

Table 6.3: *MUAV* grasping scenario: control parameters

The *MUAV* response is given in Figure 6.10 and shows the desired convergence towards the desired states. This validates the performance of the control strategy. The signal discontinuities in Figure 6.10 are caused by changes of the control task. In Figure 6.10d ${}^{\mathcal{G}}\Psi_{\mathcal{P}grasp}$ is indicating the direction of the bottle depending on the current *UAV* position according to (6.9). For means of visibility, the direction vector is shown here as Euler angle ${}^{\mathcal{G}}\Psi_{\mathcal{E}}$. The visible oscillations are directly related to the oscillations in ${}^{\mathcal{G}}y$ as shown in Figure 6.10b. However, the scale of the plots confirms the accuracy of the closed-loop *MUAV* system using motion capture tracking. As expected, a significant control action is only visible in the transition phase between two steps, as shown in Figure 6.10e. The computation time in Figure 6.10f confirms the real-time applicability. To sum up, the grasping of the bottle in Figure 6.9 validates the applicability of the proposed control approach for manipulation tasks.

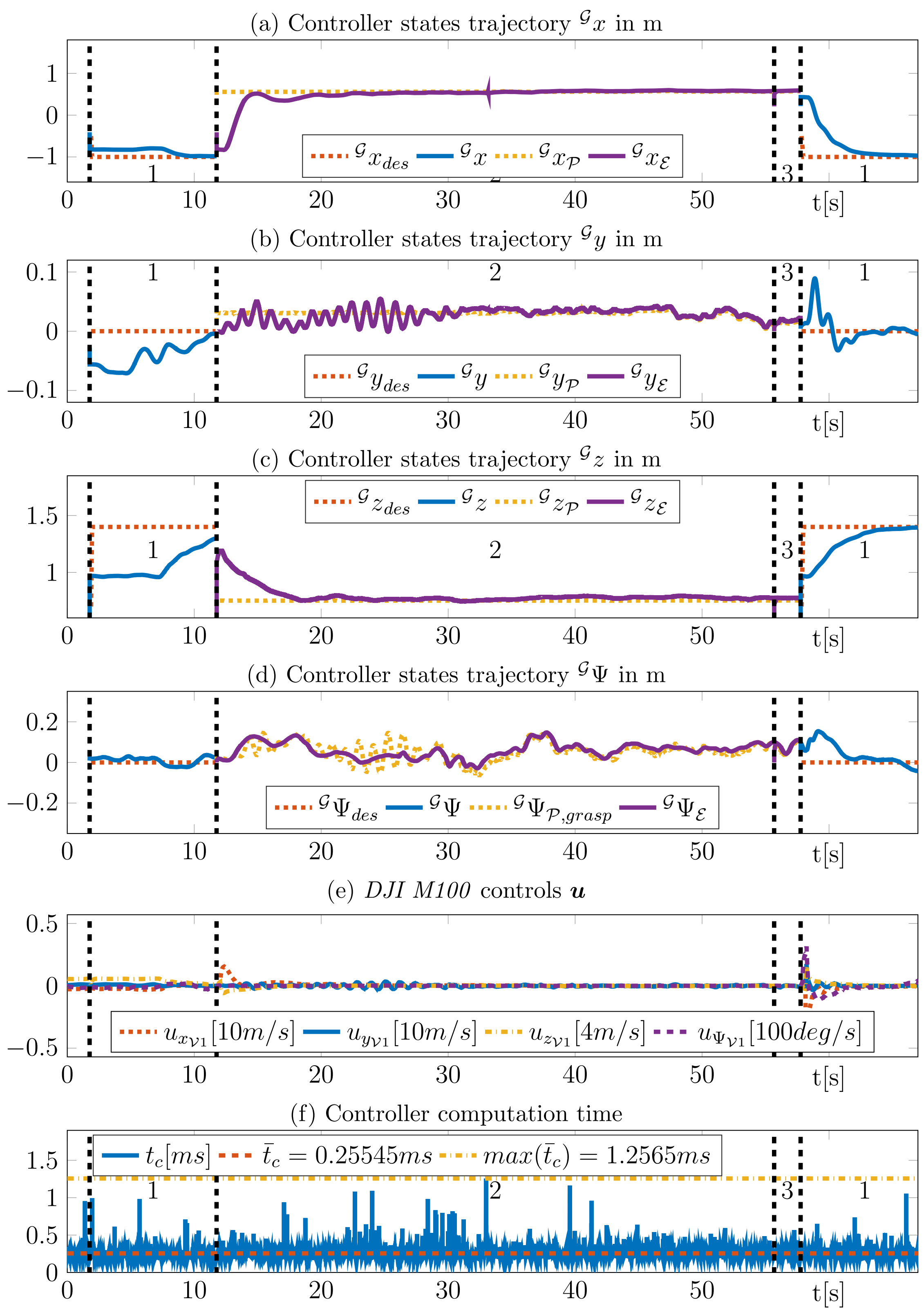


Figure 6.10: *MUAV* grasping scenario: trajectories from controller perspective

6.5 Conclusion

This chapter has presented a novel *NMPC* concept for under-actuated *MUAV*s. For this purpose, a two-joint robotic arm has been attached to the *DJI M100* quadrotor. Subsequentially, the *FK* of the combined *MUAV* have been derived. Based on thes *FK* and the *DJI M100 HMDV* (3.55) a *NMPC* pose control has been developed. The *FK* is therefore applied in the cost function of the *NMPC* to track the end effector position. For computational complexity reasons, the heading Ψ is controlled separately within the *NMPC*. Two different Ψ tracking options have been presented, to either directly track a given orientation or to point the *MUAV* towards a given object. In addition, a closed-form *IK* solution has been used to determine the necessary joint angles to reach a desired end effector pitch angle. To conclude this work, the developed approach has been successfully validated in an end effector pose tracking scenario and a simple bottle grasping application is shown.

Future work will consider the modeling of the nonlinear pitch and roll behavior of the internal *DJI M100* controller which allows deriving a full quaternion model. The performance of this computationally more complex model with the presented approach will be further investigated. In addition, the deployment of environmental constraints for grasping scenarios in cluttered environments will be investigated. Another further priority of future work will the research of industrial applications with the presented *MUAV* system.

Chapter 7

Cooperative control

Today, distributed systems are a part of every day life. Examples for such systems are the internet, globally distributed value chains, telecommunication, traffic regulation, transportation of goods, power supply systems, etc. The cooperative aerial manipulation scenario shown in Figure 1.1. represents such a distributed systems. Distributed systems consist of a multitude of interacting single entities, called agents. In the context of this work, each single UAV is such an agent.

The advantages of a distributed system can be shown with a cooperative aerial manipulation example. By deploying several UAVs to search and localize an object, measurements can be improved with sensor fusion, safety by redundancy is provided and the search time can be minimized. Additional task capabilities can be introduced to the system in the form of specialized agents. For example, a $MUAV$ can be deployed for the manipulation task as soon as the object is localized. In the meanwhile, the UAVs can provide perception information to the $MUAV$. This allows to reduce the sensing payload on the $MUAV$ which results in more available payload and energy efficient manipulation.

In order to achieve such a global objective, the agents have to be controlled cooperatively. However, the control of distributed systems suffers from their complexity and is facing two main issues:

1. Maintaining the inherent flexibility of distributed systems:
   Refers to the flexibility of changing single elements or control objectives at runtime. For example, if a UAV is defective, another UAV can take over its task. If the new robot has different physical dynamics, the control of the whole system has to be readapted to maintain an optimal result.

2. Computational efficiency for real-time applicability:
   As dynamics and interaction mechanisms of each agent have to be represented within the control, the computational burden for large sets of agents e.g. robots is high. Due to their fast dynamics, this is particularly problematic for *UAV*s.

Corresponding to the nature of large-scale distributed systems, these issues are typically addressed by a distributed control strategy. This means, every agent or subgroup of agents are controlling themselves in relation to their neighboring agents. However, for real-time *MPC* of fast systems even the control of such a subgroup is computationally challenging.

For this reason, this chapter is focusing on the development of a computationally efficient central *NMPC* strategy for small-scale systems. This precedes a future extension to distributed control approaches. To address the issues of computational efficiency while maintaining flexibility, this chapter is contributing a Distributed System Event-Based Nonlinear Model Predictive Control (*DENMPC*) framework. For this purpose, the related work is given in §7.1. To address the flexibility of the *MPC*, §7.3 presents a scheme which allows to compose an *OCP* in a modular way from different agents, constraints and control objectives. In §7.4, the *DENMPC* framework *C++*-implementation with this modularization scheme is discussed. To achieve computational efficiency, *DENMPC* is exploiting the mathematical structure of *OCP* components. This refers to the avoidance of computationally expensive matrix multiplications. In addition, each compiled *OCP* component is accessed via a fast functors and pointers scheme. Out of these compiled components, arbitrary optimal control problems can be recomposed at runtime. This results in low computation times while allowing runtime changes of the system topology, dynamics, couplings and control objectives. The change of the *OCP* can thereby be triggered via event as for example *ROS* messages, timers, etc. To state the computational efficiency of the proposed control framework, §7.5 is validating the framework features experimentally in real *UAV* scenarios. A summary and future perspective of the proposed methods is given in §7.6.

7.1 Related work in cooperative control

A very comprehensive study on swarm mechanics, interaction constraints and control is given in [GP11]. This includes control strategies for swarm expansion, contraction, rotation and topology changes. Also different agent models for example generic single and double integrator systems are discussed. In [ZM13], a survey on formation control and coordination for unmanned vehicles has been presented. It provides detailed application references for swarms of mobile robots. In this context, also references for control techniques based on virtual structures, leader-follower schemes, defined agent behaviors, graph theory and potential field approaches are given. These methods are well established for nonholonomic mobile robots and can be be applied straight forward to UAVs.

One such control technique is feedback linearization to derive leader-follower control laws for distance and orientation. This is well established for first order kinematic models of mobile robots. In [CGHC09], this concept is shown for autonomous underwater vehicles. The simple interaction scheme of leader-follower approaches thereby can be scaled easily. For this purpose, the follower only has to be considered as leader for the next robot. To achieve more complex formations, [DOK98] introduces different control laws for leader follower scenarios. By controlling distance and angle to multiple followers, the article shows how specific formations can be achieved. The developed controllers are applied on nonholonomic wheeled robots. In [DFK$^+$02], this idea is extended by the capability of switching between different leader-follower control laws. This allows to change the formation according to the scenario. A leader-follower approach for nonholonomic wheeled robots is presented in [CMPT08]. The particularity of this approach is the consideration of control constraints and a set of admissible positions. In addition, the article presents a comprehensive stability analysis.

Virtual structure approaches define the desired formation explicitly. This is presented in [MGS11] for an example of car-type mobile robots. In contrast, behavioral control is based on switching between elementary behaviors of agents. Relevant to this thesis, a multi-$MUAV$ control scenario has been presented in [MPT$^+$16]. In this context, each $MUAV$ is capable of tracking a desired end effector pose and keeping a given distance to its neighbor. Furthermore, every $MUAV$ can track according to the field of view of its sensor and avoid the manipulators mechanical joint limits. For nonholonomic wheeled robot, a behavior-based formation control approach is presented in [LBY03].

A common tool to describe distributed systems mathematically is graph theory. Control design based on graph theoretic considerations are given in [FM04, CdQ15, DF08, DSH07]. For this purpose, the connectivity and respectively rigidity of the formation is controlled using the eigenvalues of the Laplacian matrix of the related graph. [DSH07] is thereby considering network connectivity by using coverage and communication constraints.

All these control concepts are task specific and changes in the agents dynamics, scenario constraints and objectives require a redesign of the control laws. As shown in §5.6, artificial potential functions can be used to generalize control task descriptions. In [GC02], potential functions are used to describe repulsive and attractive inter-agent behavior. The robots are thereby steered according to the gradients of these potentials. Experiments on wheeled omnidirectional mobile robots with dynamic obstacle *CA* show the effectiveness of the proposed approach. The same principle is presented in [GDDSL12] for quadrotor formation flying. In addition, the controller is able to consider obstacle with repulsion attributes.

Closely related to the potential function approach, the control performance can be increased by predicting the agents future behavior. This is for example used to reduce oscillations in rigid formation scenarios. The combination of potential functions with behavior prediction leads to *MPC*. As discussed in §4, *MPC* is based on a formulation of the control scenario as optimization problem. This offers a generic way of handling complex scenarios without adapting the control policy. An example of a central *NMPC* of omnidirectional ground robots is presented in [NMSC13]. The article shows the experimental evaluation of the *NMPC* with three robots in a leader-follower as well as in a formation control scenario. In [NSCM14], the related cost function weights are determined in an iterative tuning approach. A typical example of *MPC* for *UAV*s is trajectory tracking in formation flight while considering *CA* constraints. In [KCK13], *MPC* is presented for collision free traffic regarding automotive transportation scenarios. A non-convex *MPC* for cooperative control is presented in [AMA14]. Here, the first objective is to tackle *CA*. The secondary performance objective is to deal with the quality of the collision free trajectory. A comprehensive study on multi-*UAV* *MPC* control strategies is given in [BMPL$^+$14]. The provided examples are *CA*, area exploration and formation flying using virtual structure.

The scenario size and complexity using *MPC* is limited by the computational burden in relation with real-time applicability. This computational effort can be split, by considering separate subproblems consisting of each agent with its local neighborhood. The result is a distributed *MPC*. A comprehensive collection of such algorithms is given and discussed in detail in [Neg14]. As a result, the controllers can also be applied in larger scale scenarios. The distributed controller topology thereby provides safety by redundancy. The disadvantage is the lower control performance due to the lack of global information about all scenario components. In addition, communication and convergence mechanisms have to be implemented. This is necessary to achieve convergence towards the global optimal solution. An example of such an *NMPC* distribution for fixed-wing *UAV* formation flight is given in [SK09]. The article compares central and distributed *NMPC* for a simulative leader-follower formation flight scenario with *CA*. Another distributed *MPC* example is given in [TMK12] for *UAV* trajectory planning. There, the distribution is realized by broadcasting a *UAV*'s trajectory to its neighbors in the form of trajectory polynomial coefficients. Each agent is subsequently solving the planning problem for a subsystem with its neighbors using *SQP*. In [BR11], a distributed *MPC* for *UAV* formation control is developed. For this purpose, navigation constraints are linearized with switching parameters which results in a mixed-integer quadratic program.

The objective of this chapter is to incorporate the advantages of the presented central *MPC* approaches with the advantages of distributed control approaches. In order to develop a control solution with low computation times and the desired flexibility, the global *OCP* of a distributed system is analyzed in the following regarding its key components.

7.2 OCP of distributed systems

An example of a central control of a distributed system of three coupled quadrotors is given in Figure 7.1. In the presented cooperation scenario every quadrotor represents

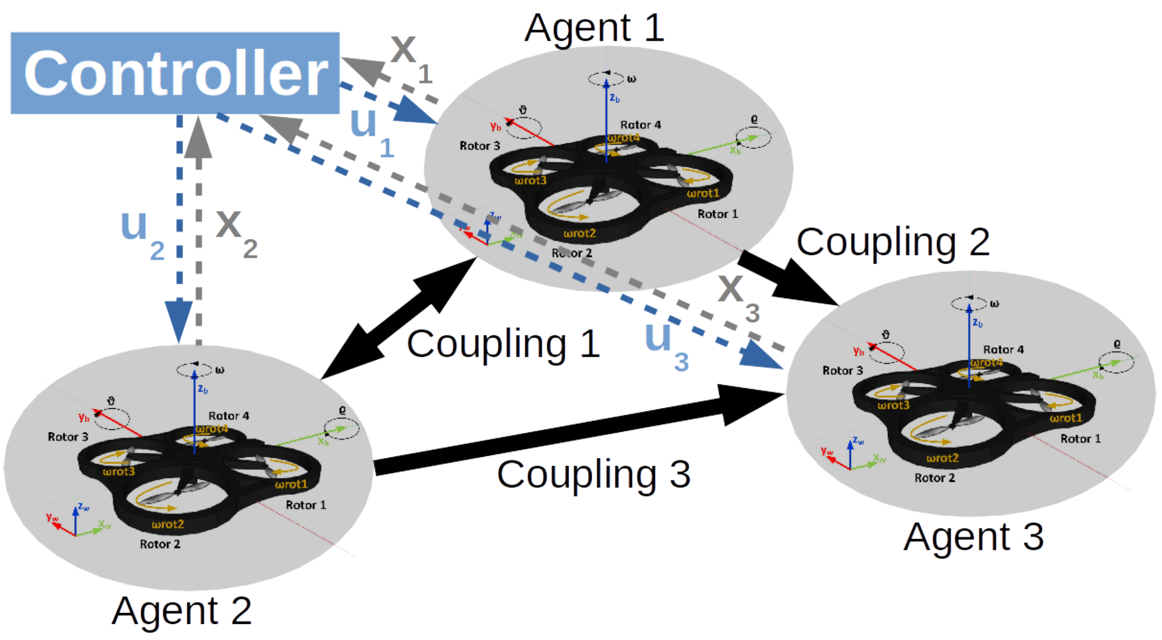


Figure 7.1: Scheme of *UAV* cooperation

an agent with its uncoupled dynamic $\boldsymbol{f}_i$, constraints $\boldsymbol{c_{eq}}_i$, $\boldsymbol{c_{in}}_i$ and costs L_i. To describe the behavior of the coupled system, the single agents are interconnected with coupling constraints $\boldsymbol{c_{in}}_{ij}$, $\boldsymbol{c_{eq}}_{ij}$ and/or coupling costs L_{ij}. The couplings in Figure 7.1 do not have to be physical couplings, but are representations of mathematical constraints like for example a *CA* constraint. They can act on both agents (e.g. "Coupling 1") or just one agent (e.g. "Coupling 3"). The arrows in Figure 7.1 are hereby representing the direction of information. For the given example of "Coupling 3", "Agent 3" reacts on "Agent 2", but not in reverse.

More generally, a distributed system consists of a set of n_ν agents $\mathcal{V} = \{\nu_1, ..., \nu_{n_\nu}\}$. In the context of this thesis, pairwise couplings between agents are considered. Accordingly, each agent $i \in \mathcal{V}$ is exclusively coupled with $n_{\eta i}$ neighbors $j \in \mathcal{N}^i = \{\nu_j \subseteq \mathcal{V} | \{\nu_i \neq \nu_j\}\}$ by the couplings $\mathcal{E} = \{\epsilon_{ij} = \{\nu_i, \nu_j\} | \nu_i, \nu_j \in \mathcal{N}^i\}$. To globally control all n_ν agents, the individual *MPC* cost functions are added up (7.1). The interaction between agents can be expressed by additional coupling stage costs $L_{ij}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_j, \boldsymbol{u}_j)$ and coupling constraints e.g. $\boldsymbol{c_{eq}}_{ij}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_j, \boldsymbol{u}_j)$ and final coupling costs $V(\boldsymbol{x}_i, t+T)$. For means of visualization, an explicit time dependency τ is omitted in the following. Accordingly, the *OCP* described in §4 is extended for a distributed systems to the global *OCP*

$$\min_{\underline{\boldsymbol{u}}(\cdot)} \quad J(\underline{\boldsymbol{u}}) = \sum_{i=1}^{n_\nu} \left(V(\underline{\boldsymbol{x}}_i, t+T) + \sum_{j \neq i}^{n_\epsilon} V_{ij}(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j) \right) + \sum_{i=1}^{n_\nu} \left(\int_t^{t+T} L_i(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i) + \sum_{j \neq i}^{n_\epsilon} L_{ij}(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j) \right) \mathrm{d}\tau \tag{7.1}$$

$$s.\ t. \quad \dot{\underline{\boldsymbol{x}}}_i = \boldsymbol{f}_i(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i), \tag{7.2}$$

$$\boldsymbol{0} = \boldsymbol{c_{eq}}_i(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i), \ \boldsymbol{c_{eq}}_i \in \mathbb{R}^{n_{ceqi}}, \tag{7.3}$$

$$\boldsymbol{0} \geq \boldsymbol{c_{in}}_i(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i), \ \boldsymbol{c_{in}}_i \in \mathbb{R}^{n_{cini}}, \tag{7.4}$$

$$\boldsymbol{0} = \boldsymbol{c_{eq}}_{ij}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_j\right), \ \boldsymbol{c_{eq}}_{ij} \in \mathbb{R}^{n_{ceqij}}, \tag{7.5}$$

$$\boldsymbol{0} \geq \boldsymbol{c_{in}}_{ij}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_j\right), \ \boldsymbol{c_{in}}_{ij} \in \mathbb{R}^{n_{cinij}}, \tag{7.6}$$

$$\underline{\boldsymbol{x}}_i(t) = \boldsymbol{x}_i(t) \tag{7.7}$$

The couplings shown in *OCP* (7.1),(7.5),(7.6) are acting on both agents equally. To realize coupling constraints and cost functions which are only acting on one agent (e.g. leader-follower constraint), the influence of the cost functions on one agent e.g. j is neglected. This is indicated with the index "$\backslash\{\boldsymbol{u}_j\}$". For example $L_{i,j}^{\backslash\{\boldsymbol{u}_j\}}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_j, \boldsymbol{u}_j)$ is a coupling cost term, that just affects agent i. This is achieved by omitting the corresponding derivative terms e.g. $\nabla_{\boldsymbol{x}_j} L_{i,j}^{\backslash\{\boldsymbol{u}_j\}} = \boldsymbol{0}$ and

$\nabla_{\boldsymbol{u}_j} L_{i,j}^{\setminus\{\boldsymbol{u}_j\}} = \mathbf{0}$ in the solution of the optimality conditions. As a remark, depending on the system dynamics, objectives and constraints, this approach can lead to suboptimal or infeasible solutions if the problem convexity is altered. In mobile robotics this can appear for example when a unidirectional robot tries to follow an omnidirectional robot.

In the following, the idea of modularizing terms of the optimality conditions is used to create the first order optimality conditions (see §4.2) dynamically. For this purpose, the *OCP* structure is analyzed for the constraint handling methods given in §4.5.

7.3 Modularization

The key to fast *MPC* is an efficient solver code, typically compiled from a fast low-level programming language e.g. *C*. Hence, standard *MPC* implementations form the *OCP* (7.1)-(7.7) in compilation time and then use the compiled fast functions at runtime. This has the advantage of a fast execution, but with the draw-back that for each adjustment of the system topology, control objective, etc., the *OCP* has to be recompiled. This contradicts a runtime adaptability required for dynamically changing distributed systems. To address this lack of flexibility, this section is contributing a modularization approach. The idea of this modularization is to compose the *OCP* at runtime from elemental precompiled functions. By exploiting the *OCP* structure, the modularization combines the flexibility of runtime adjustments with the computational speed of compiled functions.

The current implementation of the *DENMPC* framework provides the modularization of *OCP* in the form (7.1)-(7.7). Furthermore, it comes with an incorporation of the primal barrier and auxiliary variable constraint handling method (see §4.5). In order to demonstrate this central cooperative control, the following composite state vectors of the agents are considered

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1^{\mathrm{T}} & \dots & \boldsymbol{x}_{n_{\mathcal{A}}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \ \boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_1^{\mathrm{T}} & \dots & \boldsymbol{u}_{n_{\mathcal{A}}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}. \tag{7.8}$$

Without loss of generality, the modularization is shown with the reduced *OCP*

$$
\begin{array}{lllllll}
\min\limits_{\boldsymbol{u}} & J(\boldsymbol{x}, \boldsymbol{u}_i) = \sum\limits_i V_i(\boldsymbol{x}_i(t+T), t+T) + \int_t^{t+T} \sum\limits_i L_i(\boldsymbol{x}_i(\tau), \boldsymbol{u}_i(\tau), \tau)\, \mathrm{d}\tau \\
\text{s.t.} & \dot{\boldsymbol{x}}_i = \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \tau), & \boldsymbol{x}_i(t) = \boldsymbol{x}_{i,0}, \\
& \boldsymbol{0} = \boldsymbol{c}_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \tau), & \boldsymbol{0} = \boldsymbol{c}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{u}_i, \boldsymbol{u}_j), \\
& \tau \in [t, t+T] & j \in \mathcal{N}_i,\ i \in [1, n_{\mathcal{A}}]
\end{array}
\tag{7.9}
$$

for means of readability. *OCP* (7.9) comprises the atomic *OCP*s of each agent plus coupling constraints $\boldsymbol{c}_{ij}$.

The modularization is exemplarily shown for the primal barrier inequality constraint handling. As shown in §4.2.2, the Lagrangian for the cooperative *OCP* (7.9) can be calculated accordingly

$$
\begin{aligned}
\mathcal{L} = \sum_i^{n_\nu} \Big[& L_i(\boldsymbol{x}_i, \boldsymbol{u}_i, t) + \boldsymbol{\lambda}_i^\top \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{u}_i, t) - \boldsymbol{q}_{c_{in}i}{}^\top \ln(-\boldsymbol{c}_{\boldsymbol{in}i}(\boldsymbol{x}_i, \boldsymbol{u}_i, t)) \\
& + \sum_j^{n_\eta} -\boldsymbol{q}_{c_{in}i,j}{}^\top \ln\left(\boldsymbol{c}_{\boldsymbol{in}i,j}(-\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{u}_i, \boldsymbol{u}_j, t)\right) \Big].
\end{aligned}
\tag{7.10}
$$

Each agent i is contributing to the Lagrangian $\mathcal{L}$ with its individual stage costs $\mathcal{L}_i$, dynamics $\boldsymbol{f}_i$ and constraints $\boldsymbol{c}_{\boldsymbol{in}i}$ additively. This holds also for the coupling constraints $\boldsymbol{c}_{\boldsymbol{in}i,j}$, which are affecting always two agents (agent i, agent j). Like the individual constraints $\boldsymbol{c}_{\boldsymbol{in}i}$, the couplings are contributing to $\mathcal{L}$ via the primal barrier method.

Considering the composite vectors (7.8), the optimality conditions for the global *OCP* can be expressed in terms of the contributing single agent elements

$$
\boldsymbol{0} = \nabla_{\boldsymbol{u}} \mathcal{L} = \left[\nabla_{\boldsymbol{u}_1} \mathcal{L}^\top, \ldots \nabla_{\boldsymbol{u}_{n_\nu}} \mathcal{L}^\top\right]^\top \tag{7.11}
$$

$$
\dot{\boldsymbol{x}} = \nabla_{\boldsymbol{\lambda}} \mathcal{L} = \left[\boldsymbol{f}_1(\boldsymbol{u}_1, \boldsymbol{x}_1, t)^\top, \ldots \boldsymbol{f}_{n_\nu}(\boldsymbol{u}_{n_\nu}, \boldsymbol{x}_{n_\nu}, t)^\top\right]^\top. \tag{7.12}
$$

$$
\dot{\boldsymbol{\lambda}} = -\nabla_{\boldsymbol{x}} \mathcal{L} = \left[-\nabla_{\boldsymbol{x}_1} \mathcal{L}^\top, \ldots - \nabla_{\boldsymbol{x}_{n_\nu}} \mathcal{L}^\top\right]^\top. \tag{7.13}
$$

The idea is to exploit the additive structure of the Lagrangian which is also visible in its derivatives. The modularization is demonstrated for $\nabla_{\boldsymbol{u}_i}\mathcal{L}$ in (7.11)

$$0 \overset{!}{=} \nabla_{\partial \boldsymbol{u}_i}\mathcal{L} = \nabla_{\boldsymbol{u}_i} L_i\left(\boldsymbol{x}_i, \boldsymbol{u}_i, t\right) + \left(\nabla_{\boldsymbol{u}_i}\boldsymbol{f}_i\left(\boldsymbol{x}_i, \boldsymbol{u}_i, t\right)\right)^\top \boldsymbol{\lambda}_i \ldots \tag{7.14}$$

$$- \left(\nabla_{\boldsymbol{u}_i}\boldsymbol{c_{ini}}\left(\boldsymbol{x}_i, \boldsymbol{u}_i, t\right)\right) \top \left(\boldsymbol{q}_{c_{in\,i}} \backslash \boldsymbol{c_{ini}}\left(\boldsymbol{x}_i, \boldsymbol{u}_i, t\right)\right) \ldots \tag{7.15}$$

$$- \left(\nabla_{\boldsymbol{u}_i}\boldsymbol{c_{ini,j}}\left(\boldsymbol{x}_i, \boldsymbol{u}_j, \boldsymbol{u}_i, \boldsymbol{u}_j, t\right)\right) \top \left(\boldsymbol{q}_{c_{in\,i,j}} \backslash \boldsymbol{c_{ini,j}}\left(\boldsymbol{x}_i, \boldsymbol{u}_j, \boldsymbol{u}_i, \boldsymbol{u}_j, t\right)\right). \tag{7.16}$$

where "\\" is representing an element-wise division. Regarding the structure of (7.14)-(7.16), three influences can be distinguished. (7.14) consists of the agents dynamics and stage costs which typically formulate the control objective. (7.15) is associated with the constraints that are acting on agent i, while (7.16) is the influence induced by coupling constraints. In reverse conclusion, if (7.14) is provided for different agents, (7.15) for different constraints and (7.16) for different coupling constraints, the summands of this equation can be exchanged according to the dynamics, objective, constraints and couplings. To structure the global *OCP*, the modularization also has to be executed for the other optimality conditions (7.12)-(7.13). In case of (7.12) this is trivial, as it results in a concatenation of system dynamics. For (7.13) the modularization can be executed straight forward as for (7.14). In §A.2, this is shown for all constraint handling methods presented in §4.5.

Each of the summands (e.g. $\left(\nabla_{\boldsymbol{u}_i}\boldsymbol{f}_i\left(\boldsymbol{x}_i, \boldsymbol{u}_i, t\right)\right)^\top \boldsymbol{\lambda}_i$) in (7.11)-(7.13) is a vector-valued function . In order to avoid computationally expensive matrix-multiplications (e.g. $\left(\nabla_{\boldsymbol{x}_1}\boldsymbol{f}_1\right)^\mathrm{T} \boldsymbol{\lambda}_1$), *GRAMPC* [GU14] offers the possibility to provide $\left(\nabla_{\boldsymbol{x}}\boldsymbol{f}\right)^\mathrm{T} \boldsymbol{\lambda}$ as explicit function. This is particularly advantageous, as the Jacobians of the system function are typically sparse. The same principle can also be applied for all other components of the optimality conditions that consist of a multiplication of matrix-valued functions with vectors:

$$\text{e.g. } \nabla_{\boldsymbol{x}_1}\boldsymbol{f}_1\left(\boldsymbol{x}_1, \boldsymbol{u}_1, t\right)^\mathrm{T} \boldsymbol{\lambda}_1 \equiv \mathbf{s}\left(\boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{\lambda}_1, t\right) \tag{7.17}$$

Accordingly, each atomic term in Table 7.1 can be expressed by a vector-valued function ($\mathbf{s}$). As a result, no matrix multiplication is needed at runtime as these products are computed and optimized in compilation-time. In the following these vector-valued summands are referred to as atomic functions.

These can be provided in fast compiled code and modularly added or removed from the optimality conditions. For example a change of an agent has influence on all optimality conditions. Therefore, it makes sense to be able to change all of its corresponding atomic functions at once. In *DENMPC*, this is realized by packing associated atomic functions into a single *C++*-class which can be addressed via functors. Out of these predefined modules, the corresponding central *OCP* for any arbitrary topology of the system can be formed. The structure of Table 7.1 is demonstrating how *DENMPC* is associating the atomic functions to agents, constraints and couplings. The object-oriented implementation of this structure in *DENMPC* is explained in the following.

Table 7.1: Atomic functions to realize agents and couplings

"Agent" i	
Costs:	$L_i\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$, $\nabla_{\boldsymbol{x}_i} L_i\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$, $\nabla_{\boldsymbol{u}_i} L_i\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$
Final costs:	$V_i\left(\boldsymbol{x}_i\right)$, $\nabla_{\boldsymbol{x}_i} V_i\left(\boldsymbol{x}_i\right)$,
Dynamics:	$\boldsymbol{f}_i\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$, $\left(\nabla_{\boldsymbol{x}_i}\boldsymbol{f}_i\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_i$, $\left(\nabla_{\boldsymbol{u}_i}\boldsymbol{f}_i\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_i$,
Equality constraints:	$\boldsymbol{c}_{\boldsymbol{eq}_i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$, $\left(\nabla_{\boldsymbol{x}_i}\boldsymbol{c}_{\boldsymbol{eq}_i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_i}$, $\left(\nabla_{\boldsymbol{u}_i}\boldsymbol{c}_{\boldsymbol{eq}_i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_i}$, $\left(\nabla_{\boldsymbol{\nu}_i}\boldsymbol{c}_{\boldsymbol{eq}_i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_i}$
Inequality constraints:	$\boldsymbol{c}_{\boldsymbol{in}i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$, $\left(\nabla_{\boldsymbol{x}_i}\boldsymbol{c}_{\boldsymbol{in}i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}i}$, $\left(\nabla_{\boldsymbol{u}_i}\boldsymbol{c}_{\boldsymbol{in}i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}i}$, $\left(\nabla_{\boldsymbol{\nu}_i}\boldsymbol{c}_{\boldsymbol{in}i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}i}$
"Constraint": additional constraints / cost functions for agent i	
Costs:	$L_i\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$, $\nabla_{\boldsymbol{x}_i} L_i\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$, $\nabla_{\boldsymbol{u}_i} L_i\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$
Final costs:	$V_i\left(\boldsymbol{x}_i\right)$, $\nabla_{\boldsymbol{x}_i} V_i\left(\boldsymbol{x}_i\right)$,
Equality constraints:	$\boldsymbol{c}_{\boldsymbol{eq}_i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$, $\left(\nabla_{\boldsymbol{x}_i}\boldsymbol{c}_{\boldsymbol{eq}_i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_i}$, $\left(\nabla_{\boldsymbol{u}_i}\boldsymbol{c}_{\boldsymbol{eq}_i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_i}$, $\left(\nabla_{\boldsymbol{\nu}_i}\boldsymbol{c}_{\boldsymbol{eq}_i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_i}$
Inequality constraints:	$\boldsymbol{c}_{\boldsymbol{in}i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)$, $\left(\nabla_{\boldsymbol{x}_i}\boldsymbol{c}_{\boldsymbol{in}i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}i}$, $\left(\nabla_{\boldsymbol{u}_i}\boldsymbol{c}_{\boldsymbol{in}i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}i}$, $\left(\nabla_{\boldsymbol{\nu}_i}\boldsymbol{c}_{\boldsymbol{in}i}\left(\boldsymbol{x}_i,\boldsymbol{u}_i\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}i}$
"Coupling": coupling between agent i and its neighbor agent j	
Costs:	$L_{ij}\left(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{u}_i,\boldsymbol{u}_j\right)$, $\nabla_{\boldsymbol{x}_i} L_{ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)$, $\nabla_{\boldsymbol{x}_j} L_{ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)$, $\nabla_{\boldsymbol{u}_i} L_{ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)$, $\nabla_{\boldsymbol{u}_j} L_{ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)$
Final Costs:	$V_{ij}\left(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{u}_i,\boldsymbol{u}_j\right)$, $\nabla_{\boldsymbol{x}_i} V_{ij}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)$, $\nabla_{\boldsymbol{x}_j} V_{ij}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)$,
Equality Constraints:	$\boldsymbol{c}_{\boldsymbol{eq}_{ij}}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)$, $\left(\nabla_{\boldsymbol{x}_i}\boldsymbol{c}_{\boldsymbol{eq}_{ij}}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{ij}}$, $\left(\nabla_{\boldsymbol{x}_j}\boldsymbol{c}_{\boldsymbol{eq}_{ij}}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{ij}}$, $\left(\nabla_{\boldsymbol{u}_i}\boldsymbol{c}_{\boldsymbol{eq}_{ij}}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{ij}}$, $\left(\nabla_{\boldsymbol{u}_j}\boldsymbol{c}_{\boldsymbol{eq}_{ij}}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{ij}}$, $\left(\nabla_{\boldsymbol{\nu}_i}\boldsymbol{c}_{\boldsymbol{eq}_{ij}}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{ij}}$
Inequality Constraints:	$\boldsymbol{c}_{\boldsymbol{in}ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)$, $\left(\nabla_{\boldsymbol{x}_i}\boldsymbol{c}_{\boldsymbol{in}ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}ij}$, $\left(\nabla_{\boldsymbol{x}_j}\boldsymbol{c}_{\boldsymbol{in}ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}ij}$, $\left(\nabla_{\boldsymbol{u}_i}\boldsymbol{c}_{\boldsymbol{in}ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}ij}$, $\left(\nabla_{\boldsymbol{u}_j}\boldsymbol{c}_{\boldsymbol{in}ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}ij}$, $\left(\nabla_{\boldsymbol{\nu}_i}\boldsymbol{c}_{\boldsymbol{in}ij}\left(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{x}_j,\boldsymbol{u}_j\right)\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}ij}$

7.4 DENMPC framework implementation

To provide a flexible control approach for distributed systems, this thesis contributes the ”Distributed System Event-based Nonlinear Model Predictive Control“(*DENMPC*) framework.

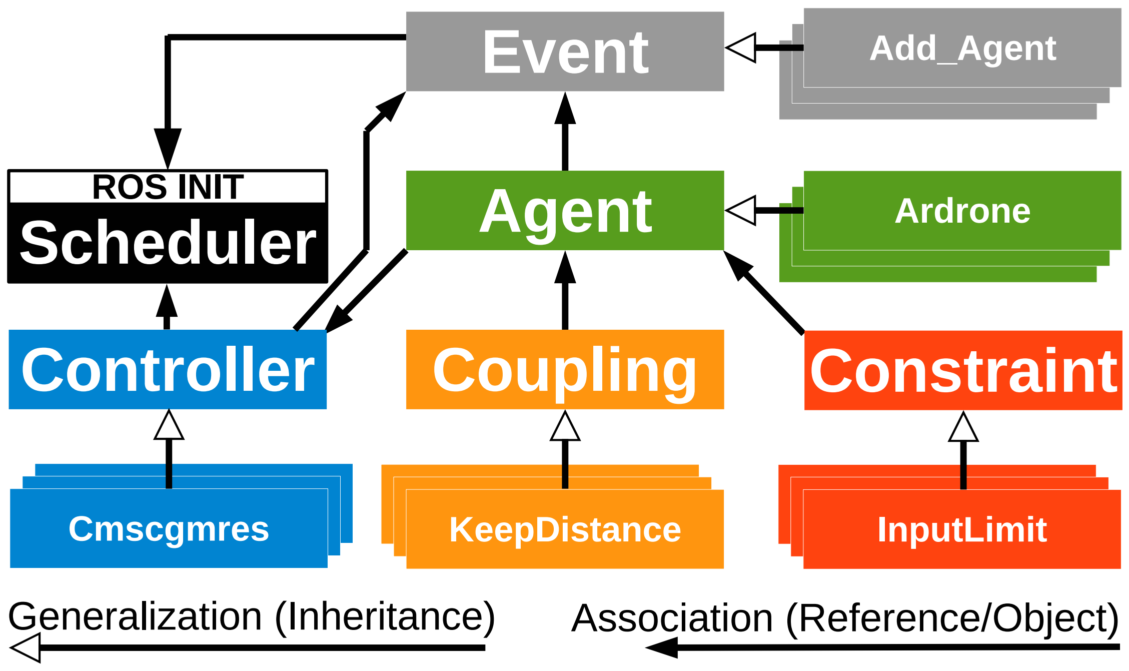


Figure 7.2: Schematic DENMPC class structure

The principle structure of the *DENMPC* package is shown in Figure 7.2. It consists of the five base class containers **Controller**, **Agent**, **Constraint**, **Coupling** and **Event** which provide the basic functionality for the user defined *MPC*. Each featured user agent, constraint, etc. is implemented as child class of these base class containers and therefore inherits the interface which is required for the modular composition of the *NMPC*. This includes the declaration of the atomic functions given in Table 7.1. The **Scheduler** class is handling the communication of the agents with the controllers.

The **Agent** class represents a generalization of user defined agents. It provides the interface for the atomic functions given in Table 7.1 to represent system dynamics, cost functions, constraints and their derivatives. Furthermore, each agent contains a list of pointers to additional constraints and/or couplings, which are acting upon it. Besides the declaration of the atomic functions for the modularization, the **Agent** class provides a *ROS* communication interface. This interface allows the subscription of measurement data and publication of controls.

According to the modularization, also the atomic functions of constraints and couplings can be structured in the base classes **Constraint** and **Coupling**. These contain interfaces for stage costs, final costs, equality and inequality constraints functions, as well as their derivatives. In contrast to the individual agent **Constraint**, the **Coupling** class functions is associated with two agents. This allows establishing interactions between agents and as a result, the extension to arbitrary multi-agent systems.

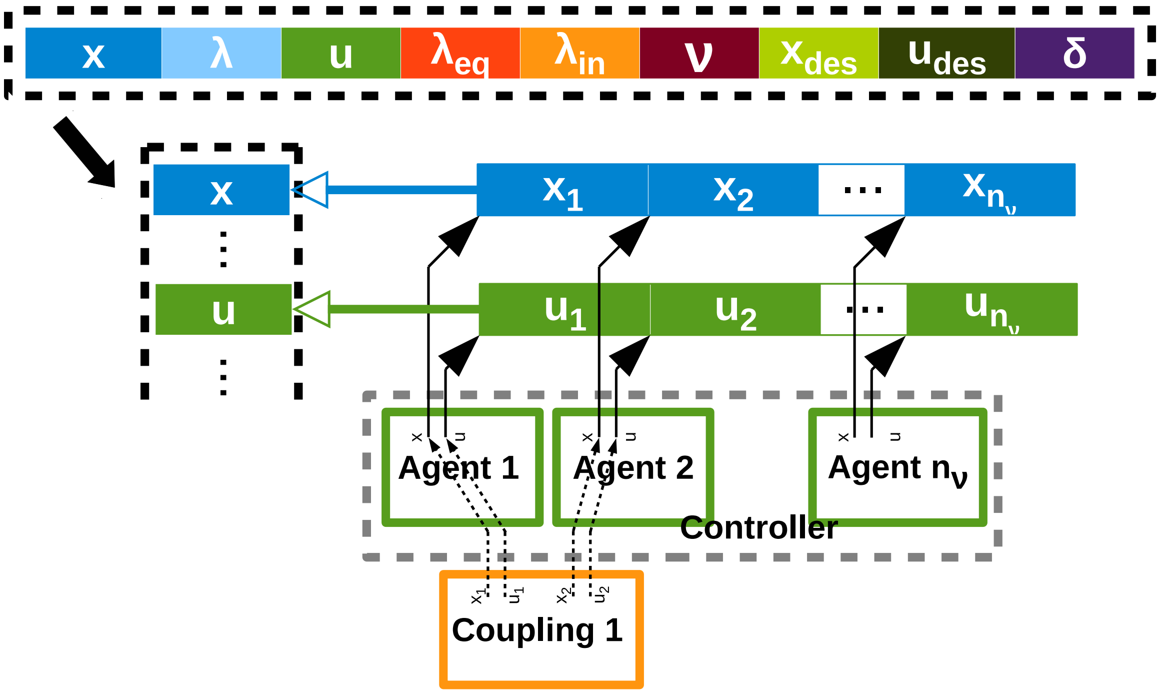


Figure 7.3: Schematic DENMPC memory and concatenation structure

The main function of the **Controller** base class is the modularization described in §7.3. This includes the concatenation of states and optimization variables (e.g. (7.8)). A schematic explanation of this concatenation is given in Figure 7.3. Based on the concatenation, the composition of the optimality conditions is implemented. The idea of concatenating all variables of each single agent, constraint and coupling in global vectors is to provide an interface for the implemented solvers. As a result, the solver only considers the solution of a single global *OCP* with global state and optimization variable vectors. This facilitates the implementation of solvers, as the modularization does not have to be taken into account. The global vectors are thereby depending on the scenario and the constraint handling method. For example, for the auxiliary variable method (see §A.2) the slack-variable and inequality Lagrange multiplier are implemented as additional optimization variables. All global vectors of states $\boldsymbol{x}$, optimization variables $\boldsymbol{w}$, controls $\boldsymbol{u}$, target states $\boldsymbol{x_{des}}$, target controls $\boldsymbol{u_{des}}$, parameters $\varkappa$, states Lagrange multipliers $\boldsymbol{\lambda}$, equality constraint Lagrange multipliers $\boldsymbol{\lambda_{eq}}$, inequality constraint Lagrange multipliers $\boldsymbol{\lambda_{in}}$, inequality constraint slack variables $\boldsymbol{\nu}$ are allocated in one memory block. The **Agent**, **Constraint** and **Coupling** objects store the information at which address their individual component $\boldsymbol{x}_1$ is stored within the global vectors. Accordingly, the controller contains a list of pointers to all agents that are controlled.

In order to minimize the computational overload introduced by the modularization, a pointer/functor access scheme is used. A schematic example of the pointer access scheme to the atomic functions is given in Table 7.1 for the composition of the global $\nabla_{\boldsymbol{x}} L$. To compose the global *OCP* functions, the controller iterates through all controlled agents (pointer *pag*), associated constraints (pointer *pcon*) as well as couplings (pointer *pcuo*) and sums up their individual influence. The computational overhead is limited to the functor/pointer access times and the iteration management overhead as well as the allocation of auxiliary variables. According to this access scheme, the complete optimality conditions (7.11)-(7.13) are composed and then provided to the *CMSC/GMRES* solver.

Listing 7.1: Schematic pointer access scheme for the example of composing $\nabla_{\boldsymbol{x}} L$

```
//Determine the global stage cost state derivative
void Controller::dldx(out,t,u,x){
   //Loop over agents
   for(int i=0;i<this->agentlist.size();i++){
      //Get pointer to agent
      pag=agentlist_[i];
      //Get dldx values of agent
      pag->dldx(tmp,t,pag->u,pag->x);
      //Add the agent related elements of tmp to out
      add(out,tmp);
      //Loop over constraints in agents
      for(int j=0;j<pag->getConstraint_Dim();j++){
         //Get constraint pointer
         pcon=pag->constraint_[j];
         //Get dL/dx values of constraint
         pcon->dldx(tmp,t,pcon->u,pcon->x)
         //Add the constraint's agent elements of tmp to out
         add(out,tmp);
      }
      //Loop over couplings in agents
      for(int k=0;k<pag->getCoupling_Dim();k++){
         //Getting coupling pointer
         pcou=pag->coupling_[k];
         //Get dL/dx1 values of coupling
         pcou->dldx1(tmp,t,pcou->u1,pcou->x1,pcou->u2,pcou->x2);
         //Add the coupling's 1st agent elements of tmp to out
         add(out,tmp);
         //Get dL/dx2 values of coupling
         pcou->dldx2(tmp,t,pcou->u1,pcou->x1,pcou->u2,pcou->x2);
         //Add the coupling's 2nd agent elements of tmp to out
         add(out,tmp);
      }
   }
}
```

Besides the computational efficiency, the pointer based modularization allows a manipulation of the complete scenario by just modifying the pointer references. Agent n_ν can be easily exchanged by referring the pointer to agent i in the agent vector (*agentlist* in Listing 7.1) to another agent instance. This is also valid for control objectives, constraints, etc. To give an example, two different couplings have been predefined:

- follow drone with fixed distance
- hover at position.

Each of these objectives is represented by a set of the atomic functions given in Table 7.1. An array of coupling pointers is storing the couplings used for the composition of the central OCP. First the "hover at position" coupling is applied on a quadrotor by adding the pointer to the "hover at position" coupling to the list of couplings. Next, the "follow drone with fix distance" objective shall be applied. Accordingly, the previous pointer to "hover at position" is substituted by a pointer to the "follow drone with fix distance" element. An advantage of the proposed modularization is that without equation simplifications in compile-time the composed functions are mathematically identical to the nonmodular formulation. If the additional calculation overhead is neglectable, the controllability as well as stability of the controlled system is not affected and directly determined by system and solver properties,

To make use of this modularization, the **Event** class handles the event-triggered online addition and removal of agents, constraints and couplings to the global OCP. Furthermore, it allows the online modification of parameters. The purpose of this is for example the online adjustment of the state tracking penalty factors, parameters of the agent dynamics or the distances in formation coupling constraints. The appearance of such an event is checked at the beginning of each control update and can be caused by *ROS* communication events, timers, etc. As a result the *OCP* can be modified dynamically. Further information about the implementation of *DENMPC* is given in the code documentation within the package [Den16].

7.4.1 Code generation

In order to facilitate prototyping and the control development process, the derivatives and accordingly the atomic terms listed in Table 7.1 can be determined automatically from the *OCP* equations listed in Table 7.2. For example, *ACADO* is addressing this by automatic differentiation. However, to avoid the implementational complexity of automatic differentiation, a separate code-generator is used in the context of this thesis. This code-generator has been implemented as *Mathematica* package which allows the utilization of *Mathematica*'s symbolic calculation (e.g. derivation) capability. Accordingly, the agents, constraints and couplings and their related *OCP* equations listed in Table 7.2, are implemented in *Mathematica* and subsequently translated into *C++* code. This code-generator is currently under extensive testing and will be published under GPL3 open-source licence.

Table 7.2: Functionals and functions required for the code generation

“Agent” i
Costs: $L_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$, Final costs: $V_i(\boldsymbol{x}_i)$, Dynamics: $\boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$, Equality constraints: $\boldsymbol{c}_{\boldsymbol{eq}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i)$, Inequality constraints: $\boldsymbol{c}_{\boldsymbol{in}i}(\boldsymbol{x}_i, \boldsymbol{u}_i)$
“Coupling”: coupling between agent i and its neighbor agent j
Costs: $L_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{u}_i, \boldsymbol{u}_j)$, Final Costs: $V_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{u}_i, \boldsymbol{u}_j)$, Equality Constraints: $\boldsymbol{c}_{\boldsymbol{eq}_{ij}}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_j, \boldsymbol{u}_j)$, Inequality Constraints: $\boldsymbol{c}_{\boldsymbol{in}ij}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_j, \boldsymbol{u}_j)$,
“Constraint”: additional constraints / cost functions for agent i
Costs: $L_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$, Final costs: $V_i(\boldsymbol{x}_i)$, Equality constraints: $\boldsymbol{c}_{\boldsymbol{eq}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i)$, Inequality constraints: $\boldsymbol{c}_{\boldsymbol{in}i}(\boldsymbol{x}_i, \boldsymbol{u}_i)$

7.5 Validation of DENMPC

In order to validate the features of *DENMPC*, this chapter is providing experimental results regarding the runtime adaptability and computational efficiency of the proposed approach. For this purpose four experiments are conducted. The first scenario in §7.5.1 is comparing *DENMPC* with an explicit implementation of the global

OCP without modularization. For this purpose, a formation flight scenario of two *AR.Drone 2.0*s is conducted. The second scenario in §7.5.2 is an extension of the task-based tracking introduced in §5.6.7. In this scenario, one *AR.Drone 2.0* is tracking another *AR.Drone 2.0* using sensor constraints. This scenario shows the performance of the *DENMPC* framework for more complex tasks and unidirectional couplings. In §7.5.3, the third scenario is showing how one *AR.Drone 2.0* is dynamically added to a formation of two *AR.Drone 2.0*s. This scenario demonstrates the runtime adaptability. Finally, §7.5.4 is showing a heterogeneous cooperative control scenario in the form of cooperative aerial manipulation. In this scenario, an *AR.Drone 2.0* is tracking the end effector position of a *DJI M100 MUAV*. In order to demonstrate the variability of *DENMPC*, the dynamics, constraints, cost functions, constraint handling methods are altered in all four scenarios. The utilized hardware setup is given in §A.1. In the following section the computational efficiency of the modular composition method is validated in a *UAV* cooperation scenario.

7.5.1 Modularization benchmark

In this section, a simple formation scenario of two quadrotors is controlled with *NMPC*. The quadrotors are implemented as 3D-models in the simulation environment *V-REP*. In order to validate the computational efficiency of the modularization approach with fast pointer/functor access, the approach is compared with an explicit implementation of the *OCP*. The idea of the scenario is to track a given target position with UAV_1, while keeping a distance of $d_{des} = 1\,\mathrm{m}$ between both *UAV*s. This coupling is realized with the potential function

$$L_{21}\left(\boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{x}_2, \boldsymbol{u}_2\right) = q_d \sqrt{\left(\overrightarrow{\boldsymbol{p}}_1 - \overrightarrow{\boldsymbol{p}}_2\right)^{\mathrm{T}} \left(\overrightarrow{\boldsymbol{p}}_1 - \overrightarrow{\boldsymbol{p}}_2\right) - d_{des}}. \tag{7.18}$$

$\overrightarrow{\boldsymbol{p}}_1$, $\overrightarrow{\boldsymbol{p}}_2$ are the position vector of UAV_1, respectively UAV_2.

The applied *OCP* of the scenario is given as

$$\min_{\underline{\boldsymbol{u}}_1,\underline{\boldsymbol{u}}_2} J = \int_t^{t+T} \left(\boldsymbol{x}_{\boldsymbol{des}1} - \underline{\boldsymbol{x}}_1\right)^{\mathrm{T}} \boldsymbol{Q}_1 \left(\boldsymbol{x}_{\boldsymbol{des}1} - \underline{\boldsymbol{x}}_1\right) + \underline{\boldsymbol{u}}_1^{\mathrm{T}} \boldsymbol{R}_1 \underline{\boldsymbol{u}}_1 \tag{7.19}$$

$$+ \left(\boldsymbol{x}_{\boldsymbol{des}2} - \underline{\boldsymbol{x}}_2\right)^{\mathrm{T}} \boldsymbol{Q}_2 \left(\boldsymbol{x}_{\boldsymbol{des}2} - \underline{\boldsymbol{x}}_2\right) + \underline{\boldsymbol{u}}_2^{\mathrm{T}} \boldsymbol{R}_2 \underline{\boldsymbol{u}}_2$$

$$+ q_d \sqrt{\left(\overrightarrow{\boldsymbol{p}}_1 - \overrightarrow{\boldsymbol{p}}_2\right)^{\mathrm{T}} \left(\overrightarrow{\boldsymbol{p}}_1 - \overrightarrow{\boldsymbol{p}}_2\right) - d_{des}}\ \mathrm{d}\tau$$

s.t.

$$\boldsymbol{f}\left(\boldsymbol{x}_i, \boldsymbol{u}_i\right) = \begin{bmatrix} {}^{\mathcal{V}1}\dot{x}_i \cos\left({}^{\mathcal{V}}\Psi_i\right) - {}^{\mathcal{V}1}\dot{y}_i \sin\left({}^{\mathcal{V}}\Psi_i\right) \\ {}^{\mathcal{V}1}\dot{x}_i \sin\left({}^{\mathcal{V}}\Psi_i\right) + {}^{\mathcal{V}1}\dot{y}_i \cos\left({}^{\mathcal{V}}\Psi_i\right) \\ 0.8827 \cdot u_{\mathcal{V}1_{z,i}} \\ -0.005879 \cdot {}^{\mathcal{V}}\Psi_i + 1.265 \cdot u_{\mathcal{V}1_{\omega,i}} \\ -0.8799 \cdot {}^{\mathcal{V}1}\dot{x}_i + 3.273 \cdot u_{\mathcal{V}1_{x,i}} \\ -0.5092 \cdot {}^{\mathcal{V}1}\dot{y}_i + 1.458 \cdot u_{\mathcal{V}1_{y,i}} \end{bmatrix} \quad i = 1, 2 \tag{7.20}$$

$$\boldsymbol{0} \leq \boldsymbol{c}_{\boldsymbol{ini}} = \left[\underline{u}_{\mathcal{V}1_{x,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{y,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{z,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{\omega,i}}{}^2 - 1\right]^{\mathrm{T}} \tag{7.21}$$

$$\boldsymbol{x}_1(0) = \left[0, 0, 0, 0, 0, 0\right], \qquad \boldsymbol{x}_2(0) = \left[0, 1, 0, 0, 0, 0\right] \tag{7.22}$$

$$\boldsymbol{Q}_1 = \mathrm{diag}\left(\left[1, 1, 2, 1, 0, 0\right]\right), \qquad \boldsymbol{Q}_2 = \mathrm{diag}\left(\left[0, 0, 2, 1, 0, 0\right]\right) \tag{7.23}$$

$$\boldsymbol{R}_1 = \mathrm{diag}\left(\left[10, 10, 10, 10\right]\right), \qquad \boldsymbol{R}_2 = \mathrm{diag}\left(\left[10, 10, 10, 10\right]\right) \tag{7.24}$$

$$q_d = 10,\ d_{des} = 1, \qquad \underline{\boldsymbol{\lambda}_{\boldsymbol{eq}}}_i(0) = \left[1, 1, 1, 1\right] \cdot 10^{-4} \tag{7.25}$$

$$\boldsymbol{\nu}_i(0) = \left[0.9, 0.9, 0.9, 0.9\right], \quad r_{\nu,i}(0) = \left[1, 1, 1, 1\right] \cdot 10^{-4} \tag{7.26}$$

$$\xi = 1,\ \upsilon = 1,\ \Delta t = 0.1, T = 1, \tag{7.27}$$

$$N = 10,\ h = 0.001,\ \epsilon = 10^{-8},\ i_{max} = 30 \tag{7.28}$$

For ease of notation the time dependency of variables and functions are not explicitly shown in (7.19)-(7.21). The position tracking is realized in the *OCP* with a quadratic penalty (7.23)-(7.24) in the cost function (7.19). $\boldsymbol{Q}_2$ is hereby used to track the z-axis of UAV_2 to force it onto the xy-plane. The *UAV* system dynamics are represented by the *RHMY* with the parametrization as shown in (7.20).

The control limitation constraints (7.21) are realized via the auxiliary variable method (§4.5.2). The corresponding initialization of the Lagrange multipliers $\boldsymbol{\lambda}_{\boldsymbol{eq}}$, the slack variables $\boldsymbol{\nu}$ and the slack penalty $\boldsymbol{r}_\nu$ is given in (7.25)-(7.26). The applied *CMSC/GMRES* parameters are given in (7.27)-(7.28) according to the convention given in §5.3. The system is initialized with the initial states (7.22) which fulfill

$d_{des} = 1$. To analyze the systems closed-loop behavior, a position displacement of target $\boldsymbol{x_{des1}}$ is introduced at time $t \approx 10\,\mathrm{s}$

$$\boldsymbol{x_{des1}}(0) = [0,0,0,0,0,0]\,, \quad \boldsymbol{x_{des1}}(\approx 10) = [1,0,0,0,0,0]\,. \tag{7.29}$$

Figure 7.4-7.5 is showing the comparison of the modular *DENMPC* implementation with the nonmodular explicit implementation. The xy-plot in Figure 7.4 confirms the desired behavior. UAV_1 is reaching the target position which is indicated with a red rectangle. Both UAVs are trying to keep the distance $d_{des} = 1\,\mathrm{m}$. This is visible by the displacement of UAV_2 and the curvature of the UAV_1 trajectory. Due to the trajectory tracking, both UAVs do not show a significant deviation from the tracked $z = 0$. Hence, the visualization of the z-axis is omitted here. This behavior is further validated by the state and distance plots in Figure 7.5. Subsequent to the imposed change of $\boldsymbol{x_{des1}}$, the distance is converging towards the desired value $d_{des} = 1\,\mathrm{m}$. In correspondance, the actuation converges towards zero.

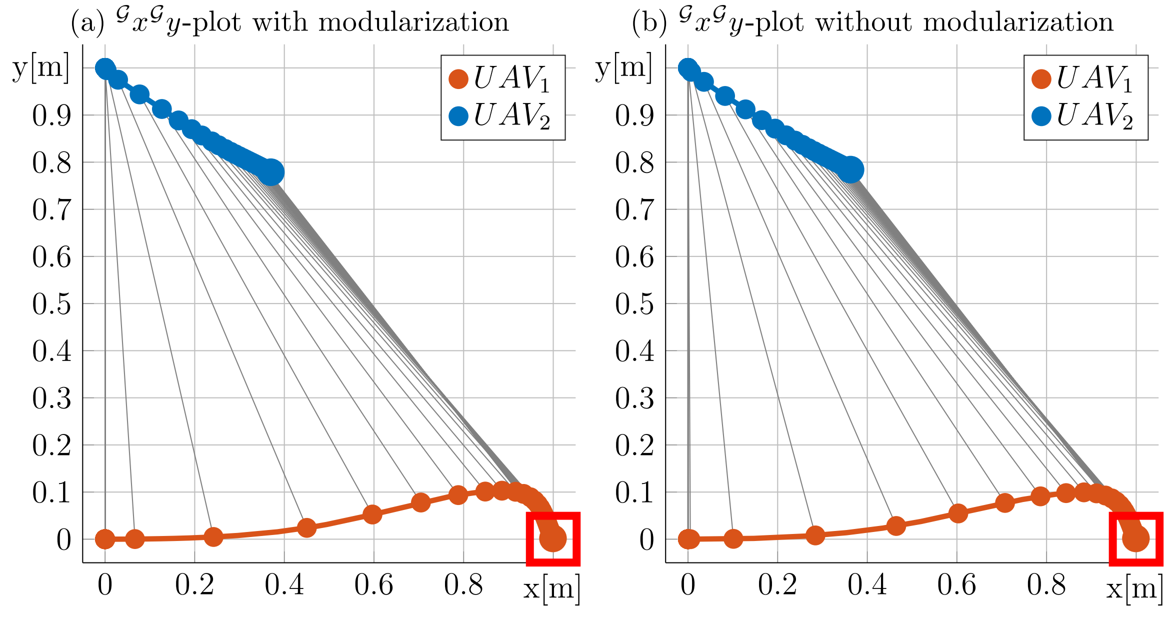


Figure 7.4: Comparison with (left) and without (right) modularization: UAV_1 and UAV_2 keep distance $d_{des} = 1$ while UAV_1 is tracking target

The trajectories of the modular *DENMPC* and nonmodular implementation in Figure 7.4-7.5 show identical behavior. This is expected, as the modular global *OCP* and nonmodular global *OCP* are mathematically identical. Small differences in the trajectories are caused by numerical and timing errors of the simulator as well as the interrupts of the running control process on the computer. These interrupts are

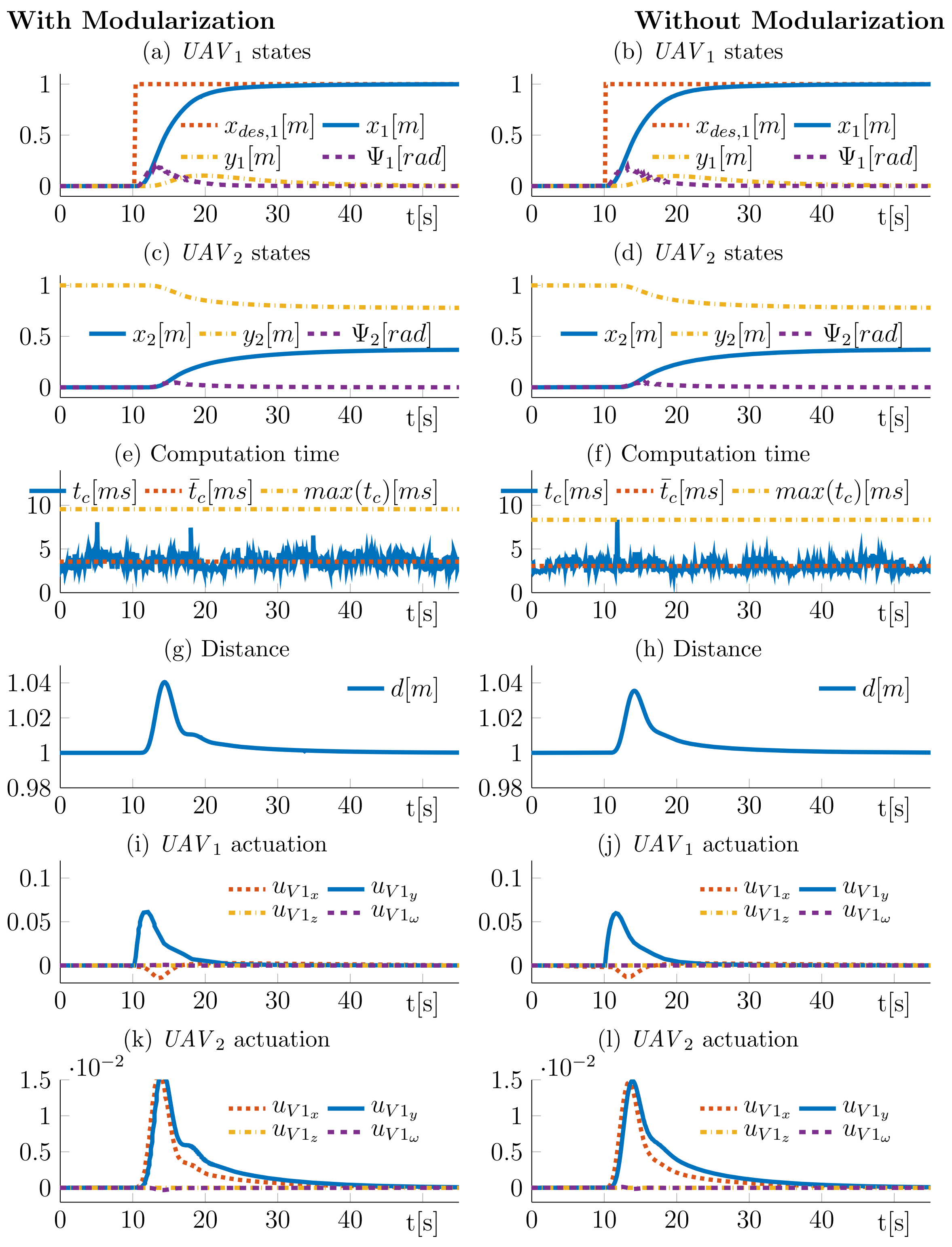


Figure 7.5: Comparison with and without modularization UAV_1 and UAV_2 keep distance $d_{des} = 1$ and UAV_1 is tracking target

also assumed to lead to the peak values of the computation time $max(t_c) \leq 10\,\text{ms}$ in Figure 7.5, as these peaks appear arbitrarily and without correlation to the system trajectory (harware specifications in A.1). Nevertheless, the computation time does not exceed $\Delta t = 100\,\text{ms}$ which validates the real-time applicability. This is stating the efficiency of the applied *CMSC/GMRES* method in order to cooperatively control a small swarm of *UAV*s. In reverse conclusion, this setup would allow the real-time control of up to 10 similar scenarios, controlled by a single computer The computational efficiency of the proposed modularization is evaluated with the mean computation time. The modular approach with $\bar{t}_{comp} = 3.551\,\text{ms}$ shows only a computational overhead of 16% to $\bar{t}_{comp} = 3.057\,\text{ms}$ of the nonmodular *OCP*. This remarkably low computational overhead comes with the advantage of online adaptability of dynamics, objectives and topology and confirms the efficiency of the proposed modularization. For more complex agent dynamics, this ratio further declines as the number of functor/pointer access would stay the same, while the effective time within the functions would increase.

7.5.2 Unidirectional sensor-constrained tracking

In order to analyze the *DENMPC* performance in a more complex real multi-*UAV* scenario with unidirectional coupling, this section is extending the task-based *MPC* validation scenario presented in §5.6.7. Here, one quadrotor (UAV_1) is equipped with a camera to track the target quadrotor UAV_2. Hence, the task is to keep the target quadrotor in the camera frame, as shown in Figure 7.6.

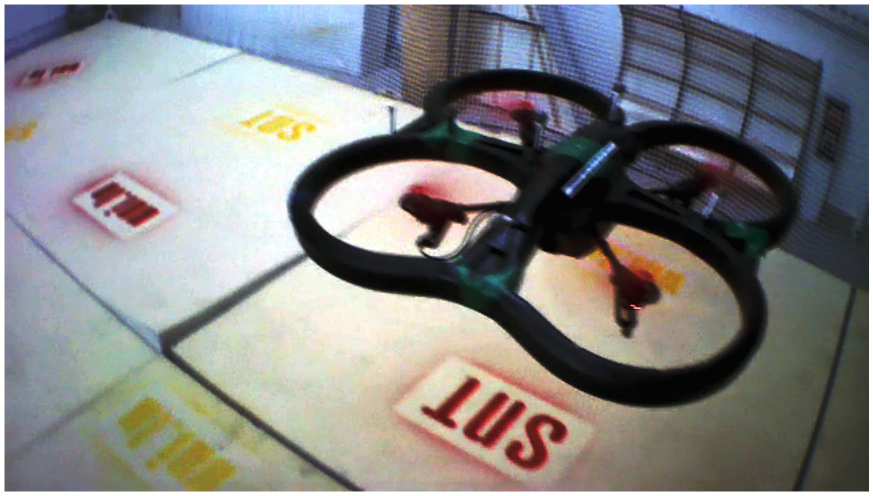


Figure 7.6: Cooperative sensor-constrained tracking scenario: Visual tracking of quadrotor from camera equipped quadrotor

The control of both quadrotors is accomplished with *DENMPC* using the potential functions for sensor-based tracking as developed in §5.6. The quadratic state tracking

$$L_i := \left(\underline{\boldsymbol{x}}_{\boldsymbol{des}_i}(\tau) - \underline{\boldsymbol{x}}_i(\tau)\right)^{\top} \boldsymbol{Q}_i \left(\underline{\boldsymbol{x}}_{\boldsymbol{des}_i}(\tau) - \boldsymbol{x}_i(\tau)\right) + \underline{\boldsymbol{u}}_i(\tau) \boldsymbol{R}_i \underline{\boldsymbol{u}}_i(\tau), \tag{7.30}$$

sensor cone constraint (5.72), coordinate transformation (5.74)

$$\begin{bmatrix} {}^{\mathcal{S}}\underline{\overrightarrow{\boldsymbol{p}}}_1 \\ 1 \end{bmatrix} = {}^{\mathcal{S}}\boldsymbol{T}_{\mathcal{G}} \left(\beta_{FoV}, -d_s, -{}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}_0\right) \begin{bmatrix} {}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}_1 \\ 1 \end{bmatrix} \tag{7.31}$$

in combination with the *CA* (5.80), cohesion (5.86) constraint and quadrotor dynamics *RHMY* (3.27) with parameters (3.29), the *OCP* yields

$$\begin{aligned} \min_{\underline{\boldsymbol{u}}_1, \underline{\boldsymbol{u}}_2} J = & \int_t^{t+T} \sum_{i=1}^{2} \underline{L_i} \\ & + \kappa_{H0} L_{c_{in}}{}^{\backslash\{\underline{\boldsymbol{u}}_2\}} \left({}^{\mathcal{S}}\underline{\overrightarrow{\boldsymbol{p}}}_1, \alpha_{FoV}, \kappa_{G0}, \kappa_{A0}\right) \\ & + \kappa_{H1} L_{maxD}^{\backslash\{\underline{\boldsymbol{u}}_2\}} \left({}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}_1, {}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}_1, d_{rotmax}, \kappa_{H1}, \kappa_{G1}, \kappa_{A1}\right) \\ & + \kappa_{H2} L_{minD}^{\backslash\{\underline{\boldsymbol{u}}_2\}} \left({}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}_1, {}^{\mathcal{G}}\underline{\overrightarrow{\boldsymbol{p}}}_1, d_{rotmin}, \kappa_{H2}, \kappa_{G2}, \kappa_{A2}\right) \mathrm{d}\tau \\ \text{s.t. } & \boldsymbol{0} \leq \boldsymbol{c_{in}}_i = \left[\underline{u}_{\mathcal{V}1_{x,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{y,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{z,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{\omega,i}}{}^2 - 1\right] \\ & \boldsymbol{0} = \boldsymbol{f}_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i, t\right) \end{aligned} \tag{7.32}$$

As introduced in §7.2, the index $\backslash\{\boldsymbol{u}_2\}$ of the cost functions $L_{c_{in}}$, L_{maxD} and L_{minD} indicates, that the influence of the cost functions on UAV_2 is neglected. This means that the sensor cone, *CA* and cohesion constraints are only affecting UAV_1. The advantage of this is, the future states of both UAVs are considered for the computation of the optimal controls. However, the effect of the tracking costs can be limited to one quadrotor. The parameters of (7.32) for the experimental validation are given as

$$\boldsymbol{Q}_1 = \text{diag}\left(\left[0,0,0,0,0,0.7,0.7\right]\right), \quad \boldsymbol{R}_1 \; = \; \text{diag}\left(\left[1,1,1,1\right]\right) \tag{7.33}$$

$$\boldsymbol{Q}_2 = \text{diag}\left(\left[1.5,1.5,1.6,1,1,0,0\right]\right), \boldsymbol{R}_2 \; = \; \text{diag}\left(\left[1,1,1,1\right]\right) \tag{7.34}$$

$$q_{c_{in},1} = \left[1,1,1,1\right] \cdot 10^{-4}, q_{c_{in},2} = \left[1,1,1,1\right] \cdot 10^{-4} \tag{7.35}$$

$$\varkappa_c \; : \; d_s = 0.17, \alpha_{FoV} = 0.5, \beta_{FoV} = 0.5, \tag{7.36}$$

$$\kappa_{H0} = 0.4, \kappa_{G0} = 0.01, \kappa_{A0} = 2.0$$

$$\varkappa_{minD} \; : \; d_{rotmin} = 1, \kappa_{H1} = 0.4, \kappa_{H1} = 4.5, \kappa_{G1} = 0.001, \kappa_{A1} = 3.0 \tag{7.37}$$

$$\varkappa_{maxD} \; : \; d_{rotmax} = 2, \kappa_{H2} = 0.4, \kappa_{H2} = 1.5, \kappa_{G2} = 0.001, \kappa_{A2} = 3.0 \tag{7.38}$$

$$\varkappa_{cmscgmres} \; : \; N = 20, T = 1\,\text{s}, \epsilon = 10^{-8}, \xi = 10, \Delta t = 0.01\,\text{s}, i_{max} = 30, \tag{7.39}$$

$$\upsilon = 2 \tag{7.40}$$

Here, $\varkappa_{cmscgmres}$ are the *CMSC/GMRES* solver parameters and given in a coherent notation to §5.3. The input limitation constraints are realized with primal barrier constraint handling (see §4.5.1) using (7.35).

To examine the dynamic behavior of the proposed control solution, the target position of UAV_2 is moving in a circular trajectory. For UAV_1, the choice of $\boldsymbol{Q}_1$ leads to a tracking of zero forward velocity ${}^{\mathcal{V}1}\dot{x}_1(t) = 0$ and sideward velocity ${}^{\mathcal{V}1}\dot{y}_1(t) = 0$ which yields the desired states

$$\boldsymbol{x}_{\boldsymbol{des}1} = \text{diag}\left(\left[0,0,0,0,0,0,0\right]\right) \tag{7.41}$$

$$\boldsymbol{x}_{\boldsymbol{des}2} = \text{diag}\left(\left[\tfrac{1}{2}\cos(0.3t), \tfrac{1}{2}\sin(0.3t), 1, 0, 0, 0.0, 0.0\right]\right). \tag{7.42}$$

For the numerical validation, two *AR.Drone 2.0* models have been implemented in the simulation environment *V-REP* (Figure 7.7a). In correspondence, Figure 7.7b shows the real *AR.Drone 2.0*s during the experimental validation. The *UAV* position data is thereby measured by a motion capture system.

An intuitive access to the *UAV* behavior is gained by plotting the *UAV* positions and the orientation of UAV_1 by means of a vector as shown in Figure 7.8b-7.8b. To be able to associate both *UAV* positions, time related *UAV* positions are connected with a line. It is visible that UAV_2 is following the desired circular trajectory, while UAV_1 is tracking UAV_2 in an ellipsoidal movement. The orientation vectors are displayed at each $\Delta t \approx 1.68\,\text{s}$ for means of visualization. Both *xy*-plots confirm the circular trajectory of UAV_2. This is also characterized by the sinusoidal movement in Figure

(a) Numerical validation in *V-REP* (b) Real *UAV* validation

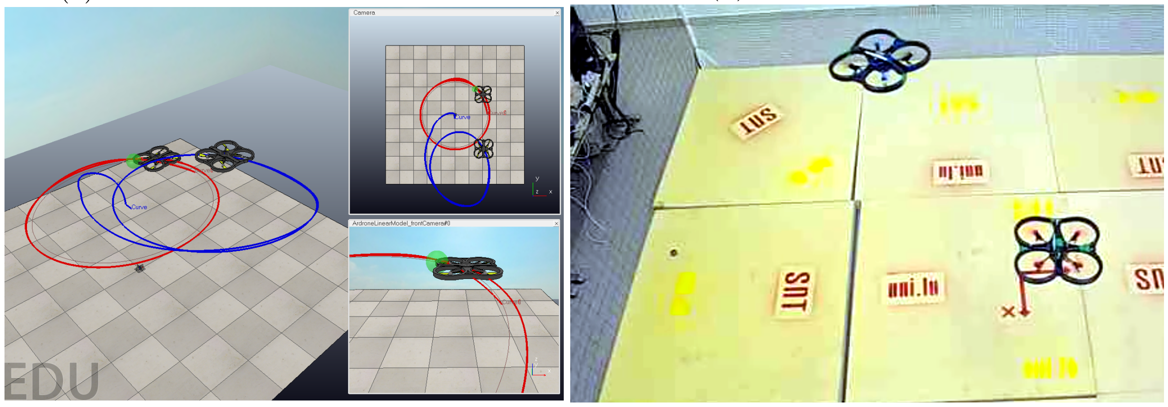


Figure 7.7: Cooperative sensor-constrained tracking scenario: setup

7.8e and Figure 7.8f. The likewise sinusoidal position trajectories of UAV_1 in Figure 7.8c are indicating a tracking of UAV_2. The position of UAV_1 is just dependent on the applied constraints. As a result, UAV_1 can rotate freely around UAV_2. This leads to the drift of the sinusoidal position trajectory of UAV_1. In contrast to the numerical validation, the initial *UAV* positions have been chosen arbitrarily in the real scenario. As a result, the position trajectory of UAV_1 shows a different motion pattern in Figure 7.8d.

Nevertheless, the distance plots in Figure 7.9a-7.9b state that UAV_2 is tracked within the given distance limitations $d_{min} \leq d \leq d_{max}$. An exception is the initial phase of the simulation. In this scenario, the repulsive behavior of the *CA* constraint causes an increase in the distance in order to satisfy $d \geq d_{min}$. At $t \approx 55\,\mathrm{s}$, the minimum distance constraint of the real *UAV* scenario is violated due to disturbance. Due to the soft implementation of the sensor constraint, the *OCP* maintains feasible. This allows the recovery from the constraint violation. To conclude, the distance trajectory is validating the active *CA* and cohesion constraint.

For the validation of the controller performance, Figure 7.9c-7.9d are showing the absolute tracking angle. According to (7.43), the tracking angle for the multi-*UAV* scenario yields

$$\alpha_t = \| \arccos(\frac{{}^{S}\overrightarrow{\boldsymbol{p}}_2 \cdot [1,0,0]^\top}{|{}^{S}\overrightarrow{\boldsymbol{p}}_2|}) \|. \tag{7.43}$$

As desired, the cone constraint keeps the tracking angle α_t smaller than the sensor beam width angle $\alpha_t \leq \alpha_{FoV} = 0.5\,\mathrm{rad}$. Figure 7.9c shows, that the tracking angle α_t in the simulation is in fact much smaller. This is caused by the smooth approximation

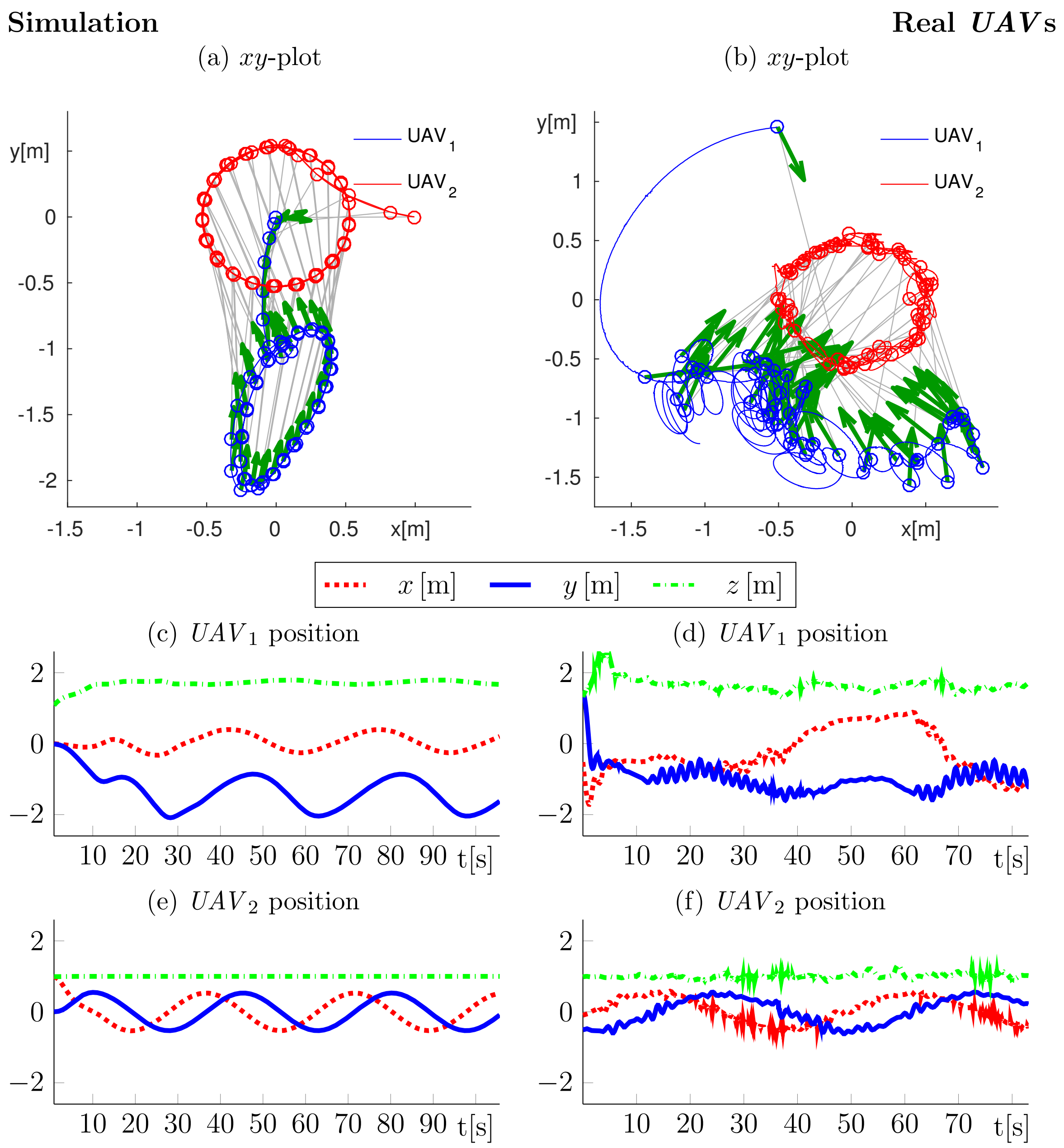


Figure 7.8: Cooperative sensor-constrained tracking scenario: position

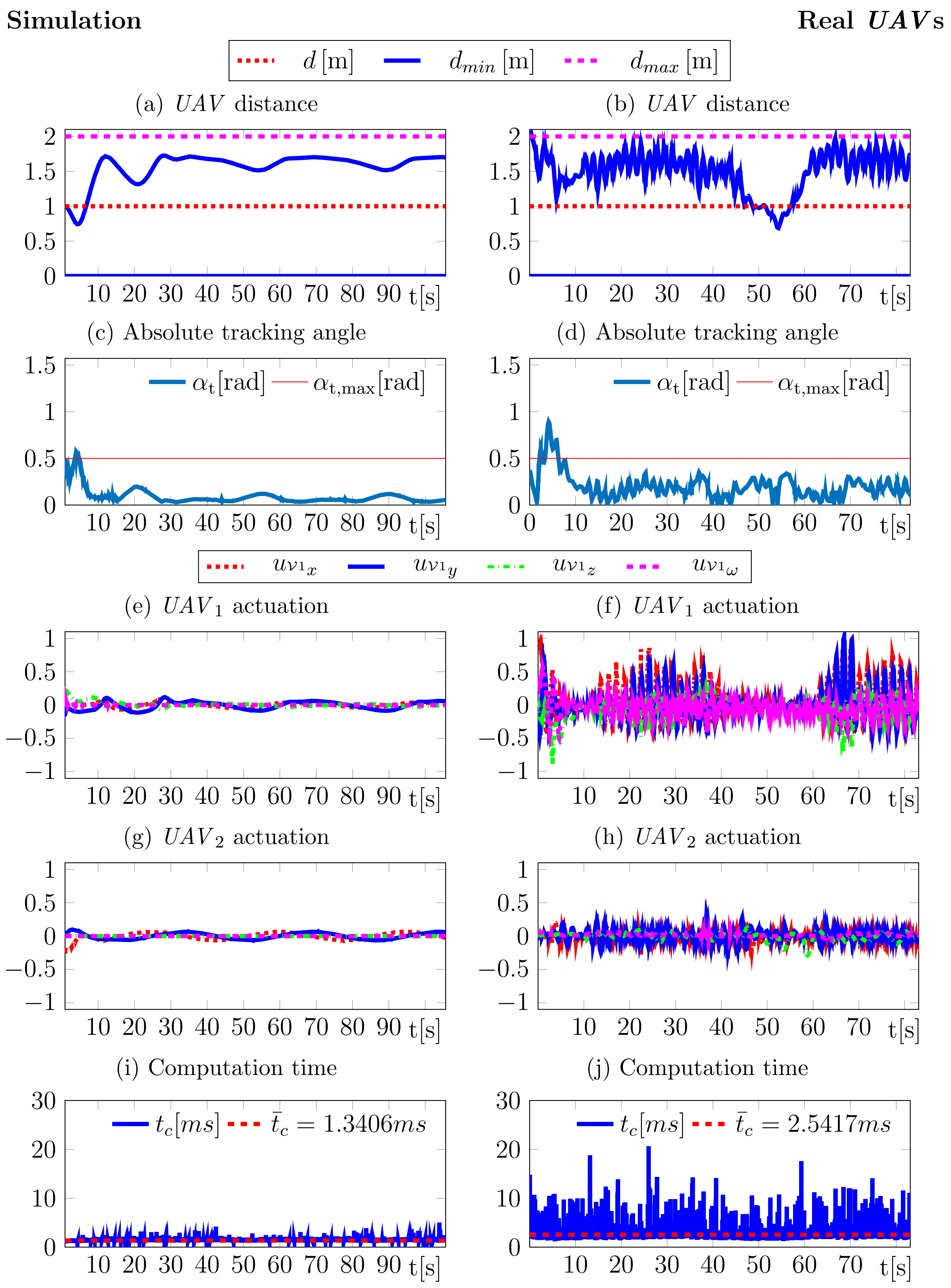


Figure 7.9: Cooperative sensor-constrained tracking scenario: data

of the unit steps by sigmoids (4.137). The result is a convergence towards the center of the cone. Especially for undisturbed systems, the tracking angle α_t is consequently significant smaller than the beam width angle α_{FoV}. To reduce this effect κ_A can be increased. The disturbed real *UAV*s exploit more of this tracking angle range. Due to the initial conditions, α_t is violating the applied cone constraint at the beginning. For $t \geq 10\,\mathrm{s}$, it is tracked within the given bounds $\alpha_t \leq \alpha_{FoV} = 0.5\,\mathrm{rad}$. Hence, the experimental results are validating the desired *NMPC* performance.

The control trajectories in Figure 7.9e-7.9h confirm that the input limits are respected. In contrast to the numerical solution, the real *UAV*s show a significant higher control action. This is caused by disturbance rejection and model errors.

To evaluate the computational efficiency of *DENMPC* for the given scenario, Figure 7.9i-7.9j are giving the *MPC* computation time on a *Dell Latitude E5440* (see §A.1). The real experimental evaluation requires additional drivers which leads to higher computational load. The results are CPU interrupts of the controller which cause sudden peaks in the computation time of $t_c \approx 20\,\mathrm{ms}$. Furthermore, the real *UAV* controller shows a higher computational load due to disturbance rejection. Nevertheless, the *DENMPC* average computation times $\bar{t_c} \approx 2.54\,\mathrm{ms}$, respectively $\bar{t_c} \approx 1.34\,\mathrm{ms}$, are very low in comparison with the control update interval of $\Delta t = 10\,\mathrm{ms}$. This states the real-time applicability of *DENMPC* for this scenario.

7.5.3 Online adaptability

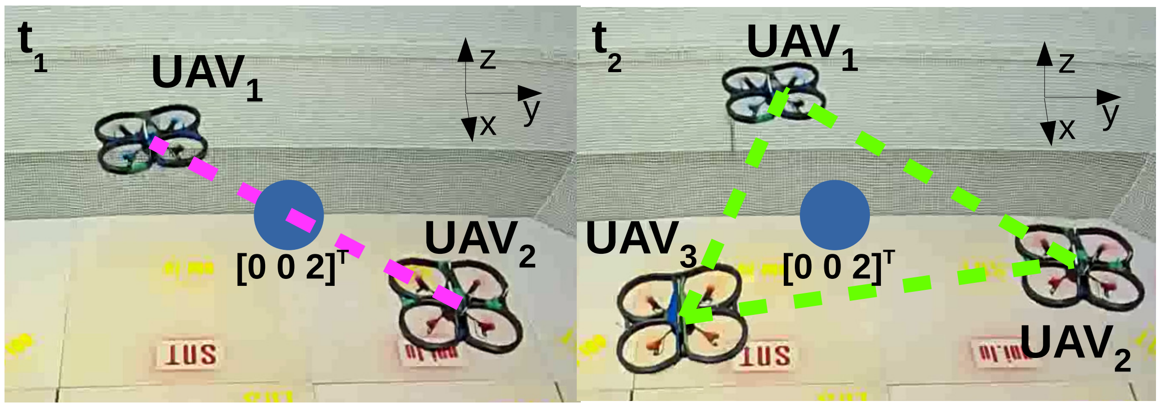


Figure 7.10: Runtime formation change scenario: Video footage: t_1: 2 agents (left) vs. t_2:3 agents (right)

This section demonstrates runtime changes of agents, constraints and couplings with *DENMPC*. To visualize this adaptability, a switch between two formations is conducted online. Initially two *AR.Drone 2.0* quadrotors form a linear formation around a given target. Subsequently, a third *AR.Drone 2.0* is dynamically added to the scenario. This causes the formation to change to a triangle around the target, as shown in Figure 7.10. This formation change is caused by a combination of position tracking and *CA* constraints. For this purpose each agent is tracking the same point $\vec{\boldsymbol{p}}_{des} = \left[0, 0, 2\right]^\top$ which is realized with $\boldsymbol{x_{des}} = \left[0, 0, 2, 0, 0, 0\right]^\top$ and minimal energy consumption $\boldsymbol{u_{des}} = \boldsymbol{0}$. This is realized for each *UAV* i with standard quadratic state tracking stage costs:

$$L_i\left(\boldsymbol{x}_i\left(t\right), \boldsymbol{u}_i\left(t\right)\right) = \left(\boldsymbol{x_{des}} - \boldsymbol{x}_i\left(t\right)\right)^\top \boldsymbol{Q}\left(\boldsymbol{x_{des}} - \boldsymbol{x}_i\left(t\right)\right) + \boldsymbol{u}_i\left(t\right)^\top \boldsymbol{R}\boldsymbol{u}_i\left(t\right). \quad (7.44)$$

In order to keep a minimum distance d_{min}, a *CA*-constraint is implemented between each pair of *UAV*s using the saturation function constraint handling (see §4.5.3)

$$L_{i,j}\left(\vec{\boldsymbol{p}}_i\left(t\right), \vec{\boldsymbol{p}}_j\left(t\right)\right) = \frac{\kappa_H}{1 + e^{-\kappa_A\left(d_{min}{}^2 - \left(\vec{\boldsymbol{p}}_i(t) - \vec{\boldsymbol{p}}_j(t)\right)^\top\left(\vec{\boldsymbol{p}}_i(t) - \vec{\boldsymbol{p}}_j(t)\right)\right)}}. \quad (7.45)$$

The goal of keeping a minimum distance of $d_{min} = 1.5\,\mathrm{m}$ while targeting the same position is contradicting. For two and three *UAV*s, the result are equidistant formations around the given target position.

The related *OCP*s are formed with (7.44), (7.45), the *RHMY* (3.27) with parameter set (3.29) and the control constraints $\|u\| \leq 1$. Starting with two agents (UAV_1, UAV_2) this results in the *OCP*

$$\begin{aligned} \min_{\boldsymbol{u}}\ J = & \int_t^{t+T} \sum_{i=1}^{2} L_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) + L_{1,2}\left(\underline{\vec{\boldsymbol{p}}}_1, \underline{\vec{\boldsymbol{p}}}_2\right) \ \mathrm{d}\tau \\ \text{s.t. } & \boldsymbol{0} \leq \boldsymbol{c_{in}}_i = \left[\underline{u}_{\mathcal{V}1_{x,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{y,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{z,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{\omega,i}}{}^2 - 1\right], \\ & \boldsymbol{0} = \boldsymbol{f}_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) \end{aligned} \quad (7.46)$$

For means of simplicity, the initial states are not shown in the *OCP* formulation. It shall be mentioned that the *UAV* positions (e.g. $\underline{\vec{\boldsymbol{p}}}_1$) are part of the state vectors (e.g. $\boldsymbol{x}_1$).

At time t_1, a *ROS* message is triggering a change of the *OCP*. As a result, an additional quadrotor and the corresponding constraints ($\boldsymbol{c_{in}}_3$) and *CA* couplings

$(L_{1,3}, L_{2,3})$ are added at runtime to the scenario. The *OCP* for three *UAV*s yields

$$\begin{aligned}\min_{\boldsymbol{u}}\ J = \int\limits_{t}^{t+T} & \sum_{i=1}^{3} L_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) + L_{1,2}\left(\underline{\overrightarrow{\boldsymbol{p}}}_1, \underline{\overrightarrow{\boldsymbol{p}}}_2\right) \\ & + L_{2,3}\left(\underline{\overrightarrow{\boldsymbol{p}}}_2, \underline{\overrightarrow{\boldsymbol{p}}}_3\right) + L_{1,3}\left(\underline{\overrightarrow{\boldsymbol{p}}}_1, \underline{\overrightarrow{\boldsymbol{p}}}_3\right)\ \mathrm{d}\tau \\ \text{s.t.}\ \boldsymbol{0} \leq \boldsymbol{c_{in}}_i = & \left[\underline{u}_{\mathcal{V}1_{x,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{y,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{z,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{\omega,i}}{}^2 - 1\right] \\ \boldsymbol{0} = & \boldsymbol{f}_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right).\end{aligned} \tag{7.47}$$

Finally, at time instance t_2, a second *ROS* message triggers the removal of UAV_3. Consequently, the *OCP* is switching again from form (7.47) to form (7.46).

To solve the given problem, *DENMPC* is using *CMSC/GMRES* with primal barrier constraint handling. The scenario parameters have been determined empirically to achieve a smooth system response

$$\boldsymbol{Q}_i = \operatorname{diag}\left(\left[2, 2, 8, 3, 10.5, 10.5\right]\right) \tag{7.48}$$

$$\boldsymbol{R}_i = \operatorname{diag}\left(\left[5.5, 5.5, 3, 3.1\right]\right) \tag{7.49}$$

$$\varkappa_{CA}\ :\ d_{min} = 1.5\,\mathrm{m}, \kappa_H = 4.0, \kappa_A = 2.0 \tag{7.50}$$

$$\varkappa_{cmscgmres}\ :\ N = 10, T = 3\,\mathrm{s}, h = 0.001\,\mathrm{s}, \xi = 10, \Delta t = 0.01\,\mathrm{s}, \tag{7.51}$$

$$i_{max} = 6, \upsilon = 2, \epsilon = 0.1. \tag{7.52}$$

For ease of visualization, the movement is limited to the xy-plane using a high z-axis penalty. The result is shown in Figure 7.10. The trajectory of the agents in 3D space is given in Figure 7.11. Each agent and the central target are depicted as a small sphere. The distance between corresponding agents at each time instance is visualized as gray line. While the linear formation (7.46) is emphasized by a magenta colored connection line, the green connection lines are showing the triangular formation according to *OCP* (7.47). Before t_1, UAV_3 is not within the bounds of the plotted region, as indicated by the distance connectors in Figure 7.11. This is validating that *DENMPC* is able to switch its control behavior online.

Triggered by a *ROS* message at time $t_1 \approx 8\,\mathrm{s}$, UAV_3 is added to *OCP* (7.46). This results to *OCP* (7.47). Figure 7.11 shows how the formation is then shifted from a linear to a triangular formation between t_1 and t_2. As all *UAV*s are tracking the same point and the *CA* is only considering the Euclidean distance of the agents, the formation can rotate freely around the tracked center point. A second *ROS*

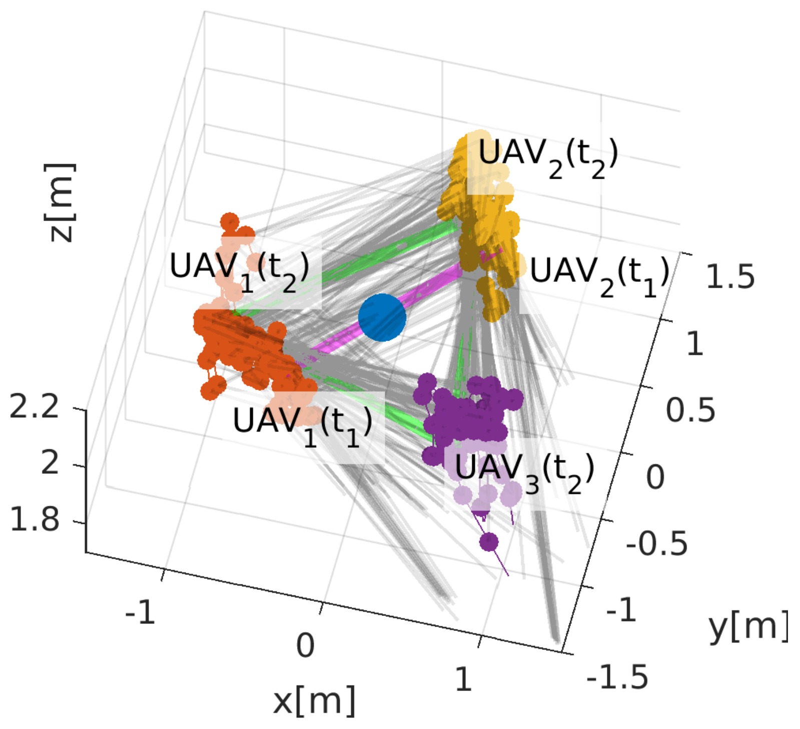


Figure 7.11: Runtime formation change scenario: 3D position plot with formation visualization

message at t_2 is removing UAV_3 as well as the associated constraints from the system. Accordingly, UAV_1 and UAV_2 are forming the original linear formation again.

For a more detailed overview on the timing, the system trajectories are given in the following. The corresponding position trajectories in Figure 7.12a-7.12c are confirming position changes at $t_1 \approx 8\,\text{s}$ (left vertical bar in the graphs) and $t_2 \approx 44\,\text{s}$ (right vertical bar in the graphs). As expected, Figure 7.12c shows that UAV_3 is not actively controlled for $t < t_1$ and $t > t_2$. This is stated by the high position oscillations at the beginning and the y-drift at the end of the plot. The position plots also show, that the UAV positions are more disturbed between t_1 and t_2. This increase is caused by the influence of the additional UAV_3 airflow. In combination with the adhesive target tracking and the repulsive CA coupling, this manifests in oscillations. These oscillations can be reduced by penalizing the UAV velocity states under the cost of slower trajectory changes. Another way to reduce this disturbance is to consider a disturbance model within the quadrotor model. The disturbance is causing a higher control action as shown in Figure 7.12d-7.12f. Due to the control

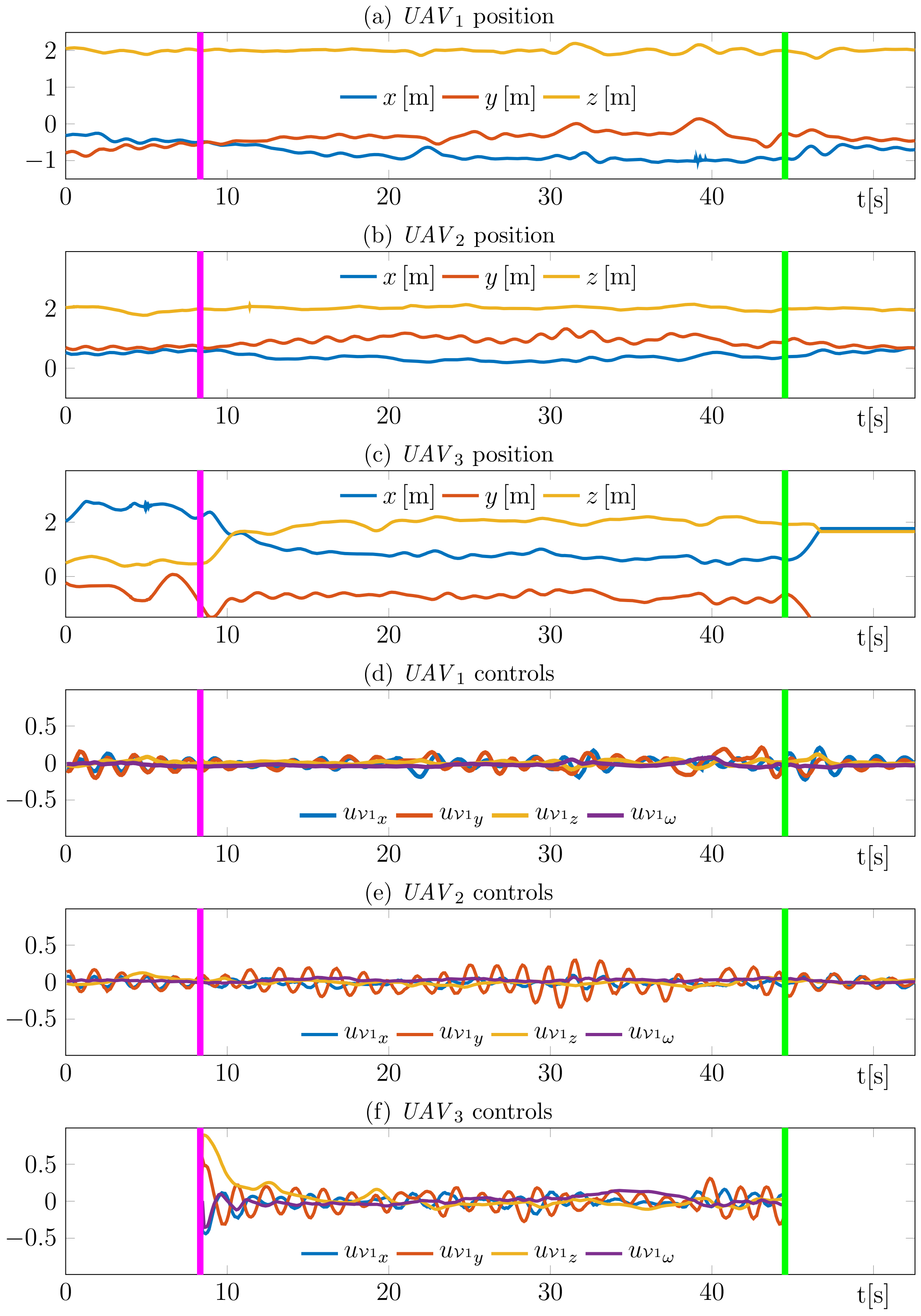


Figure 7.12: Runtime formation change scenario: *UAV* trajectories

limit constraints, the controls do not exceed the predefined limit of $max(|u_{\nu 1_z}|) = 1$. This indicates an active constraint handling.

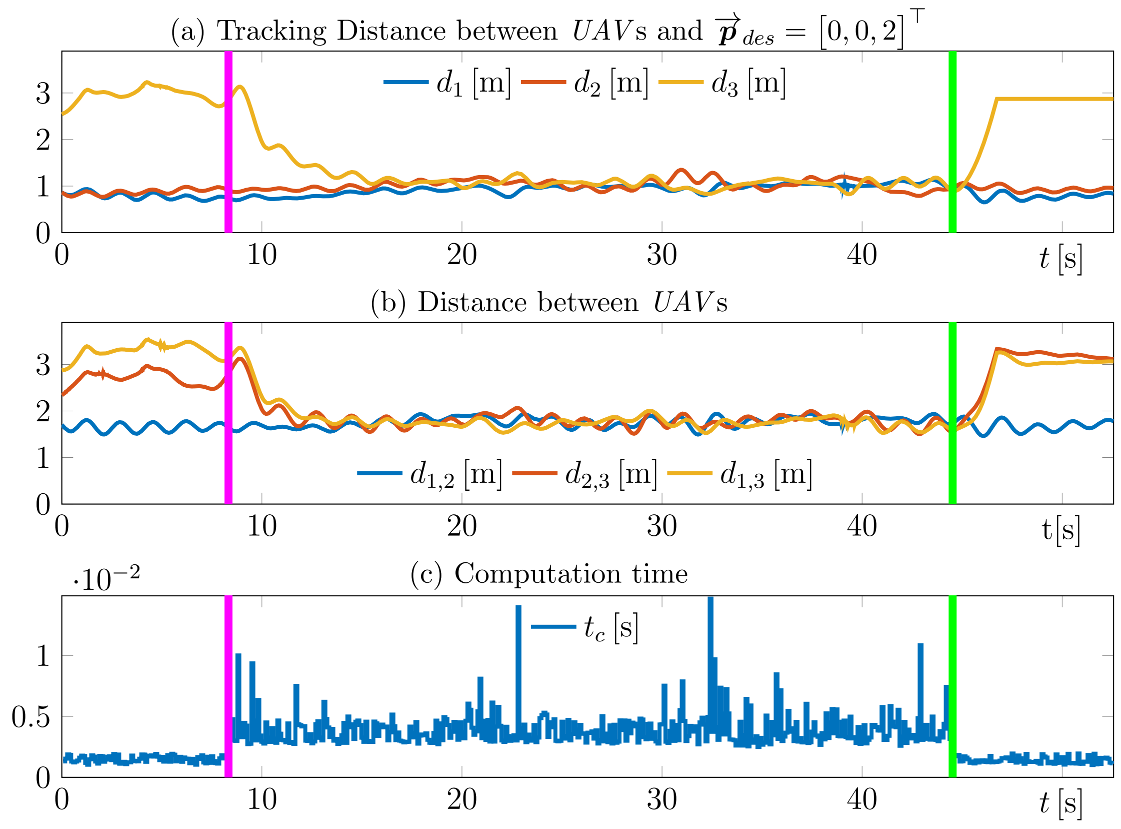


Figure 7.13: Runtime formation change scenario: distance, tracking and computation time

To evaluate the control performance, the distance graph in Figure 7.13b shows that at the beginning of the experiment $\|\vec{\boldsymbol{p}}_3 - \vec{\boldsymbol{p}}_2\|_2 \approx 2.5\,\text{m}$ and $\|\vec{\boldsymbol{p}}_3 - \vec{\boldsymbol{p}}_1\|_2 \approx 3.0\,\text{m}$. Accordingly, UAV_3 is not tracking the target $\boldsymbol{x_{des}}$ before t_1. At time instance t_1, the distance of UAV_3 to the other agents converges to $d = 1.5\,\text{m}$. The target tracking error graph in Figure 7.13a confirms the equidistant alignment of the agents to the target. This confirms an active *CA*-coupling and target tracking. The increase in the distance after t_2 is evidence that the position tracking is not active anymore. As previously discussed at time instance t_1, the tracking error of UAV_3 is high $\|[0,0,2]^\top - \vec{\boldsymbol{p}}_3\|_2 \approx 3\,\text{m}$.

The computational efficiency of *DENMPC* is stated by the computation time in Figure 7.13c. For the *OCP* (7.46) of the double agent system (UAV_1, UAV_2) the average computation time is $\bar{t_c} = 1.5\,\text{ms}$, the maximum computation time is $max(t_c) = 2.0\,\text{ms}$. The measurements were made on a standard notebook of type *Dell Latitude E5440* (see §A.1). The three agent system *OCP* (7.47) with UAV_1, UAV_2

and UAV_3 is solved in $\bar{t_c} = 4\,\text{ms}$. Also here, the higher computational load leads to more CPU interrupts which manifests in some single peaks up to $max(t_c) = 14.7\,\text{ms}$. At first sight, adding one agent causing a doubling of the computation time might look over-proportional. However, it has to be taken into account that the amount of *CA* couplings is increased by two. The average computation time of $\bar{t_c} = 4\,\text{ms}$ for the three drone scenario with a control update interval of $\Delta t = 0.1\,\text{s}$ results to an average computational load of 4% on the utilized hardware. This states the computational efficiency of *DENMPC*, under use of the proposed constraint handling technique, modularization, framework structure and *CMSC/GMRES* solver.

7.5.4 Cooperative aerial manipulation

To address the problem statement §1.1, a cooperative aerial manipulation scenario is conducted. This experiment serves the cooperative control validation for a heterogeneous system. For this purpose the aerial manipulation scenario in §6.4.1 is extended by a sensor *UAV*. The resulting setup with a *DJI M100 MUAV* and an *AR.Drone 2.0* sensing drone is shown in Figure 7.14.

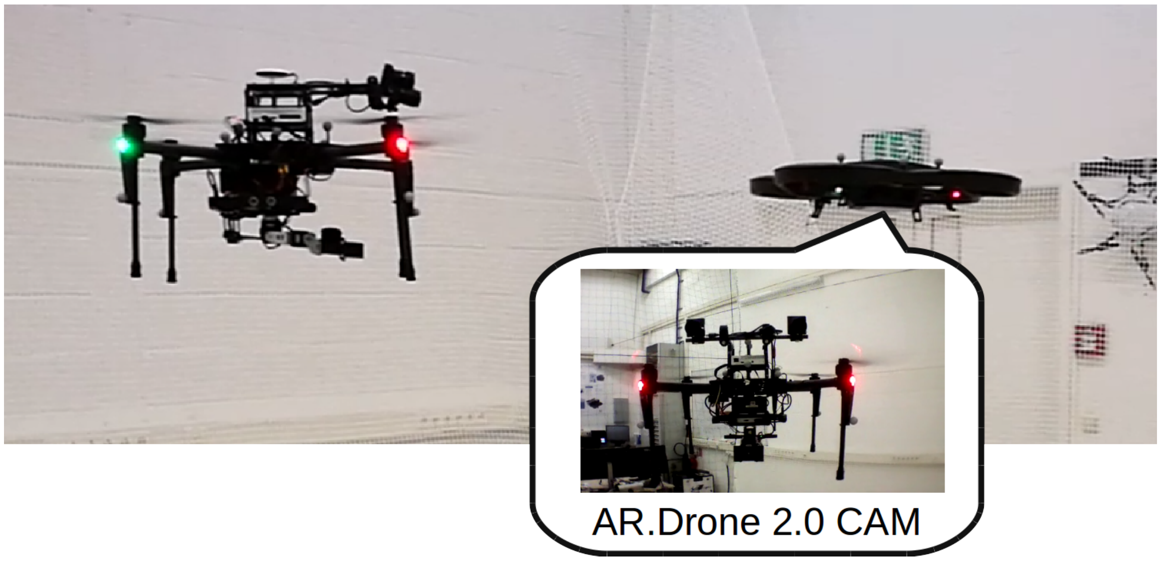


Figure 7.14: Cooperative aerial manipulation video footage

In this scenario, the *DJI M100 MUAV* is tracking a set of different poses with its end effector as shown in §6.4.1. As previously, the forward kinematics *FK* (6.8) are used to track a target pose $\overrightarrow{\boldsymbol{p}}_{des,1}$ and orientation $\boldsymbol{o}_{des,1}$. Both are provided in the form of the desired state $\boldsymbol{x_{des}}_1$ of UAV_1. The end effector pose tracking cost function yields

$$\begin{aligned}
&L_{\mathcal{E}}\left(\boldsymbol{x}_{\boldsymbol{des}1}, \boldsymbol{x}_1, \theta_1, \theta_2\right) \\
&= \left(\overrightarrow{\boldsymbol{p}}_{des,1} - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}\left(\theta_1, \theta_2, \boldsymbol{\rho}_1, \overrightarrow{\boldsymbol{p}}_1\right)\right)^{\mathrm{T}} \boldsymbol{Q}_{ee,p} \left(\overrightarrow{\boldsymbol{p}}_{des,1} - {}^{\mathcal{G}}\overrightarrow{\boldsymbol{p}}_{\mathcal{E}}\left(\theta_1, \theta_2, \boldsymbol{\rho}_1, \overrightarrow{\boldsymbol{p}}_1\right)\right) \\
&+ \left(\boldsymbol{o}_{des,1} - \boldsymbol{o}_1\right)^{\mathrm{T}} \boldsymbol{Q}_{ee,o} \left(\boldsymbol{o}_{des,1} - \boldsymbol{o}_1\right) \qquad (7.53)
\end{aligned}$$

In order to continuously provide sensor information to the *MUAV*, an *AR.Drone 2.0* is cooperatively controlled to face the front of the *MUAV* in a distance of $d_{des} = 1.5\,\mathrm{m}$. The related cost function is developed according to the trigonometric scheme shown in Figure 7.15

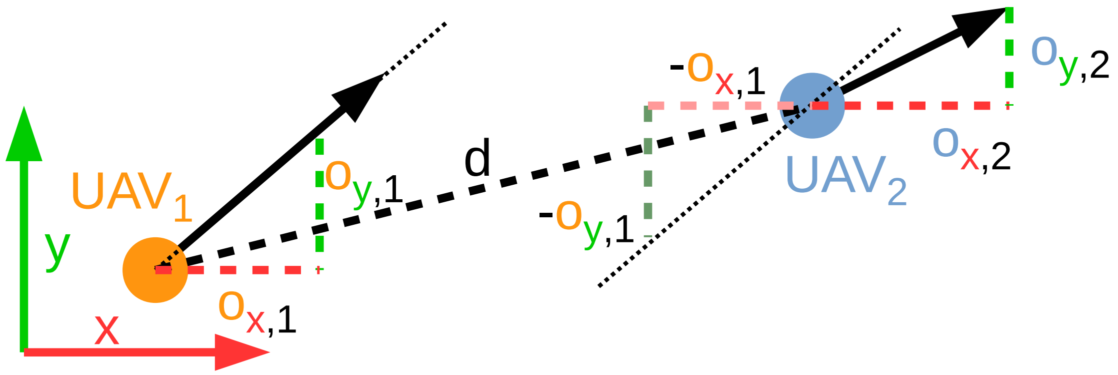


Figure 7.15: Trigonometric scheme for tracking function

$$\begin{aligned}
&L_S\left(\boldsymbol{x}_1, \boldsymbol{x}_2\right) = q_{\boldsymbol{o}}\left(\boldsymbol{o}_2 - \left(-\boldsymbol{o}_1\right)\right)^{\mathrm{T}} \boldsymbol{I}^{2\times 2}\left(\boldsymbol{o}_2 - \left(-\boldsymbol{o}_1\right)\right) \qquad (7.54) \\
&+q_d\left(\overrightarrow{\boldsymbol{p}}_2 - \overrightarrow{\boldsymbol{p}}_1 - d_{des}\cdot\left[o_{x1}, o_{x1}, 0\right]^{\mathrm{T}}\right)^{\mathrm{T}} \boldsymbol{I}^{3\times 3}\left(\overrightarrow{\boldsymbol{p}}_2 - \overrightarrow{\boldsymbol{p}}_1 - d_{des}\cdot\left[o_{x1}, o_{x1}, 0\right]^{\mathrm{T}}\right).
\end{aligned}$$

(7.54) can also be extended to allow the facing of different orientations of the *MUAV* using trigonometric addition theorems.

To show the cooperative effect in the scenario, the end effector joint angles are changed during the scenario. In accordance to this translation, the *AR.Drone 2.0* is adapting to the *DJI M100*'s center pose. The *UAV* motions are predicted with the $\boldsymbol{f}_1 = \boldsymbol{f}_{DJI\ M100}$ (3.55) and $\boldsymbol{f}_2 = \boldsymbol{f}_{AR.Drone\ 2.0}$ (3.49) *HMDV*. With $L_{\mathcal{E}}\left(\boldsymbol{x}_{\boldsymbol{des}1}, \boldsymbol{x}_1, \theta_1, \theta_2\right)$ (7.53) and $L_S\left(\boldsymbol{x}_1, \boldsymbol{x}_2\right)$ (7.54) the *OCP* yields

$$\min_{\underline{\boldsymbol{u}}} J = \int_t^{t+T} \sum_{i=1}^{2} \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{R}_i \boldsymbol{u}_i + L_{\mathcal{E}}\left(\boldsymbol{x}_{\boldsymbol{des}1}, \boldsymbol{x}_1, \theta_1, \theta_2\right) + L_S\left(\boldsymbol{x}_1, \boldsymbol{x}_2\right) \mathrm{d}\tau \qquad (7.55)$$

$$\text{s.t.}\ \dot{\boldsymbol{x}}_i = \boldsymbol{f}_i\left(\boldsymbol{x}_i, \boldsymbol{u}_i, \tau\right) \qquad (7.56)$$

$$\boldsymbol{0} \leq \boldsymbol{c_{ini}} = \left[\underline{u}_{\mathcal{V}1_{x,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{y,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{z,i}}{}^2 - 1, \underline{u}_{\mathcal{V}1_{\omega,i}}{}^2 - 1\right] \qquad (7.57)$$

$$\boldsymbol{x}_1(0) = \left[1, 1, 1.5, 1, 0, 0, 0, 0, 0\right], \quad \boldsymbol{x}_2(0) = \left[1, 1, 2, 1, 0, 0, 0, 0, 0\right] \qquad (7.58)$$

The related parameters are chosen to

$$\boldsymbol{Q}_{ee,p} = \begin{bmatrix}1,1,1\end{bmatrix}, \quad \boldsymbol{Q}_{ee,o} = \begin{bmatrix}10,10\end{bmatrix}, \quad \boldsymbol{R}_1 \;=\; \begin{bmatrix}10,10,10,10\end{bmatrix} \tag{7.59}$$

$$q_{\boldsymbol{o}} = 1, \quad q_d = 1, \quad \boldsymbol{R}_2 \;=\; \begin{bmatrix}1,1,1,1\end{bmatrix} \tag{7.60}$$

$$\xi = 1,\; \upsilon \;=\; 1,\; \Delta t \;=\; 0.1\,\mathrm{s}, T \;=\; 1\,\mathrm{s}, \tag{7.61}$$

$$N = 10,\; h \;=\; 0.001\,\mathrm{s},\; h \;=\; 10^{-8},\; i_{max} = 10. \tag{7.62}$$

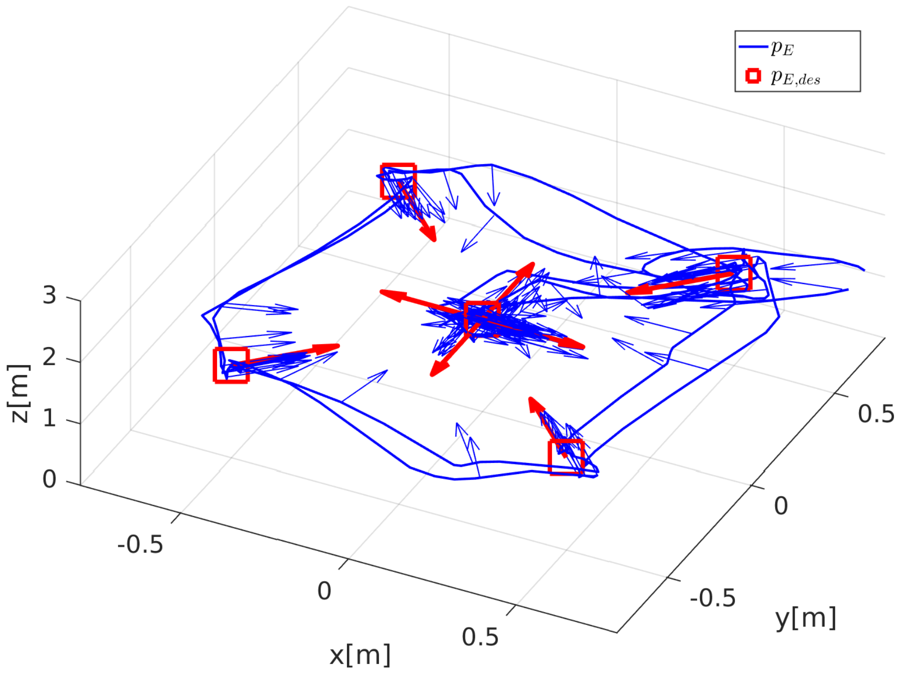


Figure 7.16: Cooperative aerial manipulation: *MUAV* end effector pose tracking

To evaluate the control performance, the system trajectories are given in the following. The spacial behavior of the *MUAV* end effector target and pose trajectories are visualized in Figure 7.16. Between the target poses, the *MUAV*'s trajectory is moving according to its dynamics. The position trajectories in Figure 7.17a-7.17c are confirming the convergence of the *MUAV* end effector towards the desired end effector position. The *DJI M100* center position is hereby translated according to the manipulator joint angles θ_1, θ_2 as shown in Figure 7.17f. This visualizes the forward kinematics of the manipulator and is particularly visible in the altitude (z) at $t = 100\,\mathrm{s}$ and $t = 130\,\mathrm{s}$ (Figure 7.17c). Figure 7.17d is showing the orientation of the *DJI M100* in terms of heading angle Ψ. Also here, a convergence towards the desired position is visible and validates the orientation tracking. The controls $\boldsymbol{u}_1$ are

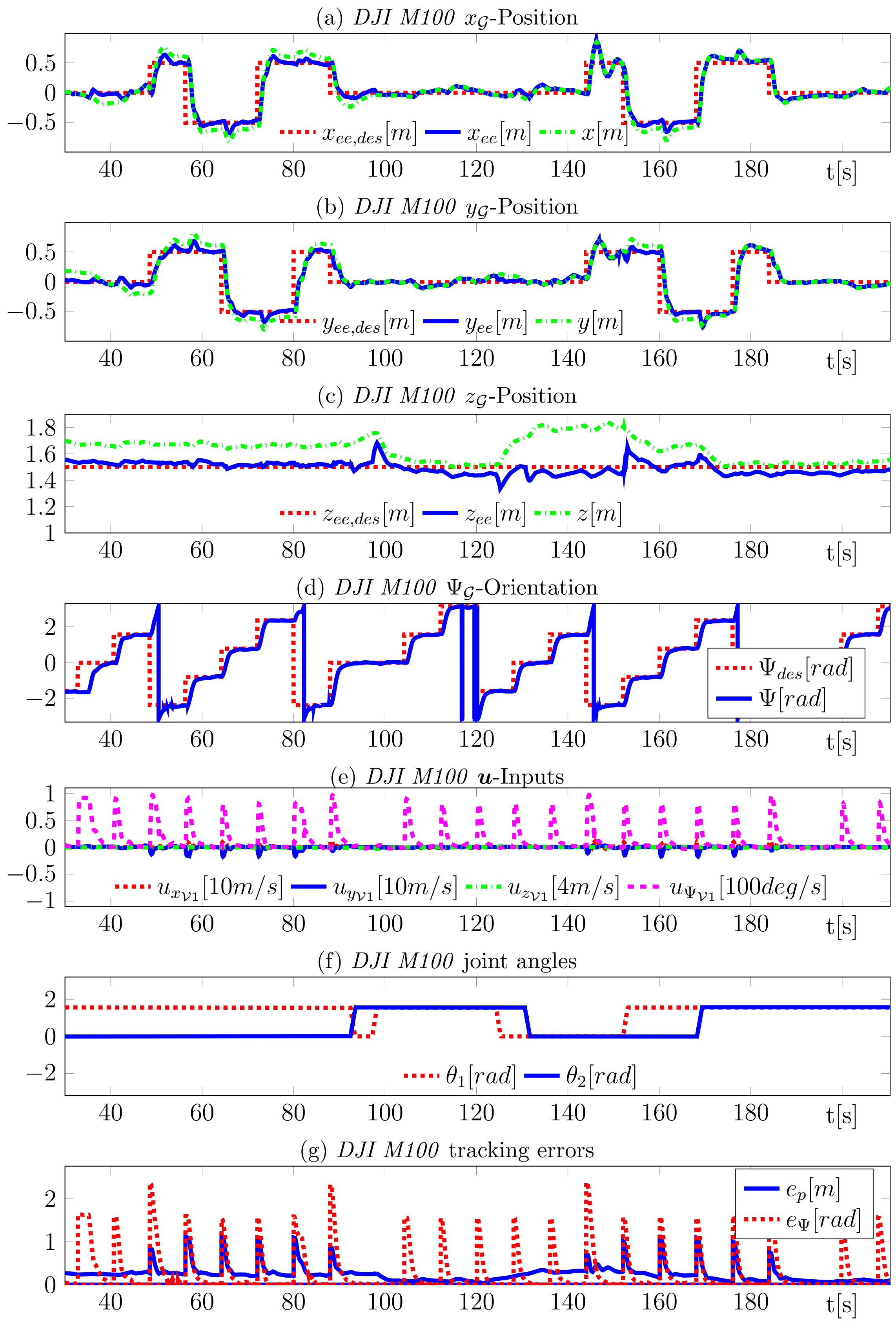


Figure 7.17: Cooperative aerial manipulation: *DJI M100* trajectories

given in Figure 7.17e and show the characteristic peaks at each change of the tracked pose. This is the direct response to the sudden changes in position and orientation error which are shown in Figure 7.17g. As expected, the *DJI M100* shows the desired behavior as in the aerial manipulation scenario in §6.4.1.

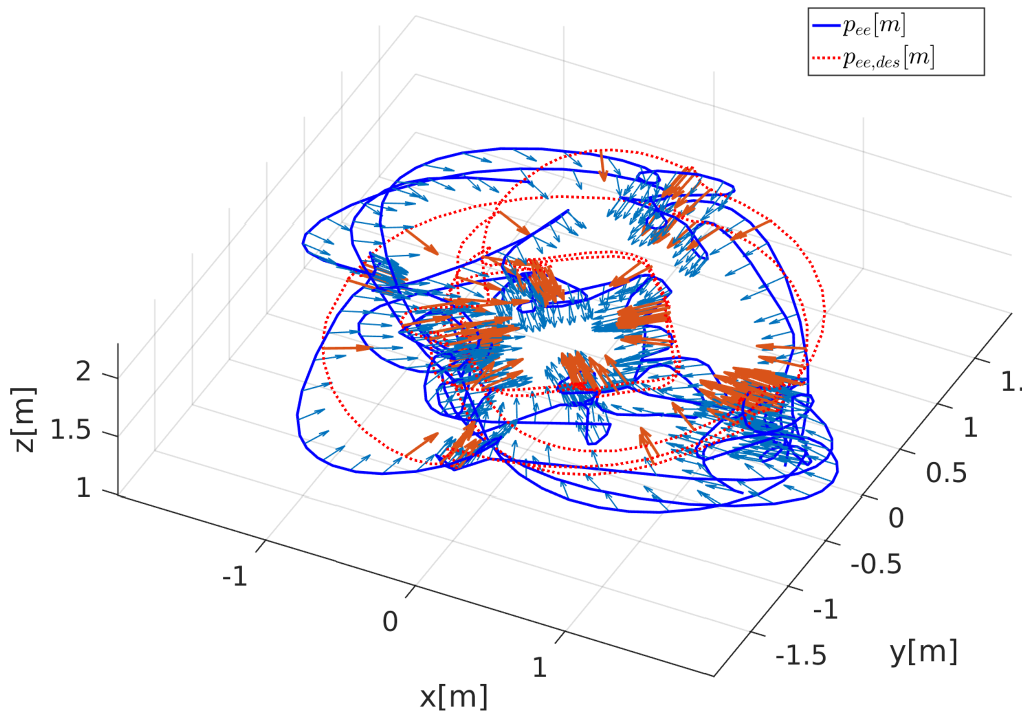


Figure 7.18: Sensing *UAV* tracking 3d plot showing the virtual *AR.Drone 2.0* reference ($p_{ee,des}$ and its trajectory p_{ee})

To evaluate the performance of the sensing *UAV*, the required pose to fulfill tracking condition (7.54) is calculated as $\boldsymbol{x_{des2}}$. Figure 7.18 is showing the 3d plot of the *AR.Drone 2.0* position. The arrows are hereby indicating the orientation of the *AR.Drone 2.0*. The resulting *AR.Drone 2.0* position trajectory is given in Figure 7.19a-7.19c. As the behavior of the sensing UAV_2 is directly related via L_S (7.54) to UAV_1, the disturbance of UAV_1 is translated to UAV_2 by means of the distance lever d. The excellent tracking performance is visible in the x and y plots, as target and actual position are congruent. The z-axis shows significantly more disturbance, as it is sensitive to airflow disturbance. This is visible at the time instances when $\boldsymbol{x_{des1}}$ is changing. The required orientation to fulfill tracking condition (7.54) is tracked according to Figure 7.19d. The jittering vertical lines between $\pm\pi$

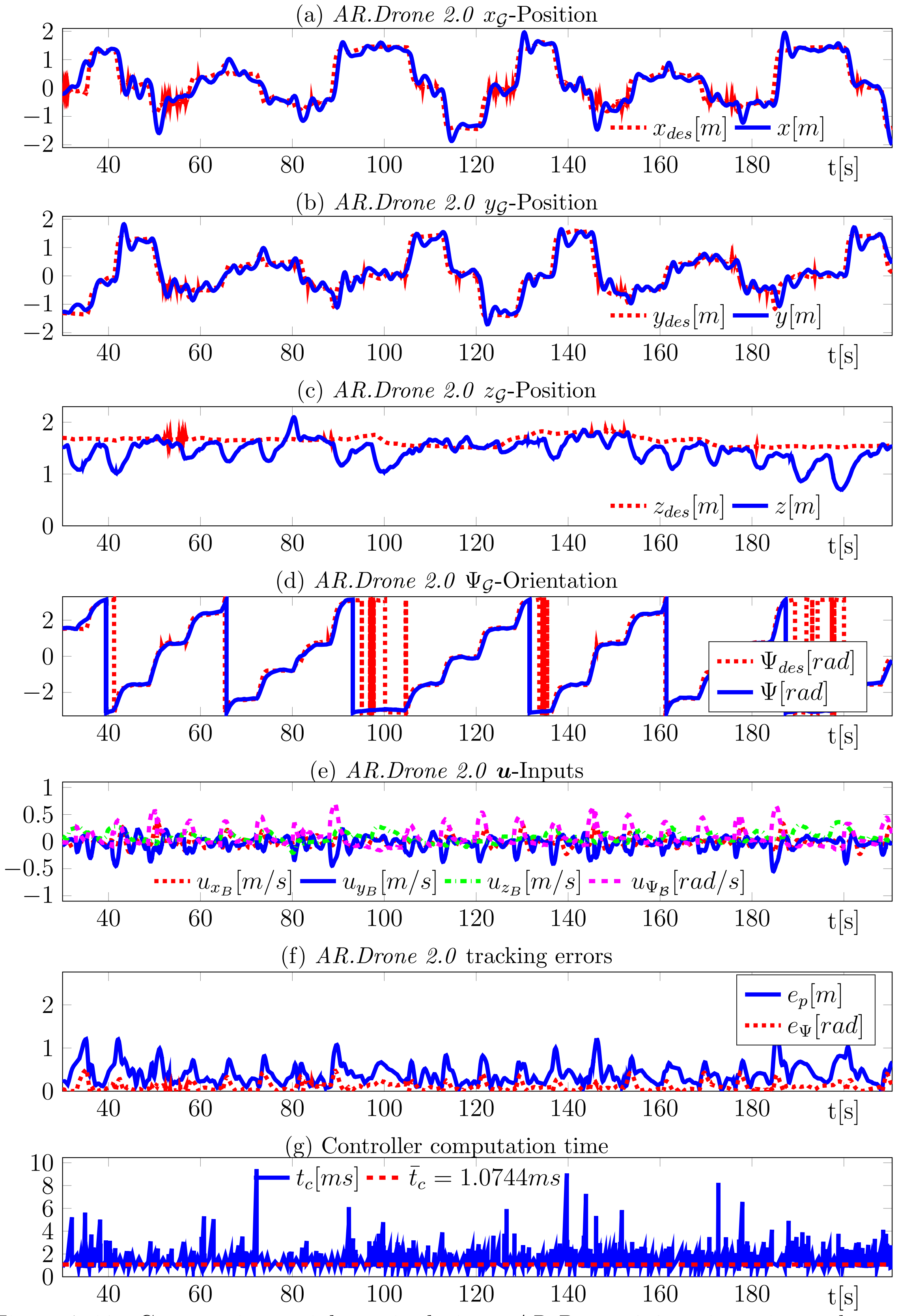


Figure 7.19: Cooperative aerial manipulation: *AR.Drone 2.0* trajectories and controller computation time

are caused by the definition of the Ψ interval. As for the *DJI M100*, also the controls of the *AR.Drone 2.0* in Figure 7.19e do show the characteristic response peak at each reference change. The error plot in Figure 7.19f is validating the corresponding disturbance rejection. The computational performance is stated in Figure 7.19g. The maximal computation time of $max(t_c) = 9.4\,\mathrm{ms}$ is smaller than the utilized control update interval of $\Delta t = 10\,\mathrm{ms}$. The real-time applicability is also stated by the low average computation time of $\bar{t_c} = 1.1\,\mathrm{ms}$.

The results of the experimental evaluation confirm the effectiveness and computational efficiency of the proposed control approach. This confirms the real-time applicability of *DENMPC* for small cooperative aerial manipulation scenarios. The presented scenario is hereby representative for cooperative aerial manipulation scenario as given in the problem statement of this thesis §1.1.

7.6 Conclusion

This chapter has addressed the cooperative control of *UAV*s with *NMPC*. The given related work has stated that to the author's knowledge, no central fast *NMPC* framework exists that is suitable to control fast distributed systems. This refers to the combination of low computation times with the capability of modifying the inherent *OCP* at runtime. This capability is required to adapt to changes in the control objectives, constraints and topology of dynamic scenarios.

For this purpose, the *OCP* and related optimality conditions have been analyzed for distributed systems. This analysis has been conducted for the constraint handling presented in §4.5. It has been shown, that the *OCP* of distributed systems can be mathematically decomposed into atomic functions. These can be provided as compiled fast *C++* code. Furthermore, by choosing these atomic functions accordingly, runtime matrix multiplications can be avoided in the composition of the Lagrangian and its derivatives. As those are typically sparse, this increases the computational efficiency. For this purpose, the atomic functions are compiled for each agent, coupling and constraint. As a result, the distributed *OCP* and related optimality conditions can be composed at runtime for any arbitrary scenario with these elements and without recompilation.

The object-oriented *C++* implementation of this modularization into base class containers has been presented and the code-generation has been discussed. In this context, the fast functor/pointer access scheme has been demonstrated which minimizes the computational overhead introduced by the modularization. In order to trigger changes of the *OCP* online, an event handling mechanism has been implemented including *ROS* messages and timers. The result is the Distributed System Event-Based Nonlinear Model Predictive Control (*DENMPC*) framework that offers the flexibility to modify the control scenario at runtime, while maintaining low computation times. This makes it particularly suitable for prototyping, single and multi robot applications as well as fault-tolerant control. The framework source code is available under [Den16].

To validate the features of *DENMPC* four different example scenarios have been conducted. The first scenario consisted of a simulative formation flight of two *AR.Drone 2.0 UAV*s. In order to evaluate the computational overhead of the modularization, the modular *DENMPC* and an explicit *NMPC* approach have been used to control the scenario. For the given scenario, the computational overhead was 16% and the real-time applicability has been confirmed. For the second scenario, a leader-follower scenario with two real *AR.Drone 2.0*s has been controlled with *DENMPC*. Here, one of the *AR.Drone 2.0*s has been tracking the other *UAV* within a confined sensor space. This scenario has validated the use of unidirectional couplings, the modularization and the task-based *MPC* approach from §5.6 for cooperative scenarios. The real *AR.Drone 2.0* trajectories have confirmed the effectiveness of the proposed control. Using the mathematically complex sensor constraint, the low average computation time of $\bar{t_c} \approx 2.54\,\mathrm{ms}$ is confirming the real-time applicability and computational efficiency. To evaluate the online adaptability, *DENMPC* has been used in a real *UAV* formation change scenario. The experiment showed a linear flight formation of two *AR.Drone 2.0* quadrotors that has been extended to a three quadrotor triangular formation. The drone trajectories have confirmed the online switching of quadrotor dynamics and objectives, *CA* couplings and input limitation constraints triggered by a *ROS* message event. Also here, the average computation time of $\bar{t_c} = 1.5\,\mathrm{ms}$ for the two drone scenario and $\bar{t_c} = 4\,\mathrm{ms}$ for the three drone scenario state the computational efficiency of *DENMPC*. The final experiment serves as example for cooperative aerial manipulation scenarios. In this scenario a *DJI M100 MUAV* from §6 is tracked by an *AR.Drone 2.0* quadrotor. Also here the control effectiveness is stated by the system's response and the real-time applicability is demonstrated with a low average computation time of $\bar{t_c} = 1.1\,\mathrm{ms}$.

The presented work represents the first stage of solving *OCP*s of distributed systems. Future work will focus on the implementation of distributed *NMPC* methods to make also use of event-based *NMPC* on large-scale systems. Further implementations will also address additional inequality constraint handling techniques. To evaluate the performance of these techniques an extensive benchmark and their mathematical evaluation is a future field of interest. Furthermore, the effects of the event-based *NMPC* adaptation have to be analyzed regarding problem feasibility and control performance. This does include an analysis of the system behavior under runtime switching of control objectives.

Chapter 8

Conclusion and future work

This thesis investigated real-time model predictive control of *UAV*s, *MUAV*s, and cooperative aerial manipulation scenarios. In correspondence to the thesis structure, there have been five major fields of research:

Quadrotor models

The prerequisite for *MPC* is a prediction model. For this purpose, different ways to describe the motion of a quadrotor *UAV* have been discussed in chapter §3. This includes a physical model based on rigid body dynamics and a hover model based on a linearization around its static equilibrium. The focus of this thesis has been on the latter, as it is particularly suited for commercial *UAV*s that have an internal attitude stabilization controller. Furthermore, less mathematical operations are needed to describe the *UAV* behavior accurately. The results are low *MPC* computation times and a high control performance. In this context, a direction vector description has been introduced to solve the Euler-angle singularity problem. This allows the full 360° control of a *UAV*'s heading angle. Finally, the model parameters for an *AR.Drone 2.0* and *DJI M100* have been identified. These are well-established representatives of commercial *UAV* solutions. The presented identification of these systems targets to facilitate the prototyping for research and industry in future.

Model predictive control

Due to the computational and energetic limitations of mobile robots, a computationally efficient *MPC* is required. In §4, the basic *NMPC* principles used within this thesis are presented. Based on these, a benchmark of several fast *NMPC* algorithms has been conducted. This includes a gradient-based method from the *GRAMPC* framework, a Gauß-Newton-*SQP* approach from *ACADO* and *C/GMRES*, respectively *CMSC/GMRES*. The computational performance has been evaluated in a simulated control scenario using the physical model of a *UAV*. In the given scenario, *CMSC/GMRES* could stabilize the system with an exceptional low computation time of $max(t_c) = 0.28\,\text{ms}$ and $\overline{t_c} = 0.17\,\text{ms}$. Due to its control performance, *CMSC/GMRES* has been chosen as base algorithm for this thesis. One of the advantages of *MPC* is the consideration of constraints. For this reason, three lightweight constraint handling techniques for robotic applications have been discussed. This includes the saturation function, primal barrier and auxiliary variable approaches. The major concern regarding *NMPC* for mobile robots is the difficulty to prove stability for such complex algorithms in combination with nonlinear systems. To facilitate future development of stability proofs, the *CMSC/GMRES* algorithm is given in very detailed step by step instructions.

***UAV* control**

Chapter §5 is contributing the *NMPC* control of real *UAV*s, in particular an *AR.Drone 2.0* and *DJI M100* quadrotor. The parametrization for both *UAV*s is determined in experiments. This is shown in detail for the *AR.Drone 2.0*. Computation times of $\bar{t}_{c AR.Drone\ 2.0} = 0.3700\,\text{ms}$ and $\bar{t}_{c DJI\ M100} = 0.4295\,\text{ms}$ state the computational efficiency of the proposed *NMPC*. The control performance is validated in pose tracking scenarios. One disadvantage of the missing integral part in the *NMPC* control policy is a constant error, if the target is moving with a constant velocity. To address this problem, a target position control has been presented. This *TPC* virtually translates the target position in order to introduce an integral control behavior.

To promote the use of *UAV*s in urban, respectively cluttered environments, the pose tracking *NMPC* has been extended with a collision avoidance constraint. The desired evasion maneuver is validated with a real *AR.Drone 2.0*. With the objective of formulating more complex control objectives, a workflow to develop suitable potential functions has been presented. Subsequently, this task-based workflow has been used to develop also *CA* and cohesion constraints. In the demonstration scenario, a real *AR.Drone 2.0* has been used to track a given target within a confined sensor space. Furthermore, an obstacle has been introduced to test the developed *CA* constraints. A set of 10 experiments have been conducted and confirmed the effectiveness of the contributed workflow for different initial conditions and obstacle trajectories.

Aerial manipulation

§6 is discussing the kinematic control of a manipulating aerial vehicle. For this purpose a robotic arm has been attached to a *DJI M100* quadrotor. The forward kinematics have been derived including position and orientation. To move the robotic joints, a closed-form inverse kinematic control has been developed. This control allows to achieve a given manipulator pitch angle. For the end effector position of the robotic arm, an *NMPC* concept has been developed. The concept is based on a separate *NMPC* treatment of the end effector position and orientation. As a consequence, the previously developed hover model can be utilized. The result is low computation times while achieving the desired control performance. This performance has been evaluated in two experiments with a real *DJI M100 MUAV*. A grasping scenario of a bottle served as demonstration of a real application. With an average computation time of $t = 0.256\,\text{ms}$, the real-time capability has been stated. This scenario has validated the performance of *NMPC* for aerial manipulation and shows the potential for industrial applications.

Cooperative control

The final field of research has been the control of multiple mobile robots. For this purpose, a central *NMPC* has been analyzed regarding the distributed nature of such systems. In order to accommodate the dynamic nature of such systems, a modularization has been introduced. This allows to change control objectives, constraints and the system topology at runtime while maintaining low computation times. In combination with event handling mechanisms that manages those changes, the open source central *NMPC* framework (*DENMPC*) has been developed. This framework targets the control of small-scale multi-robot systems, fault-tolerant systems and the control of systems with changing control objectives. To achieve low computation-times, the *C++* implementation is using compiled atomic functions with fast functor/pointer access. Furthermore, expensive matrix multiplications are already executed in compilation-time which further reduces the computation time. The features of *DENMPC* are evaluated in dedicated real *UAV* experiments, stating the runtime adaptability, computational efficiency and effectiveness. In this context, a cooperative aerial manipulation scenario has been conducted. The outline is a real *DJI M100 MUAV* which is tracking a target position with its end effector. At the same time an *AR.Drone 2.0* is controlled to face the *MUAV* with its camera. Here, the real-time applicability of *DENMPC* is confirmed with a low average computation time of only $\bar{t_c} = 1.1\,\text{ms}$ per control update. Furthermore, the runtime adaptability of *DENMPC* in combination with a task step-chain allows to execute complex tasks. One conducted example is the grasping of a bottle and pouring of its content into a mug, as shown in Figure 8.1.

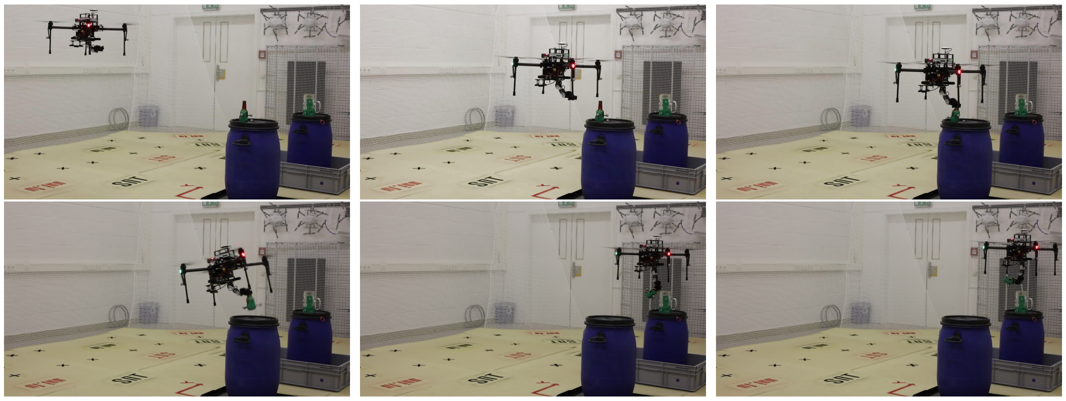


Figure 8.1: Aerial manipulation scenario: Autonomous grasping and pouring of a bottle using a task step chain in combination with *DENMPC*

With *DENMPC*, this thesis has contributed a control framework for real-time control of cooperative aerial manipulation scenarios. In this context, this work has addressed the future key abilities of aerial robots defined in the "European Roadmap for Robotics in 2020" [SPA16].

8.1 Future work

Future work will focus on the application of aerial manipulation on industrial problems. As this thesis has only focused on the control perspective, the *MUAV* perception has to be incorporated accordingly. For the given cooperative aerial manipulation scenario, this means the visual tracking of a *UAV*, respectively objects.

One step in this direction for cooperative systems has been the publication of the **ATLAS** framework in [PDKV17]. **ATLAS** is dedicated to cooperative localization using sensor fusion. A further direction of research for large-scale multi-robot systems is the distribution of the presented central *DENMPC*. This refers to the application of *DENMPC* in a distributed manner and the implementation of fast global consensus techniques.

Appendix A

Annex

A.1 Hardware & software

The experiments have been conducted and analysed with the following hard and software:

Hardware

- Computer Platform: Dell Latitude E5440 (Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz, 8GB RAM @ 1600MHz)
- Motion Capture System: Optitrack ©2018 NaturalPoint (using six Prime 13 high speed tracking cameras)
- UAVs:
  - *DJI M100*: DJI Matrice 100
  - *AR.Drone 2.0*: Parrot Augmented Reality Drone 2.0
- Manipulator Servos: Dynamixel MX28T, Dynamixel AX18A, Dynamixel2USB Interface

Software

- Ubuntu 14.04.1: kernel 4.4.0-89 (on Dell Latitude E5440)
- Debian kernel 3.1.0 (on DJI M100)
- MATLAB R2016a
- Mathematica 10.0
- ROS Indigo

A.2 Global OCP for distributed systems and optimality conditions for different constraint handling methods

This section is extending the results of section §7.3 by providing the Lagrangian derivatives for the global *OCP* of distributed systems with inequality constraints. For this purpose the symbolic conventions of §7 are used.

Global OCP with primal barrier constraint handling

Using the primal barrier method of §4.5.1 to handle inequality constraints results to the Lagrangian

$$\begin{aligned}\mathcal{L} = \sum_{i=1}^{n_\nu} \Bigg(& L_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) + \underline{\boldsymbol{\lambda}}_i^{\mathrm{T}} \boldsymbol{f}_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) + \underline{\boldsymbol{\lambda}_{\boldsymbol{eq}}}_i^{\mathrm{T}} \boldsymbol{c}_{\boldsymbol{eq}_i}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) \\ & + \sum_{l=1}^{n_{c_{in\,i}}} \left[\underline{\boldsymbol{\varphi}}_i^{\mathrm{T}} \ln\left(-\boldsymbol{c}_{\boldsymbol{in}\,i}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right)\right)\right] + \sum_{j \neq i}^{n_{\epsilon i}} \Big\langle L_{i,j}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j\right) \\ & + \underline{\boldsymbol{\lambda}_{\boldsymbol{eq}}}_{i,j}^{\mathrm{T}} \boldsymbol{c}_{\boldsymbol{eq}_{i,j}}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j\right) + \sum_{l=1}^{n_{c_{in\,i,j}}} \left[\underline{\boldsymbol{\varphi}}_{i,j}^{\mathrm{T}} \ln\left(-\boldsymbol{c}_{\boldsymbol{in}\,i,j}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j\right)\right)\right] \Big\rangle \Bigg).\end{aligned} \tag{A.1}$$

The controls of each agent $\boldsymbol{u}_i$, the constraint Lagrange multipliers $\boldsymbol{\lambda}_{\boldsymbol{eq}_i}$ for constraints and for couplings $\boldsymbol{\lambda}_{\boldsymbol{eq}_{i,j}}$ are concatenated to the global optimization variable vector

$$\boldsymbol{w} = \left[\boldsymbol{u}_1^{\mathrm{T}}, ..., \boldsymbol{u}_{n_\nu}^{\mathrm{T}}, \boldsymbol{\lambda}_{\boldsymbol{eq}_1}^{\mathrm{T}}, ..., \boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu}}^{\mathrm{T}}, \boldsymbol{\lambda}_{\boldsymbol{eq}_{1,2}}^{\mathrm{T}}, ..., \boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu-1,n_\nu}}^{\mathrm{T}}\right]^{\mathrm{T}}. \tag{A.2}$$

Accordingly, the first order optimality condition (4.52) result to (\ is referring to an element-wise division)

$$\mathcal{L}_{\boldsymbol{w}} = \tag{A.3}$$
$$\begin{bmatrix} \nabla_{\boldsymbol{u}_1} L_1 + \left(\nabla_{\boldsymbol{u}_1} \boldsymbol{f}_1\right)^{\mathrm{T}} \boldsymbol{\lambda}_1 + \left(\nabla_{\boldsymbol{u}_1} \boldsymbol{c_{eq}}_1\right)^{\mathrm{T}} \boldsymbol{\lambda_{eq}}_1 - \left(\nabla_{\boldsymbol{u}_1} \boldsymbol{c_{in}}_1\right)^{\mathrm{T}} \boldsymbol{\varphi}_1 \backslash \boldsymbol{c_{in}}_1 \\ \ldots + \sum_j^{n_{\epsilon 1}} \left\langle \nabla_{\boldsymbol{u}_1} L_{1,j} + \left(\nabla_{\boldsymbol{u}_1} \boldsymbol{c_{eq}}_{1,j}\right)^{\mathrm{T}} \boldsymbol{\lambda_{eq}}_{1j} + \left(\nabla_{\boldsymbol{u}_1} \boldsymbol{c_{in}}_{1,j}\right)^{\mathrm{T}} \boldsymbol{\varphi}_{1,j} \backslash \boldsymbol{c_{in}}_{1,j} \right\rangle \\ \vdots \\ \nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu} + \left(\nabla_{\boldsymbol{u}_{n_\nu}} \boldsymbol{f}_{n_\nu}\right)^{\mathrm{T}} \boldsymbol{\lambda}_{n_\nu} + \left(\nabla_{\boldsymbol{u}_{n_\nu}} \boldsymbol{c_{eq}}_{n_\nu}\right)^{\mathrm{T}} \boldsymbol{\lambda_{eq}}_{n_\nu} - \left(\nabla_{\boldsymbol{u}_{n_\nu}} \boldsymbol{c_{in}}_{n_\nu}\right)^{\mathrm{T}} \boldsymbol{\varphi}_{n_\nu} \backslash \boldsymbol{c_{in}}_{n_\nu} \\ \ldots + \sum_j^{n_{\epsilon n_\nu}} \left\langle \nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu,j} + \left(\nabla_{\boldsymbol{u}_{n_\nu}} \boldsymbol{c_{eq}}_{n_\nu,j}\right)^{\mathrm{T}} \boldsymbol{\lambda_{eq}}_{n_\nu j} + \left(\nabla_{\boldsymbol{u}_{n_\nu}} \boldsymbol{c_{in}}_{n_\nu,j}\right)^{\mathrm{T}} \boldsymbol{\varphi}_{n_\nu,j} \backslash \boldsymbol{c_{in}}_{n_\nu,j} \right\rangle \\ \boldsymbol{c_{eq}}_1 \\ \vdots \\ \boldsymbol{c_{eq}}_{n_\nu} \\ \boldsymbol{c_{eq}}_{1,2} \\ \vdots \\ \boldsymbol{c_{eq}}_{n_\nu-1,n_\nu} \end{bmatrix}$$

with the state derivative of the Lagrangian (4.51)

$$\mathcal{L}_{\boldsymbol{x}} = \tag{A.4}$$
$$\begin{bmatrix} \nabla_{\boldsymbol{x}_1} L_1 + \left(\nabla_{\boldsymbol{x}_1} \boldsymbol{f}_1\right)^{\mathrm{T}} \boldsymbol{\lambda}_1 + \left(\nabla_{\boldsymbol{x}_1} \boldsymbol{c_{eq}}_1\right)^{\mathrm{T}} \boldsymbol{\lambda_{eq}}_1 - \left(\nabla_{\boldsymbol{x}_1} \boldsymbol{c_{in}}_1\right)^{\mathrm{T}} \boldsymbol{\varphi}_1 \backslash \boldsymbol{c_{in}}_1 \\ \ldots + \sum_j^{n_{\epsilon 1}} \left\langle \nabla_{\boldsymbol{x}_1} L_{1,j} + \left(\nabla_{\boldsymbol{x}_1} \boldsymbol{c_{eq}}_{1,j}\right)^{\mathrm{T}} \boldsymbol{\lambda_{eq}}_{1,j} + \left(\nabla_{\boldsymbol{x}_1} \boldsymbol{c_{in}}_{1,j}\right)^{\mathrm{T}} \boldsymbol{\varphi}_{1,j} \backslash \boldsymbol{c_{in}}_{1,j} \right\rangle \\ \vdots \\ \nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu} + \left(\nabla_{\boldsymbol{x}_{n_\nu}} \boldsymbol{f}_{n_\nu}\right)^{\mathrm{T}} \boldsymbol{\lambda}_{n_\nu} + \left(\nabla_{\boldsymbol{x}_{n_\nu}} \boldsymbol{c_{eq}}_{n_\nu}\right)^{\mathrm{T}} \boldsymbol{\lambda_{eq}}_{n_\nu} - \left(\nabla_{\boldsymbol{x}_{n_\nu}} \boldsymbol{c_{in}}_{n_\nu}\right)^{\mathrm{T}} \boldsymbol{\varphi}_{n_\nu} \backslash \boldsymbol{c_{in}}_{n_\nu} \\ \ldots + \sum_j^{n_{\epsilon n_\nu}} \left\langle \nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu,j} + \left(\nabla_{\boldsymbol{u}_{n_\nu}} \boldsymbol{c_{eq}}_{n_\nu,j}\right)^{\mathrm{T}} \boldsymbol{\lambda_{eq}}_{n_\nu j} + \left(\nabla_{\boldsymbol{x}_{n_\nu}} \boldsymbol{c_{in}}_{n_\nu,j}\right)^{\mathrm{T}} \boldsymbol{\varphi}_{n_\nu,j} \backslash \boldsymbol{c_{in}}_{n_\nu,j} \right\rangle \end{bmatrix}.$$

Global OCP with auxiliary variable constraint handling

For the auxiliary variable approach as presented in §4.5.2, the global *OCP* yields

$$\begin{aligned} \mathcal{L} = \sum_{i=1}^{n_\nu} \Bigg(& L_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) + \underline{\boldsymbol{\lambda}}_i^{\mathrm{T}} \boldsymbol{f}_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) + \underline{\boldsymbol{\lambda_{eq}}}_i^{\mathrm{T}} \boldsymbol{c_{eq}}_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) \\ & + \sum_{l=1}^{n_{c_{ini}}} \left[\lambda_{ini,l}\left(c_{ini,l}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) + \nu_{i,l}^2\right) - \kappa_{i,j}\nu_{i,l}\right] + \sum_{j \neq i}^{n_{\epsilon i}} \Big\langle L_{i,j}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j\right) \\ & + \underline{\boldsymbol{\lambda_{eq}}}_{i,j}^{\mathrm{T}} \boldsymbol{c_{eq}}_{i,j}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j\right) \\ & + \sum_{l=1}^{n_{c_{in\,i,j}}} \left[\lambda_{ini,j,l}\left(c_{ini,j,l}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j\right) + \nu_{i,j,l}^2\right) - \kappa_{i,j,l}\nu_{i,j,l}\right] \Big\rangle \Bigg). \end{aligned} \tag{A.5}$$

As the auxiliary variable constraint handling does consider slack variables and inequality Lagrange multipliers as additional optimization variables, the global optimization variable vector is accordingly extended

$$\begin{aligned} \boldsymbol{w} = [& \boldsymbol{u}_1^{\mathrm{T}}, ..., \boldsymbol{u}_{n_\nu}^{\mathrm{T}}, \boldsymbol{\lambda_{eq}}_1^{\mathrm{T}}, ..., \boldsymbol{\lambda_{eq}}_{n_\nu}^{\mathrm{T}}, \boldsymbol{\lambda_{in}}_1^{\mathrm{T}}, ..., \boldsymbol{\lambda_{eq}}_{n_\nu}^{\mathrm{T}}, \\ & \boldsymbol{\lambda_{eq}}_{1,2}^{\mathrm{T}}, ..., \boldsymbol{\lambda_{eq}}_{n_\nu-1,n_\nu}^{\mathrm{T}}, \boldsymbol{\lambda_{in}}_{1,2}^{\mathrm{T}}, ..., \boldsymbol{\lambda_{eq}}_{n_\nu-1,n_\nu}^{\mathrm{T}}, \boldsymbol{\nu}_1^{\mathrm{T}}, ..., \boldsymbol{\nu}_{n_\nu}^{\mathrm{T}}, \boldsymbol{\nu}_{1,2}^{\mathrm{T}}, ..., \boldsymbol{\nu}_{n_\nu-1,n_\nu}^{\mathrm{T}} \end{aligned} \tag{A.6}$$

The first order optimality condition (4.52) result to ($*.$ is referring to an element-wise multiplication)

$$\mathcal{L}_{\boldsymbol{w}} = \tag{A.7}$$
$$\begin{bmatrix}
\nabla_{\boldsymbol{u}_1} L_1 + (\nabla_{\boldsymbol{u}_1}\boldsymbol{f}_1)^{\mathrm{T}}\boldsymbol{\lambda}_1 + (\nabla_{\boldsymbol{u}_1}\boldsymbol{c}_{\boldsymbol{eq}_1})^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_1} + (\nabla_{\boldsymbol{u}_1}\boldsymbol{c}_{\boldsymbol{in}1})^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}1} \\
\ldots + \sum_j^{n_{\epsilon 1}} \left\langle \nabla_{\boldsymbol{u}_1} L_{1,j} + \left(\nabla_{\boldsymbol{u}_1}\boldsymbol{c}_{\boldsymbol{eq}_{1,j}}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{1j}} + (\nabla_{\boldsymbol{u}_1}\boldsymbol{c}_{\boldsymbol{in}1,j})^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}1,j}\right\rangle \\
\vdots \\
\nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu} + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{f}_{n_\nu}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{n_\nu} + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{c}_{\boldsymbol{eq}_{n_\nu}}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu}} - \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{c}_{\boldsymbol{in}n_\nu}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}n_\nu} \\
\ldots + \sum_j^{n_{\epsilon n_\nu}} \left\langle \nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu,j} + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{c}_{\boldsymbol{eq}_{n_\nu,j}}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu j}} + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{c}_{\boldsymbol{in}n_\nu,j}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}n_\nu j}\right\rangle \\
\boldsymbol{c}_{\boldsymbol{eq}_1} \\
\vdots \\
\boldsymbol{c}_{\boldsymbol{eq}_{n_\nu}} \\
\boldsymbol{c}_{\boldsymbol{eq}_{1,2}} \\
\vdots \\
\boldsymbol{c}_{\boldsymbol{eq}_{n_\nu-1,n_\nu}} \\
\boldsymbol{c}_{\boldsymbol{in}1} \\
\vdots \\
\boldsymbol{c}_{\boldsymbol{in}n_\nu} \\
\boldsymbol{c}_{\boldsymbol{in}1,2} \\
\vdots \\
\boldsymbol{c}_{\boldsymbol{in}n_\nu-1,n_\nu} \\
2\boldsymbol{\lambda}_{\boldsymbol{in}1} *.\boldsymbol{\varphi}_1 - \boldsymbol{\nu}_1 \\
\vdots \\
2\boldsymbol{\lambda}_{\boldsymbol{in}n_\nu} *.\boldsymbol{\varphi}_{n_\nu} - \boldsymbol{\nu}_{n_\nu} \\
2\boldsymbol{\lambda}_{\boldsymbol{in}1,2} *.\boldsymbol{\varphi}_{1,2} - \boldsymbol{\nu}_{1,2} \\
\vdots \\
2\boldsymbol{\lambda}_{\boldsymbol{in}n_\nu-1,n_\nu} *.\boldsymbol{\varphi}_{n_\nu-1,n_\nu} - \boldsymbol{\nu}_{n_\nu-1,n_\nu}
\end{bmatrix}$$

with the state derivative of the Lagrangian (4.51)

$$\mathcal{L}_{\boldsymbol{x}} = \tag{A.8}$$
$$\begin{bmatrix}
\nabla_{\boldsymbol{x}_1} L_1 + (\nabla_{\boldsymbol{x}_1}\boldsymbol{f}_1)^{\mathrm{T}}\boldsymbol{\lambda}_1 + (\nabla_{\boldsymbol{x}_1}\boldsymbol{c}_{\boldsymbol{eq}_1})^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_1} - (\nabla_{\boldsymbol{x}_1}\boldsymbol{c}_{\boldsymbol{in}1})^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}1} \\
\ldots + \sum_j^{n_{\epsilon 1}} \left\langle \nabla_{\boldsymbol{x}_1} L_{1,j} + \left(\nabla_{\boldsymbol{x}_1}\boldsymbol{c}_{\boldsymbol{eq}_{1,j}}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{1,j}} + (\nabla_{\boldsymbol{x}_1}\boldsymbol{c}_{\boldsymbol{in}1,j})^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}1,j}\right\rangle \\
\vdots \\
\nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu} + \left(\nabla_{\boldsymbol{x}_{n_\nu}}\boldsymbol{f}_{n_\nu}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{n_\nu} + \left(\nabla_{\boldsymbol{x}_{n_\nu}}\boldsymbol{c}_{\boldsymbol{eq}_{n_\nu}}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu}} - \left(\nabla_{\boldsymbol{x}_{n_\nu}}\boldsymbol{c}_{\boldsymbol{in}n_\nu}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}n_\nu} \\
\ldots + \sum_j^{n_{\epsilon n_\nu}} \left\langle \nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu,j} + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{c}_{\boldsymbol{eq}_{n_\nu,j}}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu j}} + \left(\nabla_{\boldsymbol{x}_{n_\nu}}\boldsymbol{c}_{\boldsymbol{in}n_\nu,j}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{in}n_\nu j}\right\rangle
\end{bmatrix}.$$

Global OCP with saturation function constraint handling

For the saturation function approach presented in §4.5.3, the global *OCP* is given by

$$\begin{aligned}\mathcal{L} = \sum_{i=1}^{n_\nu} \Bigg(& L_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) + \underline{\boldsymbol{\lambda}}_i^{\mathrm{T}} \boldsymbol{f}_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) + \underline{\boldsymbol{\lambda_{eq}}}_i^{\mathrm{T}} \boldsymbol{c_{eq}}_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right) \\ & + \sum_{l=1}^{n_{c_{in\,i}}} \left[\boldsymbol{sig}\left(\boldsymbol{c_{in}}_i\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{u}}_i\right), \underline{\boldsymbol{\kappa_A}}_i\right)\right] + \sum_{j \neq i}^{n_{\epsilon i}} \Big\langle L_{i,j}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j\right) \\ & + \underline{\boldsymbol{\lambda_{eq}}}_{i,j}^{\mathrm{T}} \boldsymbol{c_{eq}}_{i,j}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j\right) \\ & + \sum_{l=1}^{n_{c_{in\,i,j}}} \left[\boldsymbol{sig}\left(\boldsymbol{c_{in}}_{i,j}\left(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j, \underline{\boldsymbol{u}}_i, \underline{\boldsymbol{u}}_j\right), \underline{\boldsymbol{\kappa_A}}_{i,j}\right)\right] \Big\rangle \Bigg).\end{aligned} \tag{A.9}$$

As the saturation function approach does not introduce additional optimization variables, the global optimization variable vector is consequently concatenated to

$$\boldsymbol{w} = [\boldsymbol{u}_1^{\mathrm{T}}, ..., \boldsymbol{u}_{n_\nu}^{\mathrm{T}}, \boldsymbol{\lambda_{eq}}_1^{\mathrm{T}}, ..., \boldsymbol{\lambda_{eq}}_{n_\nu}^{\mathrm{T}}, \boldsymbol{\lambda_{eq}}_{1,2}^{\mathrm{T}}, ..., \boldsymbol{\lambda_{eq}}_{n_{\epsilon i}}^{\mathrm{T}}]^{\mathrm{T}} \tag{A.10}$$

Under use of the element-wise multiplication $*.$, the sigmoid derivative is given by:

$$\nabla_{\boldsymbol{x}} \boldsymbol{sig}\left(c_{in}\left(\boldsymbol{x}, \boldsymbol{u}\right), \boldsymbol{\kappa_A}\right) = \boldsymbol{\kappa_A} * .\boldsymbol{sig}\left(c_{in}\left(\boldsymbol{x}, \boldsymbol{u}\right), \boldsymbol{\kappa_A}\right) * .\left(1 - \boldsymbol{sig}\left(c_{in}\left(\boldsymbol{x}, \boldsymbol{u}\right), \boldsymbol{\kappa_A}\right)\right) * .\nabla_{\boldsymbol{x}} c_{in}\left(\boldsymbol{x}, \boldsymbol{u}\right) \tag{A.11}$$

$$\nabla_{\boldsymbol{u}} \boldsymbol{sig}\left(c_{in}\left(\boldsymbol{x}, \boldsymbol{u}\right) \boldsymbol{\kappa_A}\right) = \boldsymbol{\kappa_A} * .\boldsymbol{sig}\left(c_{in}\left(\boldsymbol{x}, \boldsymbol{u}\right), \boldsymbol{\kappa_A}\right) * .\left(1 - \boldsymbol{sig}\left(c_{in}\left(\boldsymbol{x}, \boldsymbol{u}\right), \boldsymbol{\kappa_A}\right)\right) * .\nabla_{\boldsymbol{u}} c_{in}\left(\boldsymbol{x}, \boldsymbol{u}\right) \tag{A.12}$$

In order to improve the readability, κ_A is omitted as sigmoid function argument in the following. Accordingly, the first order optimality condition (4.52) result to

$$
\mathcal{L}_{\boldsymbol{w}} = \tag{A.13}
$$

$$
\begin{bmatrix}
\nabla_{\boldsymbol{u}_1} L_1 + \left(\nabla_{\boldsymbol{u}_1}\boldsymbol{f}_1\right)^{\mathrm{T}}\boldsymbol{\lambda}_1 + \left(\nabla_{\boldsymbol{u}_1}\boldsymbol{ceq}_1\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}1} \\
\ldots - \left(\nabla_{\boldsymbol{u}_1}\boldsymbol{cin}_1\right)^{\mathrm{T}}\boldsymbol{\kappa}_{\boldsymbol{A}1} * .\boldsymbol{sig}\left(c_{in1}\right) * .\left(1-\boldsymbol{sig}\left(c_{in1}\right)\right) \\
\ldots + \sum_j^{n_{\epsilon 1}} \left\langle \nabla_{\boldsymbol{u}_1} L_{1,j} + \left(\nabla_{\boldsymbol{u}_1}\boldsymbol{ceq}_{1,j}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}1j}\right. \\
\left.\ldots + \left(\nabla_{\boldsymbol{u}_1}\boldsymbol{cin}_{1,j}\right)^{\mathrm{T}}\boldsymbol{\kappa}_{\boldsymbol{A}1,j} * .\boldsymbol{sig}\left(c_{in1,j}\right) * .\left(1-\boldsymbol{sig}\left(c_{in1,j}\right)\right)\right\rangle \\
\vdots \\
\nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu} + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{f}_{n_\nu}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{n_\nu} \\
\ldots + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{ceq}_{n_\nu}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu}} - \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{cin}_{n_\nu}\right)^{\mathrm{T}}\boldsymbol{\kappa}_{\boldsymbol{A}n_\nu} * .\boldsymbol{sig}\left(c_{inn_\nu}\right) * .\left(1-\boldsymbol{sig}\left(c_{inn_\nu}\right)\right) \\
\ldots + \sum_j^{n_{\epsilon n_\nu}} \left\langle \nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu,j} + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{ceq}_{n_\nu,j}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu}j}\right. \\
\left.\ldots + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{cin}_{n_\nu,j}\right)^{\mathrm{T}}\boldsymbol{\kappa}_{\boldsymbol{A}n_\nu,j} * .\boldsymbol{sig}\left(c_{inn_\nu,j}\right) * .\left(1-\boldsymbol{sig}\left(c_{inn_\nu,j}\right)\right)\right\rangle \\
\boldsymbol{cin}_1 \\
\vdots \\
\boldsymbol{cin}_{n_\nu} \\
\boldsymbol{cin}_{1,2} \\
\vdots \\
\boldsymbol{cin}_{n_\nu-1,n_\nu}
\end{bmatrix}
$$

with (4.51) the state derivative of the Lagrangian

$$
\mathcal{L}_{\boldsymbol{x}} = \tag{A.14}
$$

$$
\begin{bmatrix}
\nabla_{\boldsymbol{x}_1} L_1 + \left(\nabla_{\boldsymbol{x}_1}\boldsymbol{f}_1\right)^{\mathrm{T}}\boldsymbol{\lambda}_1 + \left(\nabla_{\boldsymbol{x}_1}\boldsymbol{ceq}_1\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}1} \\
\ldots - \left(\nabla_{\boldsymbol{x}_1}\boldsymbol{cin}_1\right)^{\mathrm{T}}\boldsymbol{\kappa}_{\boldsymbol{A}1} * .\boldsymbol{sig}\left(c_{in1}\right) * .\left(1-\boldsymbol{sig}\left(c_{in1}\right)\right) \backslash \boldsymbol{cin}_1 \\
\ldots + \sum_j^{n_{\epsilon 1}} \left\langle \nabla_{\boldsymbol{x}_1} L_{1,j}\right. \\
\left.\ldots + \left(\nabla_{\boldsymbol{x}_1}\boldsymbol{ceq}_{1,j}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}1,j} + \left(\nabla_{\boldsymbol{x}_1}\boldsymbol{cin}_{1,j}\right)^{\mathrm{T}}\boldsymbol{\kappa}_{\boldsymbol{A}1,j} * .\boldsymbol{sig}\left(c_{in1,j}\right) * .\left(1-\boldsymbol{sig}\left(c_{in1,j}\right)\right)\right\rangle \\
\vdots \\
\nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu} + \left(\nabla_{\boldsymbol{x}_{n_\nu}}\boldsymbol{f}_{n_\nu}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{n_\nu} \\
\ldots + \left(\nabla_{\boldsymbol{x}_{n_\nu}}\boldsymbol{ceq}_{n_\nu}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu}} - \left(\nabla_{\boldsymbol{x}_{n_\nu}}\boldsymbol{cin}_{n_\nu}\right)^{\mathrm{T}}\boldsymbol{\kappa}_{\boldsymbol{A}n_\nu} * .\boldsymbol{sig}\left(c_{inn_\nu}\right) * .\left(1-\boldsymbol{sig}\left(c_{inn_\nu}\right)\right) \\
\ldots + \sum_j^{n_{\epsilon n_\nu}} \left\langle \nabla_{\boldsymbol{u}_{n_\nu}} L_{n_\nu,j} + \left(\nabla_{\boldsymbol{u}_{n_\nu}}\boldsymbol{ceq}_{n_\nu,j}\right)^{\mathrm{T}}\boldsymbol{\lambda}_{\boldsymbol{eq}_{n_\nu}j}\right. \\
\left.\ldots + \left(\nabla_{\boldsymbol{x}_{n_\nu}}\boldsymbol{cin}_{n_\nu,j}\right)^{\mathrm{T}}\boldsymbol{\kappa}_{\boldsymbol{A}n_\nu,j} * .\boldsymbol{sig}\left(c_{inn_\nu,j}\right) * .\left(1-\boldsymbol{sig}\left(c_{inn_\nu,j}\right)\right)\right\rangle
\end{bmatrix}.
$$

Bibliography

[ACLV16] H. Abaunza, P. Castillo, R. Lozano, and A. Victorino (2016). *Quadrotor aerial manipulator based on dual quaternions. 2016 International Conference on Unmanned Aircraft Systems, ICUAS 2016*, pages 152–161.

[ACMP13] G. Arleo, F. Caccavale, G. Muscio, and F. Pierri (2013). *Control of quadrotor aerial vehicles equipped with a robotic arm. 2013 21st Mediterranean Conference on Control and Automation, MED 2013 - Conference Proceedings*, (1):1174–1180.

[AMA14] B. Alrifaee, M. G. Mamaghani, and D. Abel (2014). *Centralized non-convex model predictive control for cooperative collision avoidance of networked vehicles.* In 2014 IEEE International Symposium on Intelligent Control (ISIC), pages 1583–1588.

[ÁMMGC^{+}14] J. U. Álvarez-Muñoz, Nicolas Marchand, Fermi Guerrero-Castellanos, Sylvain Durand, and A. E. Lopez-Luna (2014). *Improving control of quadrotors carrying a manipulator arm.* In XVI Congreso Latinoamericano de Control Automático (CLCA 2014), page 6 p., Mexico.

[ANT10] Kostas Alexis, George Nikolakopoulos, and Anthony Tzes (2010). *Experimental model predictive attitude tracking control of a quadrotor helicopter subject to wind-gusts. 18th Mediterranean Conference (MED)*, pages 1461–1466.

[ASO15] J. A. Acosta, M. I. Sanchez, and A. Ollero (2015). *Robust control of underactuated Aerial Manipulators via IDA-PBC. Proceedings of the IEEE Conference on Decision and Control*, February 2015:673–678.

[ATDG10] F. L. Almeida, B. M. Terra, P. A. Dias, and G. M. Goncalves (2010). *Adoption Issues of Multi-agent Systems in Manufacturing Industry.* In 2010 Fifth International Multi-conference on Computing in the Global Information Technology, pages 238–244.

[AWF11] Heng-Bin An, Ju Wen, and Tao Feng (2011). *On finite difference approximation of a matrix-vector product in the Jacobian-free Newton-Krylov method. Journal of Computational and Applied Mathematics*, 236(6):1399 – 1409.

[AZR15] Giovanni Buizza Avanzini, Andrea Maria Zanchettin, and Paolo Rocco (2015). *Constraint-based Model Predictive Control for holonomic mobile manipulators. IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem(i):1473–1479.

[BCL15] Luca Rosario Buonocore, Jonathan Cacace, and Vincenzo Lippiello (2015). *Hybrid visual servoing for aerial grasping with hierarchical task-priority control. 23rd Mediterranean Conference on Control and Automation, MED 2015 - Conference Proceedings*, pages 617–623.

[Bea08] Rw Beard (2008). *Quadrotor dynamics and control. Brigham Young University*, pages 1–47.

[BGP$^+$15] K. Baizid, G. Giglio, F. Pierri, M. A. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero (2015). *Experiments on behavioral coordinated control of an Unmanned Aerial Vehicle manipulator system.* In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 4680–4685.

[BGP$^+$17] K. Baizid, G. Giglio, F. Pierri, M. A. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero (2017). *Behavioral control of unmanned aerial vehicle manipulator systems. Autonomous Robots*, 41(5):1203–1220.

[BH06] Sourabh Bhattacharya and Seth Hutchinson (2006). *Controllability and Properties of Optimal paths for a Differential Drive robot with field-of-view constraints. Proceedings - IEEE International Conference on Robotics and Automation*, 2006(May):1624–1629.

[BH09] Gerhard Bandow and Hartmut H. Holzmller (2009). *"Das ist gar kein Modell!" : Unterschiedliche Modelle und Modellierungen in Betriebswirtschaftslehre und Ingenieurwissenschaften.* Gabler, Wiesbaden.

[BKMZ16] Dmitry Bazylev, Artem Kremlev, Alexey Margun, and Konstantin Zimenko (2016). *Design of control system for a four-rotor UAV equipped with robotic arm. International Congress on Ultra Modern Telecommunications and Control Systems and Workshops*, 2016-January:144–149.

[BMCH07] Sourabh Bhattacharya, Rafael Murrieta-Cid, and Seth Hutchinson (2007). *Optimal paths for landmark-based navigation by differential-drive vehicles with field-of-view constraints. IEEE Transactions on Robotics*, 23(1):47–59.

[BMPL$^+$14] S. Bertrand, J. Marzat, H. Piet-Lahanier, A. Kahn, and Y. Rochefort (2014). *MPC Strategies for Cooperative Guidance of Autonomous Vehicles. AerospaceLab*, (8):1–18.

[BMS04] S. Bouabdallah, P. Murrieri, and R. Siegwart (2004). *Design and control of an indoor micro quadrotor.* In Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, volume 5, pages 4393–4398 Vol.5.

[BNS04] S. Bouabdallah, A. Noth, and R. Siegwart (2004). *PID vs LQ control techniques applied to an indoor micro quadrotor.* In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), volume 3, pages 2451–2456 vol.3.

[Bou12] Patrick Bouffard (2012). *On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments.* Thesis, Master Thesis, EECS Department, University of California, Berkeley.

[BR11] A. Bemporad and C. Rocchi (2011). *Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles.* In 2011 50th IEEE Conference on Decision and Control and European Control Conference, pages 7488–7493.

[BS05] S. Bouabdallah and R. Siegwart (2005). *Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor.* In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 2247–2252.

[BS07a] S. Bouabdallah and R. Siegwart (2007a). *Design and Control of a Miniature Quadrotor.* In Kimon P. Valavanis, editor, Advances in Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy, pages 171–210. Springer Netherlands, Dordrecht.

[BS07b] S. Bouabdallah and R. Siegwart (2007b). *Full control of a quadrotor.* In 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 153–158.

[BZM$^+$14] Dmitry Bazylev, Konstantin Zimenko, Alexey Margun, Alexey Bobtsov, and Artem Kremlev (2014). *Adaptive Control System for Quadrotor Equiped with Robotic Arm. Methods and Models in Automation and Robotics (MMAR)*, pages 705–710.

[CdQ15] X. Cai and M. d. Queiroz (2015). *Adaptive Rigidity-Based Formation Control for Multirobotic Vehicles With Dynamics. IEEE Transactions on Control Systems Technology*, 23(1):389–396.

[CGHC09] Rongxin Cui, S. S. Ge, B. V. E. How, and Yoo Sang Choo (2009). *Leader-follower formation control of underactuated AUVs with leader position measurement.* In IEEE International Conference on Robotics and Automation, pages 979–984.

[Cha07] Benoit Chachuat (2007). *Nonlinear and Dynamic Optimization: From Theory to Practice.* https://infoscience.epfl.ch/record/111939/files/Chachuat_07(IC32).pdf. Lecture notes, Accessed: 2018-06-03.

[CMPT08] Luca Consolini, Fabio Morbidi, Domenico Prattichizzo, and Mario Tosques (2008). *Leader-follower formation control of nonholonomic mobile robots with input constraints. Automatica*, 44(5):1343–1349.

[Cor13] Peter Corke (2013). *Robotics, Vision and Control.* 73 edition.

[DABS14] G. Darivianakis, K. Alexis, M. Burri, and R. Siegwart (2014). *Hybrid predictive control for aerial robotic physical interaction towards inspection operations.* In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 53–58.

[Den16] J. Dentler (2016). *DENMPC: An event-based real-time nonlinear model predictive control framework.* http://wiki.ros.org/dentnmpc. Accessed: 2017-03-10.

[DF08] W. Dong and J. A. Farrell (2008). *Formation control of multiple underactuated surface vessels. IET Control Theory Applications*, 2(12):1077–1085.

[DFA07] Moritz Diehl, Rolf Findeisen, and Frank Allgöwer (2007). *A Stabilizing Real-Time Implementation of Nonlinear Model Predictive Control.* In Real-Time PDE-Constrained Optimization, pages 25–52. Society for Industrial and Applied Mathematics.

[DFH09] Moritz Diehl, HansJoachim Ferreau, and Niels Haverbeke (2009). *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation.* In Lalo Magni, DavideMartino Raimondo, and Frank Allgöwer, editors, Nonlinear Model Predictive Control, volume 384 of Lecture Notes in Control and Information Sciences, pages 391–417. Springer Berlin Heidelberg.

[DFK$^+$02] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor (2002). *A vision-based formation control framework. IEEE Transactions on Robotics and Automation*, 18(5):813–825.

[DGS$^+$16] Wei Ding, M. R. Ganesh, R. N. Severinghaus, J. J. Corso, and D. Panagou (2016). *Real-time model predictive control for keeping a quadrotor visible on the camera field-of-view of a ground robot.* In 2016 American Control Conference (ACC), pages 2259–2264.

[Die14] Moritz Diehl (2014). *Lecture Notes on Optimal Control and Estimation.* http://www.syscop.de/files/2015ss/optcont/oce_script.pdf. Accessed: 2018-06-03.

[Die17] Moritz Diehl (2017). *Numerical Optimal Control*. `https://www.syscop.de/files/2017ss/NOC/script/book-NOCSE.pdf`. Accessed: 2018-06-03.

[DKB$^+$18] Jan Dentler, Somasundar Kannan, Souad Bezzaoucha, Miguel Angel Olivares-Mendez, and Holger Voos (2018). *Model predictive cooperative localization control of multiple UAVs using potential function sensor constraints. Autonomous Robots*, pages 1–26.

[DKMV16a] J. Dentler, S. Kannan, M. A. O. Mendez, and H. Voos (2016a). *A modularization approach for nonlinear model predictive control of distributed fast systems.* In 2016 24th Mediterranean Conference on Control and Automation (MED), pages 292–297.

[DKMV16b] J. Dentler, S. Kannan, M. A. O. Mendez, and H. Voos (2016b). *A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors.* In 2016 IEEE Multiconference on Systems and Control (MSC), pages 519–525.

[DKMV16c] J. Dentler, S. Kannan, M. A. Olivares Mendez, and H. Voos (2016c). *A tracking error control approach for model predictive position control of a quadrotor with time varying reference.* In 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 2051–2056.

[DKMV17] J. Dentler, S. Kannan, M. A. Olivares Mendez, and H. Voos (2017). *Implementation and Validation of an Event-Based Real-Time Nonlinear Model Preditive Control Framework with ROS Interface for Single and Multi-robot Systems.* In 2017 IEEE 1st IEEE Conference on Control Technology and Applications (CCTA).

[DMD$^+$18] J. Dentler, M.Rosalie, G. Danoy, P. Bouvry, S. Kannan, M. A. Olivares Mendez, and H. Voos (2018). *Collision avoidance effects on the mobility of a UAV swarm using Chaotic Ant Colony with Model Predictive Control.*

[DO14a] Todd W. Danko and Paul Y. Oh (2014a). *Evaluation of visual servoing control of aerial manipulators using test gantry emulation. 2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, pages 821–829.

[DO14b] Todd W. Danko and Paul Y. Oh (2014b). *Toward coordinated manipulator-host visual servoing for mobile manipulating UAVs. IEEE Conference on Technologies for Practical Robot Applications, TePRA.*

[DOK98] J. P. Desai, J. Ostrowski, and V. Kumar (1998). *Controlling formations of multiple mobile robots.* In Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146), volume 4, pages 2864–2869 vol.4.

[dRB08] Hans de Ruiter and Beno Benhabib (2008). *Visual-model-based, real-time 3D pose tracking for autonomous navigation: methodology and experiments. Autonomous Robots*, 25(3):267–286.

[DSH07] J C Derenick, J R Spletzer, and M A Hsieh (2007). *A graph theoretic approach to optimal target tracking for mobile robot teams. Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 3422–3428.

[EIB12] Ö. Erkent and H. Işıl Bozma (2012). *Artificial potential functions based camera movements and visual behaviors in attentive robots. Autonomous Robots*, 32(1):15–34.

[FBC13] P. Falcón, A. Barreiro, and M. D. Cacho (2013). *Modeling of Parrot Ardrone and passivity-based reset control.* In 9th Asian Control Conference (ASCC), pages 1–6.

[FKM13] H. Fukushima, K. Kon, and F. Matsuno (2013). *Model Predictive Formation Control Using Branch-and-Bound Compatible With Collision Avoidance Problems. IEEE Transactions on Robotics*, 29(5):1308–1317.

[FM04] J. A. Fax and R. M. Murray (2004). *Information flow and cooperative control of vehicle formations. IEEE Transactions on Automatic Control*, 49(9):1465–1476.

[Gas13] (2013). *Active Set Methods.* In Saul I. Gass and Michael C. Fu, editors, Encyclopedia of Operations Research and Management Science, pages 2–2. Springer US, Boston, MA.

[GC02] S.S. Ge and Y.J. Cui (2002). *Dynamic Motion Planning for Mobile Robots Using Potential Field Method. Autonomous Robots*, 13(3):207–222.

[GDDSL12] L. Garcia-Delgado, A. Dzul, V. Santibanez, and M. Llama (2012). *Quad-rotors formation based on potential functions with obstacle avoidance. IET Control Theory Applications*, 6(12):1787–1802.

[GDLP13] García, Dzul, Lozano, and Pégard (2013). *Vision-Based Hovering and Navigation.*

[GK15] G. Garimella and M. Kobilarov (2015). *Towards model-predictive control for aerial pick-and-place.* In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 4692–4697.

[GKPC10] Knut Graichen, Andreas Kugi, Nicolas Petit, and Francois Chaplais (2010). *Handling constraints in optimal control with saturation functions and system extension. Systems & Control Letters.*

[GM16] Mathieu Geisert and Nicolas Mansard (2016). *Trajectory Generation for Quadrotor Based Systems using Numerical Optimal Control. International Conference on Robotics and Automation*, pages 2958–2964.

[GP11] Veysel Gazi and Kevin M. Passino (2011). *Swarm Coordination and Control Problems.* In Swarm Stability and Optimization, pages 15–25. Springer Berlin Heidelberg, Berlin, Heidelberg.

[GP17] Lars Gruene and Juergen Pannek (2017). *Nonlinear Model Predictive Control: Theory and Algorithms.* Communications and Control Engineering. Springer International Publishing.

[Gra13] Knut Graichen (2013). *Methoden der Optimierung und optimalen Steuerung.* `https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.110/Downloads/Vorlesung/Optimierung/Skript/Skript_MOOS_WS1213.pdf`. Accessed: 2018-06-03.

[GU14] Knut Graichen and Tilman Utz (2014). *GRAMPC documentation.* `https://sourceforge.net/projects/grampc/`. Accessed: 2018-06-03.

[Gut07] Martin H. Gutknecht (2007). *A Brief Introduction to Krylov Space Methods for Solving Linear Systems.* In Yukio Kaneda, Hiroshi Kawamura, and Masaki Sasai, editors, Frontiers of Computational Science, pages 53–62, Berlin, Heidelberg. Springer Berlin Heidelberg.

[HD11] M. Hehn and R. D'Andrea (2011). *A flying inverted pendulum.* In 2011 IEEE International Conference on Robotics and Automation, pages 763–770.

[Hen77] D.M. Planning Henderson (1977). *Euler angles, quaternions, and transformation matrices for space shuttle analysis. NASA JSC Report*, 12960.

[HHWT07] Gabriel M Hoffmann, Haomiao Huang, Steven L Waslander, and Claire J Tomlin (2007). *Quadrotor helicopter flight dynamics and control: Theory and experiment.* In Proc. of the AIAA Guidance, Navigation, and Control Conference, volume 2, page 4.

[HMLO02] T Hamel, R Mahony, R Lozano, and J Ostrowski (2002). *Dynamic modelling and configuration stabilisation for an X4-flyer. IFAC World Congress*, 1:200–212.

[HNB+15] Mike Huang, Hayato Nakada, Butts, Kenneth R., and Ilya V. Kolmanovsky (2015). *Nonlinear Model Predictive Control of a Diesel Engine Air Path: A Comparison of Constraint Handling and Computational Strategies. Proceedings - 5th IFAC Conference on Nonlinear Model Predictive Control - Seville, Spain.*

[Hof16] Patrick Hoffmann (2016). *Development of a controller for a lightweight manipulator.* Thesis, University of Luxembourg.

[HSV95] Richard F. Hartl, Suresh P. Sethi, and Raymond G. Vickson (1995). *A Survey of the Maximum Principles for Optimal Control Problems with State Constraints. SIAM Review*, 37(2):181–218.

[JCBHO15] A. E. Jimenez-Cano, J. Braga, G. Heredia, and A. Ollero (2015). *Aerial manipulator for structure inspection by contact from the underside. IEEE International Conference on Intelligent Robots and Systems*, 2015-December:1879–1884.

[JHA+13] M. K. Joyo, D. Hazry, S. Faiz Ahmed, M. H. Tanveer, F. A. Warsi, and A. T. Hussain (2013). *Altitude and horizontal motion control of quadrotor UAV in the presence of air turbulence.* In IEEE Conference on Systems, Process Control (ICSPC), pages 16–20.

[Jos17] Divya Joshi (2017). *Exploring the latest drone technology for commercial, industrial and military drone uses. Business Insider UK.*

[KAOMV14] Somasundar Kannan, Marouane Alma, Miguel A. Olivares-Mendez, and Holger Voos (2014). *Adaptive control of Aerial Manipulation Vehicle. Proceedings - 4th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2014*, (November):273–278.

[KCK13] Suseong Kim, Seungwon Choi, and H. Jin Kim (2013). *Aerial manipulation using a quadrotor with a two DOF robotic arm. IEEE International Conference on Intelligent Robots and Systems*, pages 4990–4995.

[Kel95] C. Kelley (1995). *Iterative Methods for Linear and Nonlinear Equations.* Society for Industrial and Applied Mathematics.

[Kel99] C. Kelley (1999). *Iterative Methods for Optimization.* Society for Industrial and Applied Mathematics.

[KK04] D.A. Knoll and D.E. Keyes (2004). *Jacobian-free Newton-Krylov methods: a survey of approaches and applications. Journal of Computational Physics*, 193(2):357 – 397.

[KKP09] Jinhyun Kim, Min-Sung Kang, and Sangdeok Park (2009). *Accurate Modeling and Robust Hovering Control for a Quad-rotor VTOL Aircraft. Journal of Intelligent and Robotic Systems*, 57(1):9.

[KNFL09] Farid Kendoul, Kenzo Nonami, Isabelle Fantoni, and Rogelio Lozano (2009). *An adaptive vision-based autopilot for mini flying machines guidance, navigation and control. Autonomous Robots*, 27(3):165.

[KOO14] Christopher Korpela, Matko Orsag, and Paul Oh (2014). *Towards valve turning using a dual-arm aerial manipulator.* In IEEE International Conference on Intelligent Robots and Systems.

[KQGD$^+$16] S. Kannan, S. Quintanar-Guzman, J. Dentler, M. A. Olivares-Mendez, and H. Voos (2016). *Control of aerial manipulation vehicle in operational space.* In 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), pages 1–4.

[Kra88] D. Kraft (1988). *A Software Package for Sequential Quadratic Programming.* Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR.

[KSJ14] Mingxin Kang, Tielong Shen, and Xiaohong Jiao (2014). *Continuation/GMRES Method based Nonlinear Model Predictive Control for IC Engines.* pages 5697–5702.

[KSX15] James R. Kutia, Karl A. Stol, and Weiliang Xu (2015). *Canopy sampling using an aerial manipulator: A preliminary study. 2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, pages 477–484.

[LAR$^+$09] D. Limon, T. Alamo, D. M. Raimondo, D. Muñoz de la Peña, J. M. Bravo, A. Ferramosca, and E. F. Camacho (2009). *Input-to-State Stability: A Unifying Framework for Robust Model Predictive Control.* In Lalo Magni, Davide Martino Raimondo, and Frank Allgöwer, editors, Nonlinear Model Predictive Control: Towards New Challenging Applications, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg.

[LBY03] J. R. T. Lawton, R. W. Beard, and B. J. Young (2003). *A decentralized approach to formation maneuvers. IEEE Transactions on Robotics and Automation*, 19(6):933–941.

[Li14] Q. Li (2014). *Grey-Box System Identification of a Quadrotor Unmanned Aerial Vehicle.* Thesis, Delft University of Technology.

[Liu03] Xiaotao Liu (2003). *A Barrier Algorithm for Large Nonlinear Optimization Problems.* Dissertation, Standford University.

[LL17] M. M. Lau and K. H. Lim (2017). *Investigation of activation functions in deep belief network.* In 2017 2nd International Conference on Control and Robotics Engineering (ICCRE), pages 201–206.

[LLM12] T. Lee, M. Leok, and N. H. McClamroch (2012). *Nonlinear robust tracking control of a quadrotor UAV on SE(3).* In 2012 American Control Conference (ACC), pages 4649–4654.

[LNGB+10] Gonzalo López-Nicolás, Nicholas R. Gans, Sourabh Bhattacharya, Carlos Sagés, Josechu J. Guerrero, and Seth Hutchinson (2010). *Homography-based control scheme for mobile robots with nonholonomic and field-of-view constraints. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(4):1115–1127.

[LQS+11] Renato Vilela Lopes, Pedro Henrique De Rodrigues Quemel, Assis Santana, Araújo Borges, and Yoshiyuki Ishihara (2011). *Model Predictive Control Applied to Tracking and Attitude Stabilization of a VTOL Quadrotor Aircraft.*

[Ltd] DJI Innovations Technology Co. Ltd. *DJI: Matrice 100 Specifications.* `https://www.dji.com/matrice100/info`. Accessed: 2017-07-31.

[MDD+17] M.Rosalie, J. Dentler, G. Danoy, P. Bouvry, S. Kannan, M. A. Olivares Mendez, and H. Voos (2017). *Area exploration with a swarm of UAVs combining deterministic Chaotic Ant Colony Mobility with position MPC.* In 2017 IEEE 2017 International Conference on Unmanned Aircraft Systems (ICUAS).

[Meo17] A. Meola (Juli 2017). *Drone market shows positive outlook with strong industry growth and trends.* `http://uk.businessinsider.com/drone-industry-analysis-market-trends-growth-forecasts-2017-7?r=US&IR=T`. Accessed: 2018-06-01.

[MGS11] Hasan Mehrjerdi, Jawhar Ghommam, and Maarouf Saad (2011). *Nonlinear coordination control for a group of mobile robots using a virtual structure. Mechatronics*, 21(7):1147 – 1155.

[MLS15] Rafik Mebarki, Vincenzo Lippiello, and Bruno Siciliano (2015). *Toward image-based visual servoing for cooperative aerial manipulation. IEEE International Conference on Robotics and Automation (ICRA).*

[Mos15] K. Moskvitch (2015). *Take off: are drones the future of farming? Engineering Technology*, 10(7-8):62–66.

[MPT+16] G. Muscio, F. Pierri, M. A. Trujillo, E. Cataldi, G. Giglio, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero (2016). *Experiments on coordinated motion of aerial robotic manipulators.* In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1224–1229.

[Neg14] (2014). *Distributed MPC Via Dual Decomposition.* In José M. Maestre and Rudy R. Negenborn, editors, Distributed Model Predictive Control Made Easy. Springer Netherlands, Dordrecht.

[NMSC13] Tiago P. Nascimento, Antonio Paulo Moreira, and André G. Scolari Conceiçao (2013). *Multi-robot nonlinear model predictive formation control: Moving target and target absence. Robotics and Autonomous Systems*, 61(12):1502 – 1515.

[Noc06] (2006). *Fundamentals of Unconstrained Optimization.* In Numerical Optimization, pages 10–29. Springer New York.

[NSCM14] Tiago P. Nascimento, André G. Scolari Conceiçao, and Antonio Paulo Moreira (2014). *Iterative weighted tuning for a nonlinear model predictive formation control.* In 2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pages 2–7.

[Oht] T. Ohtsuka. *Symlab: cgmres source code.* `http://www.ids.sys.i.kyoto-u.ac.jp/~ohtsuka/code/index.htm`. Accessed: 2017-02-26.

[Oht04] Toshiyuki Ohtsuka (2004). *A continuation/GMRES method for fast computation of nonlinear receding horizon control. Automatica*, 40(4):563–574.

[OKPO13] M Orsag, C Korpela, M Pekala, and P Oh (2013). *Stability control in aerial manipulation. 2013 American Control Conference.*

[OmKV14] Miguel A Olivares-mendez, Somasundar Kannan, and Holger Voos (2014). *V-REP & ROS Testbed for Design, Test, and Tuning of a Quadrotor Vision Based Fuzzy Control System for Autonomous Landing.* pages 172–179.

[OSB16] M. Odelga, P. Stegagno, and H. H. Bülthoff (2016). *Obstacle detection, tracking and avoidance for a teleoperated UAV.* In IEEE International Conference on Robotics and Automation (ICRA), pages 2984–2990.

[Pan15] G. Pannocchia (2015). *Offset-free tracking MPC: A tutorial review and comparison of different formulations.* In Control Conference (ECC), 2015 European, pages 527–532.

[Pat16] Prachi Patel (2016). *Agriculture Drones Are Finally Cleared for Takeoff. IEEE Spectrum.*

[PDKV17] P.Kremer, J. Dentler, S. Kannan, and H. Voos (2017). *Cooperative Localization of Unmanned Aerial Vehicles in ROS - The Atlas Node.* In 15th IEEE International Conference on Industrial Informatics (INDIN).

[Pou07] Paul Edward Ian Pounds (September 2007). *Design, Construction and Control of a Large Quadrotor Micro Air Vehicle.* Dissertation, The Australian National University.

[QCG$^+$09] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng (2009). *ROS: an open-source Robot Operating System.* In ICRA Workshop on Open Source Software.

[RD83] S. Richter and R. DeCarlo (1983). *Continuation methods: Theory and applications. IEEE Transactions on Automatic Control*, 28(6):660–665.

[ROR08] Guilherme V. Raffo, Manuel G. Ortega, and Francisco R. Rubio (2008). *MPC with Nonlinear H-Infinity Control for Path Tracking of a Quad-Rotor Helicopter. IFAC Proceedings Volumes*, 41(2):8564 – 8569. 17th IFAC World Congress.

[ROR10] Guilherme V. Raffo, Manuel G. Ortega, and Francisco R. Rubio (2010). *An integral predictive/nonlinear control structure for a quadrotor helicopter. Automatica*, 46(1):29 – 39.

[Sas99] S. Sastry (1999). *Nonlinear Systems: Analysis, Stability, and Control.* Interdisciplinary Applied Mathematics. Springer New York.

[SAVD16] S. A. Sajadi-Alamdari, H. Voos, and M. Darouach (2016). *Nonlinear model predictive extended eco-cruise control for battery electric vehicles.* In 2016 24th Mediterranean Conference on Control and Automation (MED), pages 467–472.

[SBSF16] Lucas Vago Santana, Alexandre Santos Brandão, and Mário Sarcinelli-Filho (2016). *Navigation and Cooperative Control Using the AR.Drone Quadrotor. Journal of Intelligent & Robotic Systems*, 84(1):327–350.

[SF16] Juraj Stevek and Miroslav Fikar (2016). *Teaching Aids for Laboratory Experiments with AR.Drone2 Quadrotor. IFAC-PapersOnLine*, 49(6):236 – 241. 11th IFAC Symposium on Advances in Control Education ACE 2016.

[SK09] J. Shin and H. J. Kim (2009). *Nonlinear Model Predictive Formation Flight. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(5):1116–1125.

[SKK17a] H. Seo, S. Kim, and H. J. Kim (2017a). *Aerial grasping of cylindrical object using visual servoing based on stochastic model predictive control.* In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 6362–6368.

[SKK17b] H. Seo, S. Kim, and H. J. Kim (2017b). *Aerial grasping of cylindrical object using visual servoing based on stochastic model predictive control.* In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 6362–6368.

[SO02] H. Seguchi and T. Ohtsuka (2002). *Nonlinear receding horizon control of an RC hovercraft. Proceedings of the International Conference on Control Applications*, 2:1076–1081.

[SO03] Hiroaki Seguchi and Toshiyuki Ohtsuka (2003). *Nonlinear receding horizon control of an underactuated hovercraft. International journal of robust and nonlinear control*, 13(3-4):381–398.

[SO10] Y. Shimizu and T. Ohtsuka (2010). *A real-time algorithm for nonlinear receding horizon control of descriptor systems.* In SICE Annual Conference 2010, Proceedings of, pages 219–222.

[SOD06] Y. Shimizu, T. Ohtsuka, and M. Diehl (2006). *Nonlinear receding horizon control of an underactuated hovercraft with a multiple-shooting-based algorithm.* In Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE, pages 603–607.

[SOD09] Yuichi Shimizu, Toshiyuki Ohtsuka, and Moritz Diehl (2009). *A real-time algorithm for nonlinear receding horizon control using multiple shooting and continuation/Krylov method. International Journal of Robust and Nonlinear Control*, 19(8):919–936.

[SPA16] SPARC (2016). *Robotics 2020 Multi-Annual Roadmap. Horizon 2020 Call ICT-2017.*

[ST14] Justin M. Selfridge and Gang Tao (2014). *A multivariable adaptive controller for a quadrotor with guaranteed matching conditions. Proceedings of the American Control Conference*, (November):26–31.

[STSK13] N. Shakev, A. V. Topalov, K. Shiev, and O. Kaynak (2013). *Stabilizing multiple sliding surface control of quad-rotor rotorcraft.* In Control Conference (ASCC), 2013 9th Asian, pages 1–6.

[TMK12] M. Turpin, N. Michael, and V. Kumar (2012). *Decentralized formation control with variable shapes for aerial robots.* In 2012 IEEE International Conference on Robotics and Automation, pages 23–30.

[WLY15] Jian Wang, Jing-yang Liu, and Hong Yi (2015). *Formation control of unmanned surface vehicles with vision sensor constraints. OCEANS 2015-MTS/IEEE Washington*, pages 1–8.

[WS16] Kenneth J. Waldron and James Schmiedeler (2016). *Kinematics.* In Bruno Siciliano and Oussama Khatib, editors, Springer Handbook of Robotics, pages 11–36. Springer International Publishing, Cham.

[ZM13] Y. Zhang and H. Mehrjerdi (2013). *A survey on multiple unmanned vehicles formation control and coordination: Normal and fault situations*. In International Conference on Unmanned Aircraft Systems (ICUAS), pages 1087–1096.

[ZZZ16] Z. G. Zhou, Y. A. Zhang, and D. Zhou (2016). *Geometric modeling and control for the full-actuated aerial manipulating system*. In 35th Chinese Control Conference (CCC), pages 6178–6182.