Technische Universität München
Fakultät für Informatik

Lehrstuhl für Sensorbasierte Robotersysteme
und Intelligente Assistenzsysteme

Coordinated Control for Robot-assisted Take-off and Landing of Flying Robots

Dipl.-Ing. (Univ.) Moritz Andreas Maier

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.) genehmigten Dissertation.

Vorsitzende(r):	Prof. Dr. Daniel Cremers
Prüfende(r) der Dissertation:	
	1. Prof. Dr.-Ing. Alin Albu-Schäffer
	2. Prof. Dr.-Ing. Florian Holzapfel

Die Dissertation wurde am 11.05.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 28.09.2020 angenommen.

Acknowledgments

This thesis is the result of my research endeavor over the past 6.5 years at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR). This work would not have been possible without the help of many incredible people who I had the pleasure of working with. First and foremost, I would like to thank Prof. Alin Albu-Schäffer, for supervising and examining this thesis, and Prof. Florian Holzapfel, who made himself available as the second examiner and who put me in touch with the drone industry. I am very grateful to my department head, Dr. Christian Ott, for the trust he has placed in me, for his support, and for his inspiring ideas. Special thanks to Dr. Konstantin Kondak for creating the opportunity to work on this topic, initially through a Munich Aerospace PhD scholarship.

Next, I want to thank Dr. Thomas Hulin and Dr. Mikel Sagardia, whose work in the telepresence group brought me to DLR as a working student in 2010. Sincere thanks also to Dr. Teodor Tomić and Prof. Sami Haddadin for supervising my Diploma thesis in 2013. Furthermore, I would like to thank Felix Huber for teaching me how to operate the DLR Light Weight Robot (LWR), Maximilian Laiacker for providing an implementation of the marker tracking algorithm used in this work, and Dr. Teodor Tomić and Philipp Lutz for sharing their wisdom about multicopters, programming, and electronics. Big thanks also to Florian Schmidt and Robert Burger for providing and maintaining the excellent software packages that make our robots work, to Alexander Beyer and Martin Heumos for fixing the LWR when it was broken, and to all the members of the mechanical and electronics workshops for always being eager to help. I am also grateful to my former students (in the order of their appearance at DLR) Viet Duc Pham, André Oeschger, Daniel Martínez García, Patrick Jiskra, and Fabian Beck for their enthusiasm and valuable inputs.

I also would like to thank my DLR office mates Dr. Roman Weitschat, Nico Mansfeld, Felix Huber, Dr. Matteo Saveriano, Clemens Karl, Alexander Werner, and Samantha Stoneman for all the great discussions and for creating a wonderful working atmosphere. My profound appreciation to Nico Mansfeld, Dr. Alexander Dietrich, Manuel Keppler, Dr. Teodor Tomić, Dr. Christian Ott, Fabian Beck, and Dr. Katharina Hertkorn, who have proofread my manuscript. Many thanks also to my parents Wilma and Hans-Peter for their continuous support. Finally, I am deeply grateful to my spouse Katharina, for her love as well as for her scientific advice and for her encouragement, and to our two daughters Luise and Josefine for their curiosity and for all the joy they bring into my life.

Munich, May 2020 Moritz Andreas Maier

Abstract

Up to date, state-of-the-art controllers for flying robots or Unmanned Aerial Vehicles (UAVs) are not prepared for tasks involving physical interaction, dynamically varying payload, or coordination with other robotic systems. For example, take-off or landing on a moving platform, e.g. a ship or a mobile robot, is challenging. The flying robot has to synchronize its movements to the motion of the platform. An estimate of the latter is uncertain or may not be available at all. The UAV has to react quickly, which is energy consuming and can lead to actuator saturation. This gets even worse if the aerial vehicle carries a payload or is under the influence of wind and turbulences. To ensure safe and robust take-off and landing, assistance systems for UAVs have been proposed in the literature. However, the available approaches provide insufficient support for the flying robot, lack a docking interface, do not take the distributed control system into account, and are only concept studies.

In this work, a robotic assistance system for take-off and landing of flying robots is developed. The support system is based on a robot manipulator mounted on the landing surface. It allows to autonomously release or capture a UAV and to perform assisted take-off or landing. The main contributions of this thesis are the development of suitable dynamics models and coordinated control approaches for flying robots and robot manipulators, the realization of an assistance system prototype, and the implementation and evaluation of the presented controllers in simulations and experiments.

Regarding the realization of the support system, a universal hinge mechanism for the connection between flying robot and robot manipulator is introduced. It leaves the rotational degrees of freedom of the flying robot open and results in a redundant system. It is shown that the redundancy allows to fulfill additional objectives, for example to reduce the workload of the manipulator. In this case, both systems jointly contribute to the assistance task, which is especially useful for heavy UAVs. Capturing the flying robot by means of the robot manipulator requires accurate target tracking. Suitable localization systems and a sensor fusion algorithm are evaluated and different docking interface designs are discussed.

The developed coordinated controllers consider the different dynamical properties and actuation principles of flying robots and robot manipulators. In particular, three dynamics representations are used yielding controllers with varying complexity and performance: the independent dynamics, the combined dynamics, and the decomposed dynamics of flying robot and manipulator. To provide safe physical interaction, the well-known concept of impedance control is utilized. For flying robot control, established control laws are combined with external wrench observers, a novel control allocation procedure, and a novel adaptive control approach. This results in increased robustness against contact forces, actuator saturation, and changes in atmosphere or payload while providing more accurate trajectory tracking, as verified in the experiments. For a quadrocopter with fully symmetric fixed-pitch propellers, this work shows the first autonomous flight using bidirectional thrust which drastically increases control authority. This demonstrates the potential of optimized hardware in a combined assistance system.

Zusammenfassung

Nach heutigem Stand der Technik sind die Regler von Flugrobotern oder unbemannten Luftfahrzeugen (UAVs) nicht für Aufgaben vorbereitet, die physikalische Interaktion, dynamischen Wechsel der Nutzlast oder Koordination mit anderen Robotersystemen beinhalten. So ist beispielsweise der Start oder die Landung auf einer beweglichen Plattform, wie z.B. einem Schiff oder einem mobilen Roboter, eine große Herausforderung. Der Flugroboter muss dabei seine Bewegung mit der Plattform synchronisieren, wobei die Schätzung der Plattformposition und -lage ungenau oder möglicherweise nicht verfügbar ist. Das UAV muss zusätzlich schnell reagieren, was energieaufwändig ist und zu einer Sättigung der Aktuatoren führen kann. Die Landung wird noch erschwert, wenn das UAV eine Nutzlast trägt oder starke Windböen oder Turbulenzen herrschen. Um sicheres Abheben und Landen zu gewährleisten, wurden in der Literatur Assistenzsysteme für UAVs vorgeschlagen. Diese Ansätze bieten jedoch keine ausreichende Unterstützung für den Flugroboter, verfügen über keine Andockschnittstelle, berücksichtigen das verteilte Regelungssystem nicht und sind lediglich Konzeptstudien.

In dieser Arbeit wird ein robotisches Assistenzsystem für den Start und die Landung von Flugrobotern entwickelt. Es basiert auf einem Roboterarm, der auf der Landefläche montiert ist und einem UAV autonome Hilfestellung bei Start und Landung bietet. Die wesentlichen Beiträge dieser Arbeit sind die Entwicklung geeigneter Dynamikmodelle und koordinierter Regelungsansätze für Flugroboter und Roboterarme, die Realisierung eines Prototypen sowie die Implementierung und Evaluierung der vorgestellten Regelungsansätze in Simulationen und Experimenten.

Hinsichtlich der Realisierung des Assistenzsystems wird ein Kugelgelenk für die Verbindung zwischen Flugroboter und Roboterarm vorgestellt. Das Gelenk lässt die Rotationsfreiheitsgrade des Flugroboters frei und führt zu einem redundanten Gesamtsystem. Es wird gezeigt, dass durch die Redundanz zusätzliche Ziele erreicht werden können, z.B. um die Arbeitsbelastung des Roboterarms zu reduzieren. In diesem Fall tragen beide Systeme gemeinsam zur Erfüllung der Aufgabe bei, was speziell bei schweren UAVs notwendig ist. Das Einfangen des Flugroboters mit Hilfe des Roboterarms erfordert eine genaue Zielverfolgung. Geeignete Lokalisierungssysteme sowie ein Algorithmus zur Sensordatenfusion werden evaluiert und verschiedene Designs von Andockschnittstellen werden diskutiert. Die entwickelten koordinierten Regler berücksichtigen die unterschiedlichen dynamischen Eigenschaften und Antriebsprinzipien von Flugrobotern und Roboterarmen. Insbesondere werden drei Darstellungen der Dynamik verwendet, die Regler mit unterschiedlicher Komplexität und Performanz liefern: die unabhängige Dynamik, die kombinierte Dynamik und die zerlegte Dynamik von Flugroboter und Roboterarm. Um eine sichere physikalische Interaktion zu gewährleisten, wird das bekannte Konzept der Impedanzregelung verwendet. Für die Regelung des Flugroboters werden etablierte Regelgesetze mit einem Beobachter für externe Kräfte und Drehmomente, einem neuen Ansatz zur Zuweisung der Stellgrößen, sowie einem neuartigen adaptiven Regelungsansatz kombiniert. Das führt zu gesteigerter Robustheit gegenüber Kontaktkräften, Aktuatorsättigung und Änderungen der Atmosphäre oder Nutzlast während die Genauigkeit der Trajektorienfolge erhöht wird, wie in den Experimenten verifiziert werden konnte. Für einen Quadrokopter mit vollsymmetrischen, starren Propellern zeigt diese Arbeit den ersten autonomen Flug mit bidirektionalem Schub, welcher die zur Verfügung stehende Stellgröße drastisch erhöht. Dies zeigt das Potenzial optimierter Hardware in einem kombinierten Assistenzsystem.

Contents

Acronyms

BLDC	Brushless Direct Current
CAN	Controller Area Network
CCW	Counter Clockwise
CFD	Computational Fluid Dynamics
CM	Center of Mass
CoG	Center of Gravity
CPU	Central Processing Unit
CW	Clockwise
DC	Direct Current
DGNSS	Differential Global Navigation Satellite System
DGPS	Differential Global Positioning System
DLR	Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center)
DMP	Dynamic Movement Primitive
DOB	Disturbance Observer
DoF	Degrees of Freedom
EE	End-effector
EKF	Extended Kalman Filter
ENU	East North Up
ESC	Electronic Speed Controller
FLU	Forward Left Up
FOC	Field Oriented Control
FPGA	Field Programmable Gate Array
FTS	Force Torque Sensor
GPIO	General Purpose Input Output
GPOPS	General Purpose Optimal Control Software
GPS	Global Positioning System
IDA-PBC	Interconnection and Damping Assignment Passivity-based Control
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IGE	In Ground Effect
IMU	Inertial Measurement Unit
INDI	Incremental Nonlinear Dynamic Inversion
ISO	International Organization for Standardization
JONSWAP	Joint North Sea Wave Project
LAN	Local Area Network
LVDS	Low Voltage Differential Signaling
LWR	Light Weight Robot

MAV	Micro Aerial Vehicle
MIAC	Model Identification Adaptive Control
MIT	Massachusetts Institute of Technology
MPC	Model Predictive Control
MRAC	Model Reference Adaptive Control
MSE	Mean Squared Error
MSL	Mean Sea Level
MSS	Marine Systems Simulator
NDI	Nonlinear Dynamic Inversion
NED	North East Down
OGE	Out of Ground Effect
PD	Proportional Derivative
PID	Proportional Integral Derivative
PTP	Precision Time Protocol
PWM	Pulse Width Modulation
QP	Quadratic Programming
RAFCON	RMC Advanced Flow Control
RAM	Random Access Memory
RANSAC	Random Sample Consensus
RC	Remote Control
RMC	Robotics and Mechatronics Center
RMSE	Root Mean Square Error
ROS	Robot Operating System
RPM	Revolutions Per Minute
RTD	Round Trip Delay
RTK	Real Time Kinematics
SAR	Search and Rescue
SDK	Software Development Kit
SGM	Semi-Global Matching
SPI	Serial Peripheral Interface
SQP	Sequential Quadratic Programming
TCP	Tool Center Point
TDPC	Time Domain Passivity Control
TPP	Tip Path Plane
TUM	Technical University of Munich
UART	Universal Asynchronous Receiver Transmitter
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
USB	Universal Serial Bus
UWB	Ultra Wide Band
VIO	Visual Inertial Odometry
VO	Visual Odometry
VTOL	Vertical Take-off and Landing
WLAN	Wireless Local Area Network

List of Symbols

In this thesis, scalar quantities are written as plain upper and lower case letters, for example λ, k, N. Vectors and matrices are indicated with bold lower and upper case letters (e.g. $\boldsymbol{x}$, $\boldsymbol{M}$). The total derivative w.r.t. time is indicated by a dot above the variable in question, i.e. $\dot{\boldsymbol{x}} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}$. The Euclidean norm of a vector $\boldsymbol{q}$ is denoted as $\|\boldsymbol{q}\|$, $\hat{f}$ represents an estimate of the quantity f, and $\tilde{\boldsymbol{x}}$ denotes the error of the quantity $\boldsymbol{x}$.

The list of symbols is incomplete. It contains symbols which appear frequently or are of major importance. All symbols are properly introduced in the text before they are used. Past their introduction, the arguments, e.g. of $\boldsymbol{M}(\boldsymbol{\phi})$, are usually omitted for the sake of brevity and since the meaning of the symbols is usually apparent from the context. A $*$ in the list denotes that the dimension and/or the unit of the respective variable depends on the context.

Symbol	Dim.	Units	Description
$\boldsymbol{0}$	$*$	-	Zero vector or matrix of appropriate size
$\boldsymbol{a}$	$\mathbb{R}^3$	m/s^2	Vector of linear accelerations
$\boldsymbol{B}$	$\mathbb{R}^{M\times N}$	$*$	Control allocation matrix
$\boldsymbol{C}$	$\mathbb{R}^{n\times n}$	$*$	Coriolis/centrifugal matrix
c	$\mathbb{R}$	Ns^2/rad^2	Rotor thrust coefficient
$\boldsymbol{c}$	$\mathbb{R}^3$	m	Workspace center point
$\boldsymbol{D}$	$\mathbb{R}^{m\times m}$	Ns/m, Ns/rad	Damping matrix
$\boldsymbol{d}$	$\mathbb{R}^3$	m	Distance vector from EE to target
$\boldsymbol{E}$	$*$	-	Identity matrix of appropriate size
$\boldsymbol{e}$	$\mathbb{R}^3$	-	Unit vector
$\boldsymbol{F}$	$\mathbb{R}^m$	N	Vector of (generalized) forces in task forces
$\boldsymbol{f}$	$\mathbb{R}^3$	N	Force vector
$\boldsymbol{G}$	$*$	-	Jacobian of process model
g	$\mathbb{R}$	m/s^2	Acceleration of gravity
$\boldsymbol{g}$	$\mathbb{R}^n$	Nm	Gravity vector in configuration space
$\boldsymbol{H}$	$*$	-	Jacobian of measurement model
$\boldsymbol{h}$	$\mathbb{R}^{2N}$	N, Nm	Vector of rotor thrust and torques
$\boldsymbol{I}$	$\mathbb{R}^{3\times 3}$	kg m^2	Inertia tensor of a rigid body
$\boldsymbol{J}$	$*$	-	Jacobian matrix
$\boldsymbol{K}$	$\mathbb{R}^{m\times m}$	N/m, N/rad	Stiffness matrix
k	$\mathbb{R}$	Nms^2/rad^2	Rotor torque coefficient
l	$\mathbb{R}$	m	Length of lever arm
M	$\mathbb{N}$	-	Number of generalized forces
$\boldsymbol{M}$	$\mathbb{R}^{n\times n}$	kg, kg m^2	Inertia matrix
m	$\mathbb{R}$	kg	Mass
m	$\mathbb{N}$	-	Number of DoF of task (or operational) space

N	$\mathbb{N}$	-	Number of rotors
N_n	$\mathbb{N}$	-	Number of rigid bodies
N_s	$\mathbb{N}$	-	Number of subsystems
n	$\mathbb{N}$	-	Number of DoF of configuration (or joint) space
$\boldsymbol{n}$	$\mathbb{R}^3$	-	Rotor normal vector or rotation axis
$\boldsymbol{P}$	$*$	-	Covariance matrix of the estimate
p	$\mathbb{R}$	Pa	Ambient air pressure
$\boldsymbol{p}$	$\mathbb{R}^3$	m	Cartesian position
$\boldsymbol{Q}$	$\mathbb{H} \to \mathbb{R}^{4\times4}$	-	Quaternion multiplication matrix, $\boldsymbol{Q} = \begin{bmatrix}\boldsymbol{q} & \boldsymbol{Q}_q\end{bmatrix}$
$\boldsymbol{q}$	$\mathbb{H}$	-	Unit quaternion $\boldsymbol{q} = \begin{pmatrix}\eta & \boldsymbol{\epsilon}^T\end{pmatrix}^T$
R	$\mathbb{R}$	m	Propeller radius
R_f	$\mathbb{R}$	J/kg K	Ideal gas constant of dry air
$\boldsymbol{R}$	$SO(3)$	-	Rotation matrix
r	$\mathbb{R}$	m	Circumcircle radius of multicopter frame
$\boldsymbol{S}$	$\mathbb{R}^{3\times3}$	$*$	Skew-symmetric matrix operator, $\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a}\times\boldsymbol{b}$
$SE(3)$	$\mathbb{R}^{4\times4}$	-	Special Euclidean group, set of all poses represented by a homogeneous transformation matrix
$SO(3)$	$\mathbb{R}^{3\times3}$	-	Special orthogonal group, set of all orientations represented by an orthogonal rotation matrix
$\boldsymbol{s}$	$*$	-	Vector of minimal coordinates
T	$\mathbb{R}$	N	Rotor thrust magnitude
$\boldsymbol{T}$	$\mathbb{R}^3$	N	Thrust vector in body-fixed frame
T_C	$\mathbb{R}$	°C	Air temperature in degrees Celsius
T_K	$\mathbb{R}$	K	Air temperature in Kelvin
t	$\mathbb{R}$	s	Time
t_d	$\mathbb{R}$	s	Duration of trajectory segment
$\boldsymbol{u}$	$*$	-	Control input
V	$\mathbb{R}$	-	Lyapunov function
$\boldsymbol{v}$	$\mathbb{R}^3$	m/s	Vector of linear velocities
$\boldsymbol{W}$	$*$	-	Weighting matrix
$\boldsymbol{X}$	$*$	-	Covariance matrix of process noise
$\boldsymbol{x}$	$*$	-	State vector
$\boldsymbol{Z}$	$*$	-	Covariance matrix of measurement noise
$\boldsymbol{z}$	$*$	-	Measurement vector
α	$\mathbb{R}$	-	Workload sharing factor
$\boldsymbol{\alpha}$	$\mathbb{R}^6$	m/s, rad/s	Virtual control input
β	$\mathbb{R}$	rad	Tilt angle of the (floating) base
Γ	$\mathbb{R}$	$*$	Cost function
γ	$\mathbb{R}$	-	Gain of adaptation law
δ	$\mathbb{R}$	-	Relative humidity
ε	$\mathbb{R}$	-	Adaptive parameter
θ	$\mathbb{R}$	rad	Pitch angle
κ	$\mathbb{R}$	m	Rotor thrust to torque ratio, $\kappa = \frac{\tau}{T}$

$\boldsymbol{\Lambda}$	$\mathbb{R}^{m\times m}$	$*$	Pseudo kinetic energy matrix
λ	$\mathbb{C}$	-	Eigenvalue
λ	$\mathbb{R}$	-	Gain of adaptive controller
$\boldsymbol{\mu}$	$\mathbb{R}^{m}$	$*$	Vector of Coriolis/centrifugal terms
ν	$\mathbb{R}$	m/s^2	Signal quantity
$\boldsymbol{\xi}$	$\mathbb{R}^3$	m	Cartesian position
π	$\mathbb{R}$	-	Mathematical constant
ρ	$\mathbb{R}$	kg/m^3	Air density
$\boldsymbol{\rho}$	$\mathbb{R}^3$	m	Vector from CoG of UAV to EE of manipulator
σ	$\mathbb{R}$	$*$	Standard deviation
$\boldsymbol{\tau}$	$*$	Nm	Torque vector
$\boldsymbol{\Phi}$	$SO(3)$	-	Attitude representation
$\boldsymbol{\phi}$	$\mathbb{R}^n$	rad	Vector of joint angles
φ	$\mathbb{R}$	rad	Roll angle
$\boldsymbol{\chi}$	$\mathbb{R}^n \rightarrow \mathbb{R}^m$	m, rad	Forward (or direct) kinematics
Ψ	-	-	Coordinate frame
ψ	$\mathbb{R}$	rad	Yaw (or heading) angle
Ω	$\mathbb{R}$	rad/s	Rotational speed of single rotor
$\boldsymbol{\Omega}$	$\mathbb{R}^N$	rad/s	Vector of rotor speeds
$\boldsymbol{\omega}$	$\mathbb{R}^3$	rad/s	Angular velocity vector of rigid body
$\boldsymbol{\varpi}$	$\mathbb{R}^N$	rad^2/s^2	Vector of squared rotor speeds

Indices

0	Initial value
b	(Floating) base
d	Desired
e	Error
f	Final value
g	Gravity
i	Inertial frame
i	Rotor number
j	Joint
k	Current time step
k	Number of rigid body
m	Model
o	Origin
p	Predecessor
r	Robot manipulator
u	Flying robot (or UAV)

Abbreviations

att	Attitude
ax	Axis
cf.	*Confer* (compare)
comp	Compensation
ctrl	Control
e.g.	*Exempli gratia* (for example)
ext	External
i.e.	*Id est* (that is)
max	Maximum
min	Minimum
nsp	Nullspace
sat	Saturation
sgn	Signum (sign)
vs.	Versus
w.r.t.	with respect to

"The fact that the great scientist believed in flying machines was the one thing that encouraged us to begin our studies."
— Wilbur Wright

1 Introduction

1.1 Motivation

This thesis considers a recently fast-growing class of robotic systems: flying robots, or so-called unmanned aerial vehicles (UAVs). Due to their ability to fly, they are theoretically able to operate in an infinitely large three-dimensional space, which is the greatest advantage compared to wheeled mobile robots or stationary robot manipulators.

Unmanned aerial vehicles are commonly classified in terms of take-off and landing capabilities [1]. As depicted in Figure 1.1, one can distinguish between fixed-wing, vertical take-off and landing (VTOL), and hybrid UAVs. VTOL UAVs are the primary focus of this work. They can be further subdivided in helicopters as shown in Figure 1.1b and 1.2a, which have variable-pitch rotor blades, and multicopters as depicted in Figure 1.2b and 1.2c with fixed-pitch propellers. The cumulative thrust and therefore the possible payload, but also the weight of the sum of the components rise with the number of rotors of a multicopter. The most common multicopters are quadrocopters (four rotors), followed by hexacopters (six rotors), and octacopters (eight rotors). Due to their small size, multicopters are also commonly referred to as micro aerial vehicles (MAVs). VTOL UAVs are nowadays widely used in commercial applications, e.g. for agricultural spraying (Figure 1.2a), aerial photography and filming (Figure 1.2b), surveillance and inspection tasks (Figure 1.2c), or search and rescue (SAR) missions. Another application of flying robots, which is still in its infancy and has not yet fully arrived in industry, is aerial manipulation [5].

As mentioned before, VTOL UAVs are able to take-off and land vertically. However, take-off and landing within narrow spaces, for example on a ground vehicle equipped with a small

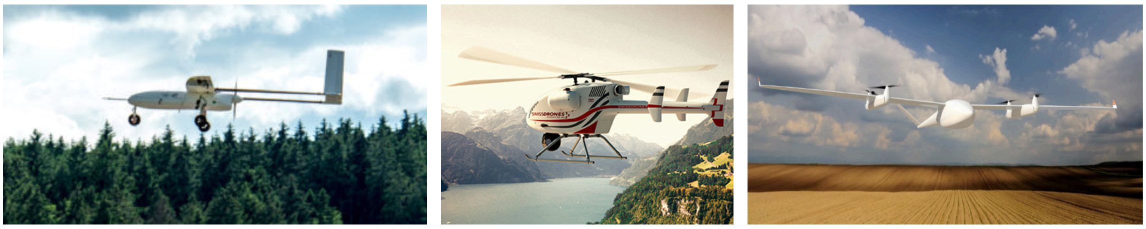

(a) Fixed-wing UAV [2]. **(b)** VTOL UAV [3]. **(c)** Hybrid UAV [4].

Figure 1.1: Three different classes of unmanned aerial vehicles (UAVs).

(a) Yamaha RMAX helicopter used for precision agriculture [6].

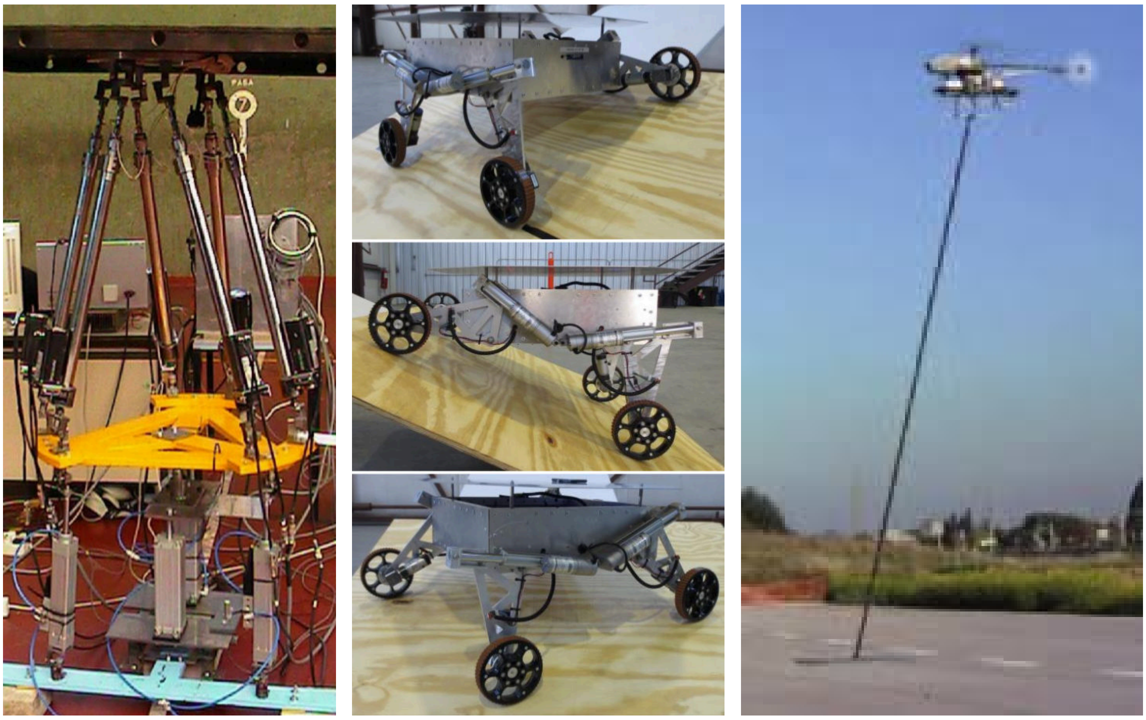
(b) DJI Inspire quadrocopter suitable for professional photography or filming [7].

(c) Intel Falcon 8 octacopter performing inspection of industrial sites, e.g. power plants or oil rigs [8].

Figure 1.2: Exemplary VTOL UAVs for different applications.

landing platform, require precise pose estimation. Moreover, the risk of a collision with an obstacle is high, especially if the surface or the ground vehicle is moving or if wind gusts occur. It is also possible that a UAV has no autonomous landing functionality, e.g. because it lacks processing power or dedicated sensors for landing spot detection. Another cause that prevents autonomous landing is saturation of the flying robot's actuators, i.e. if heavy side winds are present or during fast maneuvers. Hence, the aim of this thesis is to facilitate take-off and landing of flying robots under severe conditions. The basic idea is to use a robot manipulator to compensate the movements of the landing surface and assist the UAV during take-off and landing (see Section 1.2). In order to fulfill the task, coordinated control of the flying robot and the robot manipulator is required and the development and evaluation of suitable control approaches is the major contribution of this thesis.

Up to date, different assistance systems have been proposed in the literature. Campos et al. [9] and Godzdanker et al. [10] developed self-leveling landing platforms as shown in Figure 1.3a. These stationary platforms are able to compensate e.g. ship motion, but are limited in their operating range and do not provide any clamping mechanism to fix the VTOL UAV and prevent it from sliding. In order to increase the operating range of self-leveling platforms, Conyers et al. [11] have developed a semi-autonomous mobile landing platform for VTOL UAVs. It is a four-wheeled quadruped robot as shown in Figure 1.3b. However, the maximum inclination angle remains limited. Enlarging the platform could increase the possible inclination angle, but would lower its portability because of the larger size and weight. Another approach is tether-guided landing (see Figure 1.3c), as presented by Oh et al. [13] as well as by Sandino et al. [12]. A tether offers a much larger operating range, fixes the UAV to the landing surface, and makes GPS position measurements superfluous during the landing phase. Though, a device for measuring rope angle and rope force is needed. Moreover, the tether can only transfer forces in one direction and is therefore not able to increase the lifting force of the VTOL UAV. In order to control the tension of the tether, it needs to be pulled manually or by an electric winch. The winch can be either mounted on the UAV, which is crucial due to payload limitations, or on the ground. In the latter case, it is unclear how the rope can be conveniently attached to and detached from the flying vehicle. In order to overcome the limitations of the aforementioned approaches, robot-assisted take-off and landing of flying robots is proposed in this thesis.



(a) Self-leveling landing platform presented by Campos et al. [9]. The lower part is a parallel robot with three DoF and pneumatic prismatic actuators. It simulates ship motion and is used for testing the so-called active helideck (upper part, six DoF and prismatic electrical actuators).

(b) Mobile self-leveling landing platform developed by Conyers et al. [11]. The semi-autonomous mobile robot has four wheels and differential drive. Each wheel is mounted on a lever arm that can be rotated up and down using an electric linear actuator. Thus, the platform can be inclined up to 25 degrees.

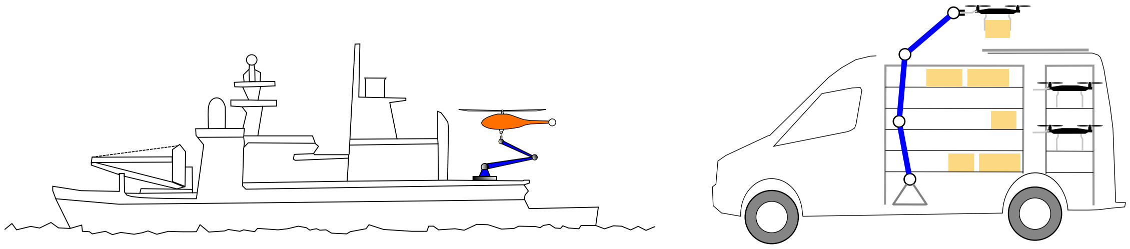
(c) Tether-guided landing demonstrated by Sandino et al. [12]. One end of the rope is attached to the unmanned helicopter via a universal hinge mechanism, which measures rope angle and force. The other end is connected to a pulley mounted on the ground. By pulling or releasing the tether, the VTOL UAV can land or take-off, respectively.

Figure 1.3: Three existing approaches for supporting VTOL UAVs during take-off and landing.

1.2 Concept of a robot manipulator as support system

The fundamental idea in this thesis is to use a robot manipulator to assist a flying robot during take-off and landing. This novel robotic assistance (or support) system consists of a serial manipulator mounted on the ground (or on a moving or floating vehicle) and a special docking mechanism. It allows to quickly capture the flying vehicle for landing or to steadily release it for take-off. Two exemplary scenarios where robot manipulator assistance for flying robots can be used are depicted in Figure 1.4.

The main purpose of the manipulator is to counteract movements of the landing surface, also referred to as base motion. So, instead of catching the flying robot, it is also possible that the manipulator only compensates the base motion and provides a steady landing area for the flying robot. The base motion may be characterized by its amplitude and frequency. The geometry, link lengths, and end stops of the manipulator define its workspace and therefore the amplitude it is able to compensate. On the other hand, the manipulator's



(a) Take-off and landing of an unmanned helicopter aboard a ship, e.g. for polar ice monitoring. Copyright © 2017 IEEE [212].

(b) Deployment of delivery MAVs from a specially designed truck.

Figure 1.4: Two examples where a robot manipulator can assist a VTOL UAV in order to take-off and land safely. The vessel (left) or the truck (right) can continue their own tasks while the UAV takes off or lands. This enables to fulfill the complete mission more efficiently.

actuator dynamics define the maximum attainable frequency. Thus, the requirements for the manipulator vary depending on the specific application.

A manipulator allows take-off and landing under heavy side wind conditions. Due to underactuation, a flying robot needs to tilt opposite to the wind direction in order to counteract the wind forces. A robot manipulator with at least six degrees of freedom is able to tilt the docking interface in any desired direction. Hence, it facilitates take-off and landing under side wind conditions, which are very challenging tasks.

The robot manipulator also enables avoiding the ground effect (see Section 3.2.3 for an explanation). The latter can potentially lead to instable flight, because of changing rotor inflow. A manipulator used for take-off and landing directly decreases the energy consumption of the flying robot. This, in turn, increases flight and operating time. Furthermore, the manipulator can be used to automate maintenance tasks before or after the flight mission, such as refueling, recharging the batteries, or replacing the payload.

The advantages of robot manipulator assistance for flying robots compared to existing approaches can be summarized as follows:

- Within the boundaries of its dexterous workspace ([14], p. 102), the robot manipulator can compensate motions of the surface (or of the moving or floating vehicle) and counteract side winds, both in all six degrees of freedom.

- The manipulator can support the flying robot (in all six degrees of freedom), such that its actuator limits are not exceeded and less power is required during take-off and landing.

- Due to its length, the manipulator enables to avoid the ground effect, and, therefore, eliminates this uncertainty of the take-off and landing tasks.

- As soon as the flying robot is connected to the manipulator, the pose estimation of the UAV reduces to an evaluation of the kinematic chain, because the geometry of the robot manipulator is known and its joint angles are measured.

- In addition to take-off and landing, the robot manipulator can be used on the ground for moving the VTOL UAV in and out of a storage rack (or transport box) and for performing maintenance tasks.

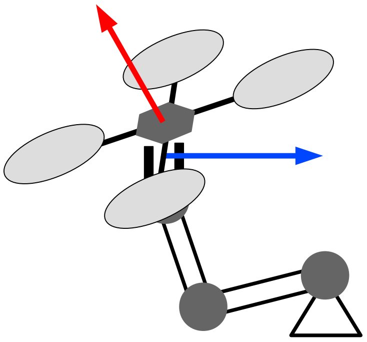
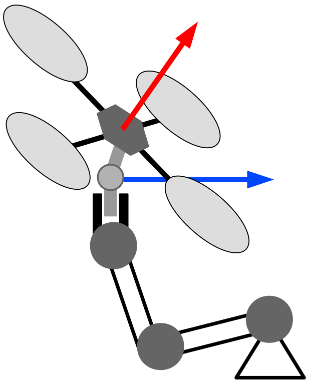
- For helicopters, the possible take-off weight can be increased if a forward velocity is induced (and maintained during flight). This is due to fact, that the large rotor blades of helicopters produce lift forces in forward flight.

1.3 Challenges and research questions

There are many open challenges for robotic assistance for flying robots. Instead of a single vehicle, now two robotic systems are involved and have to be coordinated. The need for coordinated control of the flying robot and the robot manipulator is illustrated in Table 1.1 based on an example.

It is clear that the desired coordination described on the right of Table 1.1 may be realized using various control approaches with different properties and varying complexity. Hence,

Table 1.1: Motivation of coordinated control.

	
Without coordinated control • The force vectors of flying robot (red) and of manipulator (blue) potentially point in opposite directions. • The manipulator has to counteract the force of the flying robot. • Due to the lever arm, the tilt angle of the flying robot increases, which has a destabilizing effect.	**With coordinated control** • The horizontal components of the force vectors point in the same direction. • The required forces of both systems are reduced, i.e. both contribute to the task. • More energy is available to counteract wind or to handle heavier flying robots.

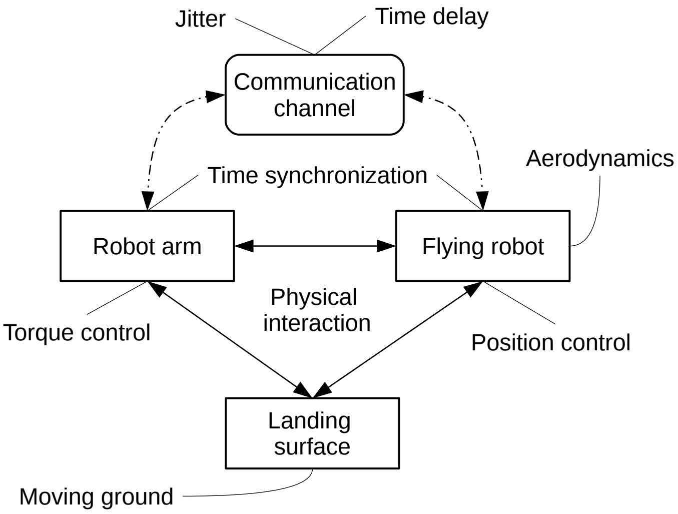


Figure 1.5: Involved components and challenges of a robot manipulator assistance system for flying robots.

different strategies for coordinated control of robot manipulators and unmanned aerial vehicles need to be investigated and evaluated.

Figure 1.5 summarizes other major challenges. Robot manipulators and flying robots have different dynamics and different actuation principles, which need to be considered for the coordinated control design. This includes resolution of task redundancy for robot manipulators as well as dedicated control of flying robots, whose rotor forces include uncertainties because of unmodeled aerodynamics and whose actuated degrees of freedom are less than six (commonly referred to as underactuation).

Since both robot manipulator and flying robot usually operate independently, they are equipped with their own control algorithms. They might have accessible interfaces, allow modifications, or neither of both. If coordinated task control is implemented on the manipulator side as shown in Figure 1.6, the flying robot needs at least to be capable of receiving commands. Depending on the interface, these commands can be on force, velocity, or pose level. In general, the manipulator is assumed to be torque-controlled and the UAV is assumed to be attitude- and position-controlled.

External disturbances are present during take-off and landing, e.g. due to wind, base motion, or physical interaction between flying robot and manipulator. To estimate and counteract those disturbances, suitable observers are required. If base motion is present, it needs to be estimated as well. The disturbances may also lead to actuator saturation. Fortunately, it is possible to distribute the workload considering all actuators of the flying robot and the manipulator, e.g. using an optimization-based control allocation.

To effectively coordinate and fulfill the task, state information needs to be exchanged and commands need to be sent and received. This requires a dedicated high bandwidth and low latency communication channel. It is reasonable to assume that the computing capacity of a ground control station is much higher compared to the on-board control computer of a flying robot. Thus, a distributed but centralized[1] control architecture is assumed. That means

[1] The most relevant terms describing the control system architecture are defined in Appendix A1.

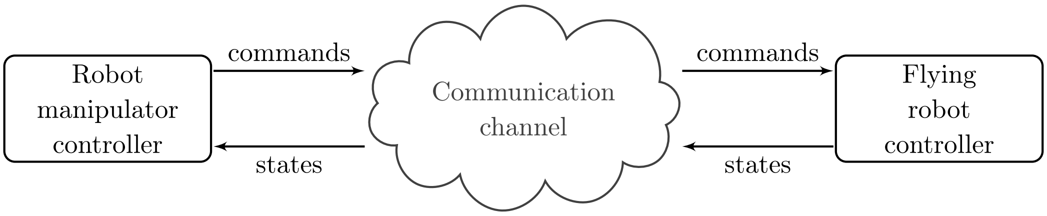


Figure 1.6: Communication channel with information directions.

that task control is implemented on the ground station which also controls the manipulator. This results in the communication channel depicted in Figure 1.6. The transmission of data inevitably involves a time delay. In order to process received data in a meaningful way and to account for time delay, all messages require timestamps and the clocks of ground station and flight control computer have to be synchronized.

Due to sensor noise and time delay, the required pose estimates of base, robot manipulator, and flying robot are subject to uncertainties. These need to be accounted for, e.g. by fusing multiple available state estimates in a suitable filter or by using a docking interface which is robust by design.

In conclusion, the problem of coordinated control of robot-assisted take-off and landing of flying robots raises the following research questions:

Q1 How can the dynamics of the robotic support system, composed of a flying robot and a manipulator on a moving base, be modeled?

Q2 How can the flying robot and the robot manipulator on a moving base be controlled in a coordinated fashion?

Q3 How can both flying robot and robot manipulator contribute to a successful realization of the assistance task?

Q4 How can the relative distance between manipulator and flying robot be measured accurately in order to realize robust take-off and landing?

Q5 How do the coordinated control approaches perform in simulations and real world experiments with different VTOL UAVs?

These major research questions are addressed in this thesis. The main contributions are summarized next.

1.4 Contributions and overview

The control and interaction of a flying robot with a ground based robot manipulator have not been studied extensively up to now. In this thesis, robot manipulator assistance for flying robots is considered. The focus lies on model-based controller design for coordinated control of the two subsystems. Specifically, independent control, uni- and bilaterally coupled control, as well as combined control (with workload sharing or optimal control allocation) are considered.

State-of-the-art visual inertial state estimation methods are used, enabling autonomous robot-assisted take-off and landing of VTOL UAVs, without the need for external localization (or motion capture) systems. Special emphasis is also put on robust control design and allocation for flying robots. Besides the control strategy, a universal hinge (or ball joint) between robot manipulator and flying robot is proposed. It allows the flying robot to control its orientation independently of the manipulator's end-effector. Hence, the flying robot is able to contribute to the take-off and landing task, which increases reliability and reduces the overall energy consumption. Suitable mechanical docking mechanisms are discussed as well. A computer vision approach is proposed to measure the relative distance between flying robot and manipulator. A suitable multi-marker tracking algorithm is presented and a Kalman filter is designed. The approach combines different sensor measurements, considers transmission time delays, and provides robustness against occlusions.

In summary, the main contributions of this work are the following:

- Derivation of a combined model of robot manipulator and flying robot and different model decompositions.
- Independent dynamics modeling of flying robots, introduction of bidirectional thrust, and generalized control allocation for multicopters under actuator saturation.
- Development of a novel adaptive control approach to overcome the uncertainty of the rotor thrust. It is applied to flying robots as well as to robot manipulators.
- Controller designs considering separate models of flying robot and manipulator, including state space control based on a linearized flying robot model, backstepping control based on nonlinear flying robot model, and robot manipulator control with base motion compensation.
- Controller designs considering the combined model of flying robot and manipulator, including task space control and heuristic or optimization-based distribution of the workload between UAV and manipulator.
- Implementation of a visual tracking algorithm for multiple fiducial markers and a Kalman filter to robustly estimate the distance between manipulator and flying robot even in the presence of occlusions and time delay.
- Presentation of a visual servo controller for the manipulator, including virtual workspace boundaries, for in-flight capturing of a flying robot.
- Development of a vision-based quadrocopter (with bidirectional thrust capabilities) and of a robot-assisted take-off and landing demonstrator based on a DLR Light Weight Robot (LWR).
- Evaluation of all control approaches in simulation, with and without base motion, and experiments with different multicopters and the LWR.

An overview of the control-related topics contained in this work is given in Table 1.2. The remainder of this work is structured as follows: In Chapter 2, fundamentals of rigid body dynamics modeling are presented. The considered robotic systems and the different phases of

Table 1.2: Thesis structure overview in terms of covered control-related topics.

<table>
<tr><td colspan="2"><b>1 Introduction</b></td></tr>
<tr><td colspan="2"><b>2 Preliminaries</b>
Rigid body dynamics of combined system
Generalized task space control</td></tr>
<tr><td colspan="2"><b>3 Modeling and control of flying robots</b>
Rigid body, propulsion, ground effect, and atmosphere models
Generalized position and attitude control
Multicopter control allocation under saturation
Adaptive control for increased robustness</td></tr>
<tr><td><b>4 Controller designs using separate models</b>
Linear state space control
Nonlinear backstepping control
Active thrust vector control
Task space control with base motion compensation</td><td><b>5 Controller designs using combined model</b>
Task space control based on combined and decomposed model
Heuristic workload sharing
Optimal control allocation
Adaptive combined control</td></tr>
<tr><td colspan="2"><b>6 In-flight capturing of a flying robot</b>
Tracking algorithm for multiple fiducial markers
Kalman filter design for sensor fusion under time delay
Visual servo control of robot manipulator</td></tr>
<tr><td colspan="2"><b>7 Conclusion</b></td></tr>
</table>

take-off and landing are introduced, the dynamic model of the combined system is presented, and generalized task space control is discussed briefly.

Chapter 3 is devoted to independent modeling and robust control of flying robots. A generalized control allocation procedure with saturation handling and an adaptive control approach are presented.

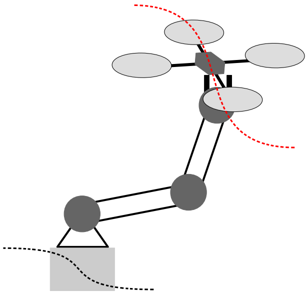
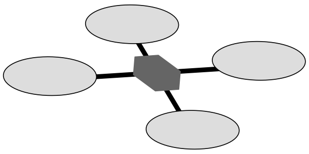
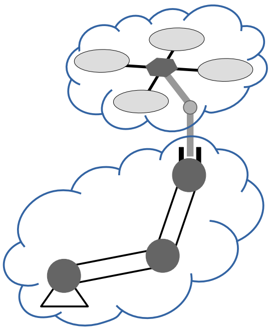
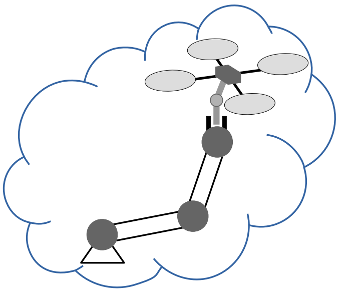
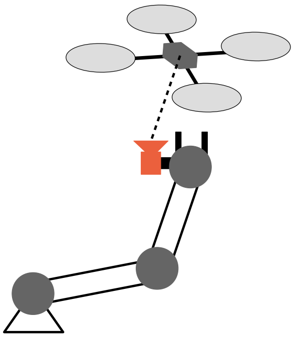
The separated dynamics of a flying robot attached to a manipulator via a ball joint are considered in Chapter 4. Based on a linearized model of the flying robot, different linear controllers and a stability citerion for the attitude controller of the UAV are derived. Then, the nonlinear dynamics of the flying robot are used to design a backstepping position and attitude controller. Both the linear and the nonlinear controllers are included in a generalized task space controller for the robot manipulator in order to realized coordinated take-off and landing of the flying robot by means of the manipulator. The task space controller is then extended to account for estimated base motion. It is evaluated in a simulation study using realistic ship motion.

The combined dynamics of the flying robot connected to the robot manipulator via a ball joint are considered in Chapter 5. Task space controllers for the combined model and for the decomposed model are presented. The latter allows for heuristic workload sharing between flying robot and manipulator. Based on task space control of the combined system, the control allocation problem for flying robot and manipulator is formulated as a quadratic optimization problem. The unconstrained problem is solved analytically and the constrained problem is solved numerically. The adaptive control approach presented in Chapter 3 is extended to the combined system to account for the uncertainty of the thrust of flying robots.

In-flight capturing of a flying robot using a robot manipulator is addressed in Chapter 6. A visual tracking approach for multiple fiducial markers attached to the UAV and a Kalman filter for estimating the relative distance between flying robot and manipulator based on distributed sensor measurements are presented. The estimated distance is used to realize Cartesian impedance visual servo control of the robot manipulator. Different mechanical docking interfaces are also discussed briefly. Finally, the thesis is summarized and all presented control approaches are compared in Chapter 7.

This thesis is based on the author's published international and peer-reviewed papers [213, 212, 214, 215] and articles [208, 209]. Their content is included in several chapters as shown in Table 1.3. The table also provides brief explanations and illustrations of the topics covered in the different chapters. A comprehensive list of the author's publications can be found at the end of this thesis.

Table 1.3: Summary of the topics covered in this thesis.

Topics		Chapters	Own references
	Introduction and modeling of a robot manipulator as support system for flying robots	1, 2	*CCTA2017* [212]
	Independent modeling and robust control of a flying robot	3	*IROS2018* [215] *RA-L2020* [209]
	Controller designs for take-off and landing considering separate models	4	*CDC2015* [213] *RA-L2016* [208] *CCTA2017* [212]
	Controller designs for take-off and landing considering combined model	5	*IROS2017* [214]
	Visual tracking and capturing of a flying robot by means of the manipulator	6	

"The enchanting charms of this sublime science reveal only to those who have the courage to go deeply into it."
— Johann Carl Friedrich Gauss

2 Preliminaries

This chapter lays the groundwork for all subsequent chapters of this thesis. The two considered robotic systems are introduced by means of examples developed at the DLR Institute of Robotics and Mechatronics: two vision-based multicopters (Section 2.1) and the light weight robot manipulator (Section 2.2). In order to deal with the complexity of robot manipulator assistance for flying robots, take-off and landing tasks are separated into multiple phases in Section 2.3. In addition, the two assistance strategies, full and partial clamping, are introduced and discussed.

The focus of this thesis is the model-based and coordinated control of a robot manipulator and a flying robot. Hence, suitable dynamics models are required, which are presented in detail in Section 2.4. This includes a summary of the considered attitude representations in Section 2.4.1. In Section 2.4.2, a synthetic and projection-based dynamics modeling approach is introduced. It allows to combine the rigid body models of flying robot, robot manipulator, and floating base, for instance during the phases where the flying robot is attached to the manipulator. The complete rigid body dynamics, referred to as combined system model, are derived and different realizations, e.g. the floating or fixed base case and the planar case, are treated in Section 2.4.3. Then, the well-known task space formulation is presented in Section 2.4.4. In Section 2.5, state-of-the-art control approaches for redundant and distributed robotic systems are summarized. Finally, different established task space controllers are introduced briefly in Section 2.6.

2.1 Overview of vision-based flying robots

As pointed out in the introduction, different types of flying robots exist, while in this work vertical take-off and landing unmanned aerial vehicles (VTOL UAVs) are considered. One example is the hexacopter Ardea [217, 210] developed at the DLR Institute of Robotics and Mechatronics and depicted in Figure 2.1. In general, the main components of a VTOL UAV are its frame or fuselage, the electrical components, e.g. sensors and on-board computing hardware, and the mechanical propulsion components, such as servos, motors, and propellers. The combination of motors and propellers is commonly referred to as the rotor system which generates the lift required for the vehicle to fly. Details regarding the dynamics and the propulsion system can be found in Section 3.2. The energy required by the propulsion system, computing hardware, and sensors is supplied by the on-board battery.

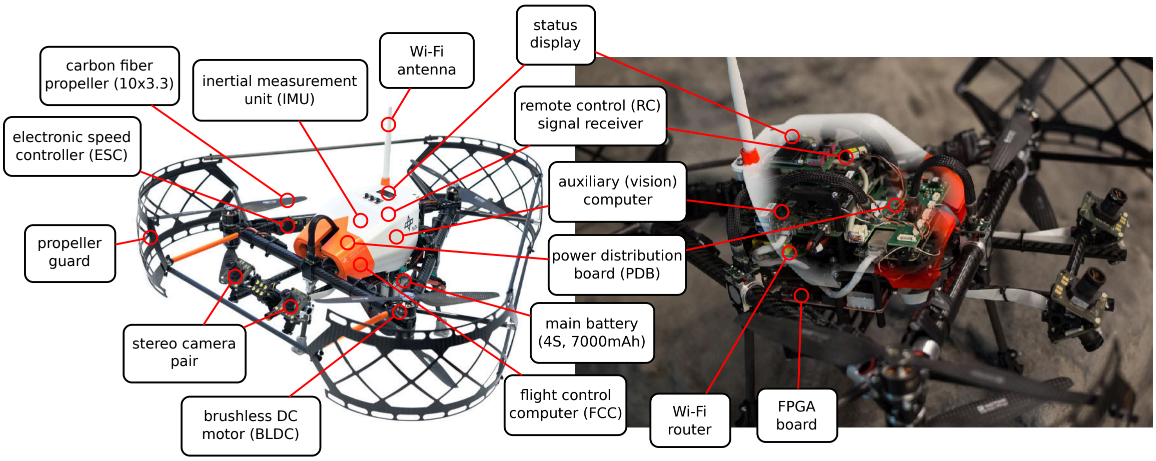
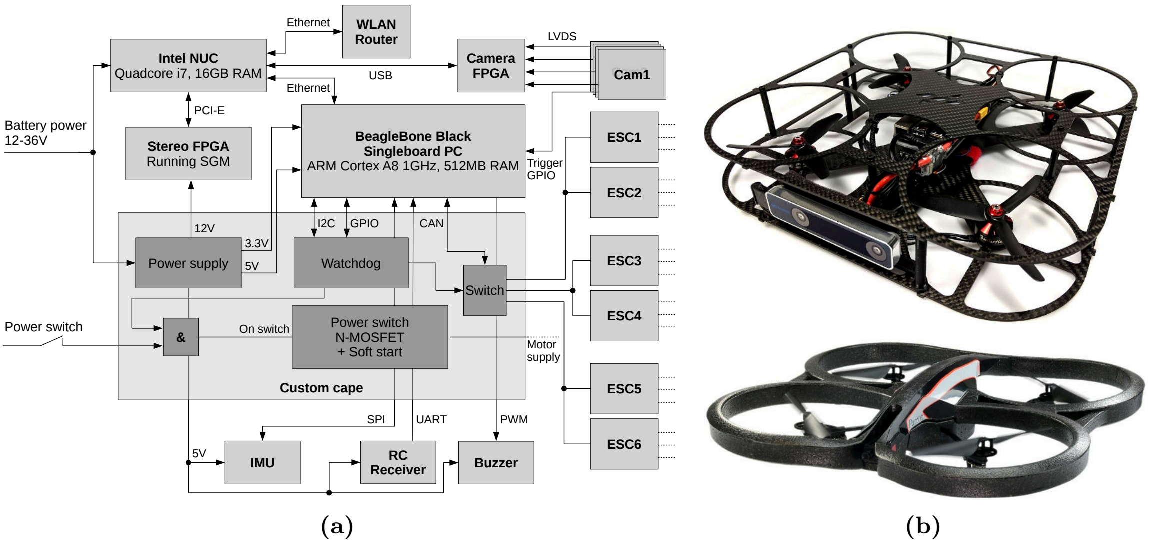

carbon fiber propeller (10x3.3)
inertial measurement unit (IMU)
Wi-Fi antenna
status display
electronic speed controller (ESC)
remote control (RC) signal receiver
auxiliary (vision) computer
propeller guard
power distribution board (PDB)
main battery (4S, 7000mAh)
stereo camera pair
flight control computer (FCC)
brushless DC motor (BLDC)
Wi-Fi router
FPGA board

Figure 2.1: Components of the vision-based autonomous hexacopter Ardea developed at the DLR Institute of Robotics and Mechatronics.

In terms of control, sensors are required to measure the actual states of the vehicle. The inertial measurement unit (IMU) provides linear accelerations and angular velocities, which are used to compute an estimate of the orientation. For outdoor operation, GPS is usually used to acquire a position measurement. However, Ardea is intented for indoor operation where GPS is not available. Hence, Ardea solely uses a wide-angle multi-camera system for pose estimation [217]. The stereo camera images are processed in an FPGA implementation of the Semi-Global Matching (SGM) algorithm [15]. The visual odometry (VO) [217] is implemented using the Robot Operating System (ROS) on an Intel NUC with Quadcore i7 processor (3 GHz). It computes an estimate of the camera motion based on the perceived images (8 Hz) which are fused with IMU measurements (200 Hz) using a Kalman filter [16, 17] to generate a pose estimate at 50 Hz. The vision-based approach enables the flying robot to operate outdoors as well as indoors.

The on-board flight control computer is a BeagleBone Black (1 GHz) running Ubuntu Linux with a custom real-time patch. Communication between Ardea and ground station is established via 5 GHz WLAN. To enable manual control, a commercial Spektrum 2.4 GHz remote control system is used. What distinguishes Ardea from common flying robots is that closed-loop motor speed control is implemented directly on the electronic speed controllers (ESCs). The ESCs receive rotor speed commands and provide motor telemetry via a CAN bus. The attitude control loop runs at 200 Hz and the position control loop at 50 Hz. For mission control a RAFCON [18] state machine is used. The hardware setup of Ardea is depicted in Figure 2.2a. System identification of Ardea and the deployed control methods are treated in detail in [19].

Another flying robot developed entirely from scratch within the scope of this thesis is the quadrocopter Sparrow [215, 209] shown in Figure 2.2b (top). It uses the same flight control code base and custom middleware as Ardea. It is equipped with a Raspberry Pi 3B+ running Raspbian Linux with real-time patch, a commercial Navio2 autopilot hat, ESCs with custom rotor telemetry, and the off-the-shelf Intel RealSense Tracking Camera T265. The latter provides a pose estimate at 200 Hz. In addition to Ardea and Sparrow, the quadrocopter AR.Drone 2.0 (Figure 2.2b, bottom) is used for the experiments presented in this thesis. It



(a) (b)

Figure 2.2: Overview of the hardware used on Ardea (left) and quadrocopters Sparrow (top right) and AR.Drone 2.0 (bottom right).

is a consumer product and uses optical flow from a single downward facing camera together with its IMU for linear velocity estimation and control. For programming the AR.Drone, a Software Development Kit (SDK) is provided by the manufacturer Parrot. Details and novel concepts regarding position and attitude control of flying robots are presented in Chapter 3.

2.2 Overview of the DLR Light Weight Robot

Robot-assisted take-off and landing of a flying robot is realized by means of a robot manipulator as shown in Figure 2.3 (a). The latter is composed of multiple rigid links connected to each other by (revolute or prismatic) joints that allow motions of the links. Encoders in the joints provide the relative positions of neighbouring links. For revolute (or rotary) joints these are the so-called joint angles. Some other common terms require introduction, since they will be used throughout this work: Tools mounted at the last link of the robot manipulator are referred to as end-effectors. In terms of control, one is usually interested in the position or motion of the tool center point (TCP). Some state-of-the-art robot manipulators, for example the DLR Light Weight Robot (LWR) [20, 21], are equipped with torque sensors in every joint. The joint torque measurements can be used to estimate external forces (supplementary to a force/torque sensor at the end-effector) and for torque control [22], which in turn enables other control techniques such as impedance control [23]. The torque-controlled LWR 4+ shown in Figure 2.3 (a) is used in this thesis. It has seven degrees-of-freedom (DoF) and its geometrical parameters as well as the considered coordinate frame convention are illustrated in Figure 2.3 (b) and Appendix A2. Each joint includes a flexible harmonic drive gear and the joint position is controlled at 3 kHz. Position, force, and impedance control of the end-effector are realized at 1 kHz [21].

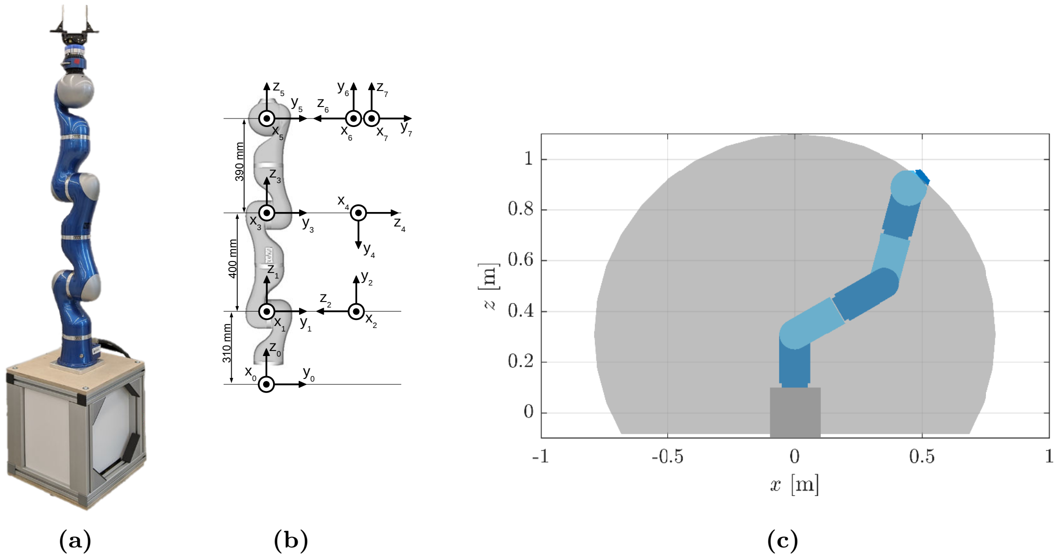


Figure 2.3: Manipulator LWR 4+ used in this thesis (a) with link lengths and coordinate frames (b) and workspace visualization (c).

Table 2.1: LWR 4+ specifications.

Robot manipulator name	Light Weight Robot (LWR) 4+						
Weight (excl. base and control computer)	16 kg						
Payload	7 kg						
Repeatability (ISO 9283)	±0.05 mm						
Work envelope	$1.7\,\mathrm{m}^3$						
Axis	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6	Axis 7
Range	±170.0°	±120.0°	±170.0°	±120.0°	±170.0°	±130.0°	±170.0°
Max. speed	±112.5°/s	±112.5°/s	±112.5°/s	±112.5°/s	±180.0°/s	±112.5°/s	±112.5°/s
Max. torque	165.0 Nm	165.0 Nm	70.0 Nm	70.0 Nm	70.0 Nm	30.0 Nm	30.0 Nm

For the purpose of this thesis, the LWR serves as an exemplary robot manipulator. It is designed for safe human-robot interaction [218], due to the compliant control of the end-effector. However, because of the latter, it is also perfectly suitable for physical interaction tasks, such as robotic take-off and landing assistance for flying robots. The dexterous workspace ([24], p. 85), i.e. the space of admissible end-effector positions and orientations, of the LWR is limited (cf. Figure 2.3 (c)). It is studied in detail using a manipulability (or capability) measure and reachability maps in [25]. Regarding take-off and landing assistance for flying robots, the workspace of the LWR will not allow to compensate large ship motion. However, the control approaches presented in this work are directly applicable to robot manipulators with sufficiently large workspace. For the capturing task, i.e. catching a flying robot midair, the maximum end-effector velocity is also of interest. It depends on the maximum joint velocities (see Table 2.1) and the current configuration of the manipulator. A further analysis of the attainable end-effector velocities in the workspace is provided in Chapter 6. For cooperative landing of a heavy flying robot, the maximum joint torques, listed in Table 2.1, are of major concern, since they define the maximum payload. They are considered for optimal distribution of the control effort between robot manipulator and flying robot in Chapter 5.

2.3 The different phases of take-off and landing

Robot-assisted take-off and landing of flying robots (or UAVs) can be subdivided into multiple tasks or phases, during which different influences become dominant and different assumptions hold. For each phase, a suitable control approach is required. First, one can obviously distinguish between start (or take-off) and landing (or touch-down). Second, the start and landing maneuvers can each be divided in three phases shown in Figure 2.4, Figure 2.5, and Table 2.2.

Table 2.2: Short description of the phases of robot-assisted take-off and landing.

Robot-assisted take-off	**Robot-assisted landing**
T1) Coordinated take-off	L1) Flying robot approach
T2) Release to flight	L2) Capture in flight
T3) Flying robot departure	L3) Coordinated touch-down

During both phase T1 and L3, the flying robot is attached to the end-effector of the robot manipulator. If, as discussed below, the actuators of the flying robot are active in T1 and L3, coordinated control of flying robot and manipulator may be required, depending on the payload capabilities of the manipulator and the weight and kinetic energy of the UAV. In phase T2 and L2 the transition from rigid connection to free flight and vice versa takes place. Hence, in T2 a switch from coordinated to independent flying robot control and in

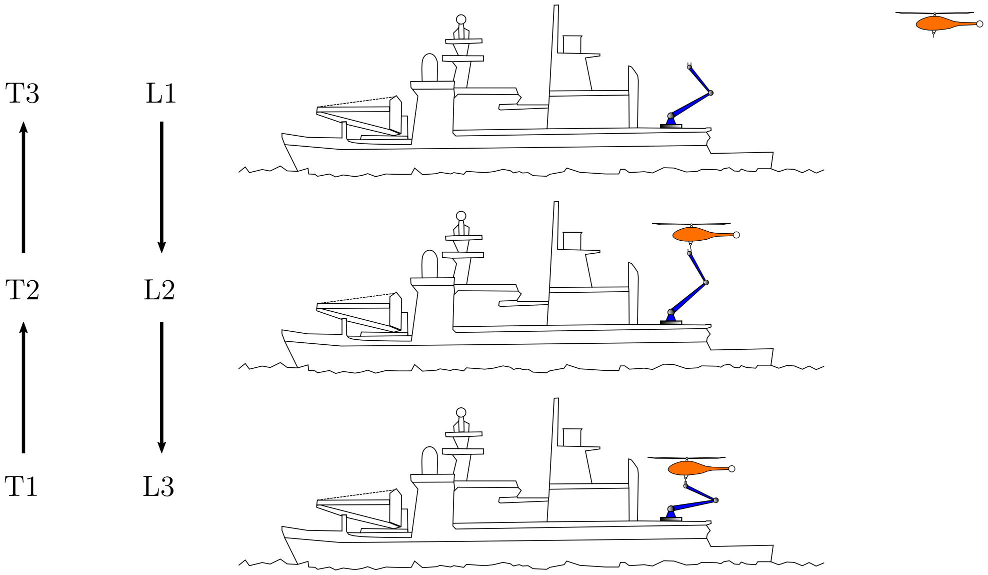


Figure 2.4: Graphical visualization of the different phases of robot-assisted take-off and landing. Copyright © 2017 IEEE [212].

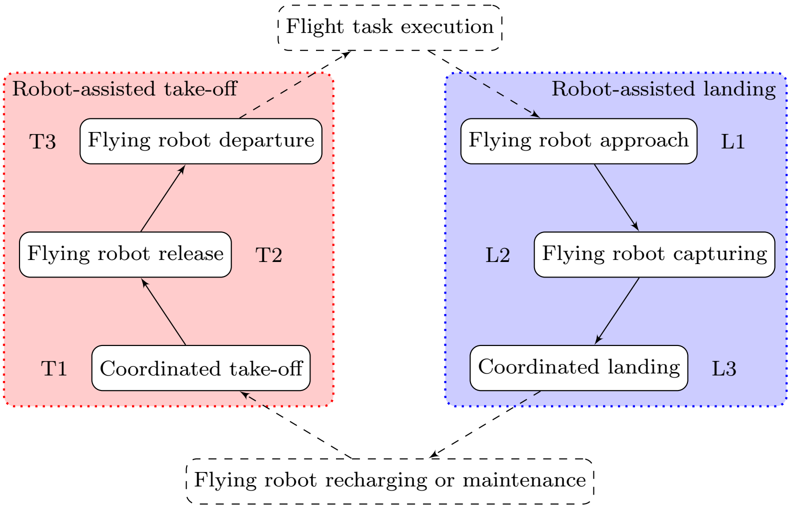


Figure 2.5: Transitions between the different phases of robot-assisted take-off and landing.

L2 a switch from independent flying robot to coordinated control is performed. In both phase T3 and L1, the flying robot and the robot manipulator can be treated completely independently.

Phase L2 is especially challenging, since it requires precise tracking of the relative pose between flying robot and manipulator as well as a robust docking mechanism. The design of the latter is crucial, since it should also compensate for inaccuracies in the pose estimation. Phases T1 and L3 are interesting in terms of control, because of the physical (and possibly rigid) connection between flying robot and manipulator. It depends on the payload capacity of the manipulator and on the actual docking mechanism, whether the flying robot needs or is able to use its actuators, i.e. the rotors, or if they are turned off while it is attached to the robot manipulator. Note that also the inertial forces matter for movements with acceleration. If the flying robot is actively controlled while connected to the robot, take-off and landing become a cooperative task for robot manipulator and flying robot. The latter requires an understanding of the actuation principles and the control systems of both flying robots and robot manipulators, as well as the development and study of a dedicated control architecture for the complete system consisting of flying robot, robot manipulator, and moving base. Hence, in this work, the following two cases of robotic take-off and landing assistance are distinguished:

- *Fully clamped*:
  The flying robot is rigidly connected to the robot manipulator, such that all its DoF relative to the TCP are locked.

- *Translationally clamped / rotationally free*:
  The flying robot is connected to the ground robot via a universal hinge (or ball joint), which leaves the rotational DoF relative to the TCP open.

In both cases, the connection can be opened and closed, e.g. by using a gripper or an electro magnet, so that the flying robot can be released for take-off and recaptured for landing.

Table 2.3: Advantages ($\oplus$) and disadvantages ($\ominus$) of full and partial clamping.

Full clamping	**Partial clamping**
$\oplus$ safe, once UAV is fixed and inactive $\oplus$ no special UAV control needed	$\oplus$ retains UAV attitude dynamics $\oplus$ allows higher UAV weight
$\ominus$ restricted to dynamics of manipulator $\ominus$ limited by payload capacity of robot	$\ominus$ coordinated control needed $\ominus$ more sensitive to wind disturbances

Table 2.4: Summary of major building blocks for robot-assisted take-off and landing of flying robots.

Take-off phases		**Landing phases**	
T1	- surface motion estimation - take-off trajectory - clamping mechanism - coordinated control or - high payload robot manipulator	L1	- approach trajectory - UAV flight control
T2	- surface motion estimation - releasable clamping mechanism - compliant release handling - release detection - controller switch	L2	- surface motion estimation - relative pose estimation - capturing mechanism - compliant contact handling - capturing detection - controller switch
T3	- departure trajectory - UAV flight control	L3	- surface motion estimation - landing trajectory - clamping mechanism - coordinated control or - high payload robot manipulator

Forces and torques of a flying robot are coupled. If the flying robot turns of its rotors, only the fully clamped case can be realized. In the partially clamped case, the flying robots actuators are active and the aim of the robot manipulator is to support during take-off and landing. Active control of the flying robot can range from hover or gravity compensation to trajectory following. As shown in Table 2.3, the two cases have different advantages and disadvantages. From the control point of view, the partially clamped case is more interesting and, therefore, primarily studied in this thesis.

In summary, the modules and building blocks listed in Table 2.4 are required during the different phases introduced in this section.

2.4 Robotic support system dynamics

In this section, the modeling of rigid multibody dynamics is presented briefly. As soon as the flying robot is connected to the end-effector of the manipulator, the dynamics become coupled. Hence, the aim of this section is to develop a model of the combined system consisting of robot manipulator and flying robot. Furthermore, suitable simplifications are derived, such that less model knowledge is required and the complexity of the controller design is reduced.

First, attitude representations and attitude kinematics are concisely reviewed. Then, a synthetic modeling procedure, the Projection Equation approach, is presented briefly. The equations of the combined system model are introduced, from which special cases, such as floating base robot manipulator system, large floating base manipulator system, fixed base system, and planar motion model can be derived. Using a representative example consisting of two rigid bodies, important properties of the combined model are highlighted and simplifications are discussed.

2.4.1 Attitude representations

Here, the three widely used representations for describing the attitude of robotic systems are presented briefly: rotation matrices, Euler angles, and quaternions. Their properties are compared in Table 2.5. An attitude representation is minimal, if the number of parameters is equal to the number of rotational DoF of the system, i.e. 3. Minimal attitude representations do not allow to globally define continuous control laws and, hence, are not global and limited to local attitude maneuvers [26]. The attitude representation is not singularity-free, if areas exist where a jump in the coordinates occurs and where the velocity goes to infinity. And it is unique, if the space of rotations $SO(3)$ (defined below in (2.1)) is covered only once.

Table 2.5: Important properties of attitude representations.

Attitude representation	Minimal?	Singularity-free?	Global?	Unique?
Euler angles	✓	✗	✗	✗
Rodrigues parameters	✓	✗	✗	✗
Angle-axis	✗	✗	✓	✗
Quaternions	✗	✓	✓	✗
Rotation matrix	✗	✓	✓	✓

Rotation matrices

Rotation matrices $\boldsymbol{R} \in SO(3)$ are an established attitude representation [27]. The special orthogonal group $SO(3)$ is defined as

$$SO(3) = \{\boldsymbol{R} | \boldsymbol{R} \in \mathbb{R}^{3\times3}, \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{E}, \det \boldsymbol{R} = 1\}, \tag{2.1}$$

for a right-handed coordinate system and with an identity matrix $\boldsymbol{E} \in \mathbb{R}^{3\times3}$ [26]. A vector ${}^{i}\boldsymbol{v} \in \mathbb{R}^3$ specified in frame i is transformed to frame b via

$${}^{b}\boldsymbol{v} = \boldsymbol{R}_{bi}\,{}^{i}\boldsymbol{v}. \tag{2.2}$$

For rotation matrices $\boldsymbol{R}$, it always holds that $\boldsymbol{R}^T = \boldsymbol{R}^{-1}$, where $\boldsymbol{R}^{-1}$ represents the inverse rotation. Therefore, it follows

$${}^{i}\boldsymbol{v} = \boldsymbol{R}_{bi}^{-1}\,{}^{b}\boldsymbol{v} = \boldsymbol{R}_{bi}^{T}\,{}^{b}\boldsymbol{v} = \boldsymbol{R}_{ib}\,{}^{b}\boldsymbol{v}. \tag{2.3}$$

Rotation matrix time derivative

Consider the angular velocity vector ${}^{b}\boldsymbol{\omega}_{ib} = \begin{pmatrix} {}^{b}\omega_x & {}^{b}\omega_y & {}^{b}\omega_z \end{pmatrix}_{ib}^{T}$ of a frame b relative to a frame i expressed in frame b. Then the time derivative of the rotation matrix is given by ([28], p. 258)

$$\dot{\boldsymbol{R}}_{bi} = -\boldsymbol{S}({}^{b}\boldsymbol{\omega}_{ib})\,\boldsymbol{R}_{bi} \tag{2.4}$$

exists, where

$$\boldsymbol{S}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \tag{2.5}$$

is the skew-symmetric matrix representation of the cross product $\boldsymbol{S}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ s. t. $\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$ for every $\boldsymbol{a} \in \mathbb{R}^3$ and $\boldsymbol{b} \in \mathbb{R}^3$. Therefore, ${}^{b}\boldsymbol{\omega}_{ib}$ is obtained from

$$\boldsymbol{S}({}^{b}\boldsymbol{\omega}_{ib}) = -\dot{\boldsymbol{R}}_{bi}\boldsymbol{R}_{bi}^{-1} = \boldsymbol{R}_{bi}\dot{\boldsymbol{R}}_{bi}^{-1} \tag{2.6}$$

by extracting the corresponding element of ${}^{b}\boldsymbol{\omega}_{ib}$ from $\boldsymbol{S}({}^{b}\boldsymbol{\omega}_{ib})$ with the operation ${}^{b}\boldsymbol{\omega}_{ib} = \boldsymbol{S}({}^{b}\boldsymbol{\omega}_{ib})^{\vee}$, where $(\cdot)^{\vee}$ denotes the vee map [29]

$$(\cdot)^{\vee} : SO(3) \to \mathbb{R}^3. \tag{2.7}$$

Operation (2.7) can be considered the inverse of operation $\boldsymbol{S}(\cdot)$.

For completeness, the angle-axis representation and the Rodrigues parameters are also mentioned here (see Table 2.5).

Angle-axis representation

Given a rotation matrix $\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$, a rotation may be defined via an angle

$$\varphi = \text{arcccos}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \tag{2.8}$$

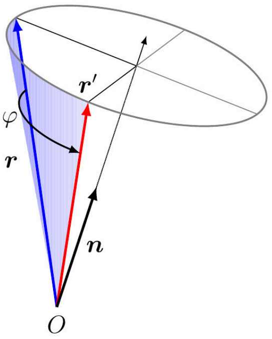


Figure 2.6: Rotation with angle φ about the rotation axis $\boldsymbol{n}$.

and an axis (see Figure 2.6)

$$\boldsymbol{n} = \frac{1}{2\sin(\varphi)} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}, \tag{2.9}$$

which is obviously singular for $\varphi = n\pi$ with $n \in \mathbb{Z}$ [27].

(Unmodified) Rodrigues parameters

The Rodrigues parameters are defined as

$$\boldsymbol{r} = \begin{pmatrix} r_1 & r_2 & r_3 \end{pmatrix}^T = \varphi \boldsymbol{n}, \tag{2.10}$$

where φ is the rotation angle and $\boldsymbol{n} \in \mathbb{R}^3$ is the so-called Euler axis [27]. The representation is minimal, because only three parameters are required, but singular at $\varphi = n\pi$ with $n \in \mathbb{Z}$ and not unique, because the axis is not uniquely defined if $\varphi = 0$ [27].

Euler angles

Euler's rotation theorem states that, in three-dimensional space, any displacement of a rigid body can be decomposed into three successive rotations about three axes x, y, and z with three rotation angles φ, θ, and ψ [28]. From the 27 possible sequences of rotations about the three axes x, y, and z, only twelve satisfy the constraint that no two consecutive axes are equal [27, 30]. For aerospace applications the z,y',x"- convention is applied. First, a rotation ψ about the z-axis, then a rotation θ about the new y'-axis and finally a rotation φ about the new x"-axis is performed. ψ is referred to as yaw, θ as pitch and φ as roll angle.

A rotation from a frame b to a frame i can be represented by

$$\begin{aligned} \boldsymbol{R}_{bi}(\psi,\theta,\varphi) &= \boldsymbol{R}_x(\varphi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi) = \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c(\psi)c(\theta) & s(\psi)c(\theta) & -s(\theta) \\ c(\psi)s(\theta)s(\varphi) - s(\psi)c(\varphi) & s(\psi)s(\theta)s(\varphi) + c(\psi)c(\varphi) & c(\theta)s(\varphi) \\ c(\psi)s(\theta)c(\varphi) + s(\psi)c(\varphi) & s(\psi)s(\theta)c(\varphi) - c(\psi)s(\varphi) & c(\theta)c(\varphi) \end{bmatrix}, \end{aligned} \tag{2.11}$$

with $c(\cdot) = \cos(\cdot)$ and $s(\cdot) = \sin(\cdot)$.

Euler angle time derivative

The angular velocity vector ${}^b\boldsymbol{\omega}_{ib}$ can be written using Euler angles via (2.6) as

$$\begin{aligned} {}^b\boldsymbol{\omega}_{ib} &= \boldsymbol{R}_x(\varphi)\boldsymbol{R}_y(\theta)\begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} + \boldsymbol{R}_x(\varphi)\begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \begin{pmatrix} \dot{\varphi} \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\varphi) & \sin(\varphi)\cos(\theta) \\ 0 & -\sin(\varphi) & \cos(\varphi)\cos(\theta) \end{bmatrix} \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}. \end{aligned} \tag{2.12}$$

Solving for the time derivatives of the Euler angles, yields

$$\begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \frac{1}{\cos(\theta)} \begin{bmatrix} \cos(\theta) & \sin(\varphi)\sin(\theta) & \cos(\varphi)\sin(\theta) \\ 0 & \cos(\varphi)\cos(\theta) & -\sin(\varphi)\cos(\theta) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{bmatrix} \begin{pmatrix} {}^b\omega_x \\ {}^b\omega_y \\ {}^b\omega_z \end{pmatrix}_{ib}. \tag{2.13}$$

For $\theta = \pm(2k+1)\frac{\pi}{2}$ with $k \in \mathbb{N}_0$ the matrix in (2.13) is singular, i.e. no solution exists.

Quaternions

Another way to represent the attitude of a rigid body is to use quaternions. A quaternion $\boldsymbol{q}$ is defined as [27, 28]

$$\boldsymbol{q} = \begin{pmatrix} \eta \\ \boldsymbol{\epsilon} \end{pmatrix}, \tag{2.14}$$

with the one-dimensional scalar part

$$\eta = \cos\left(\frac{\varphi}{2}\right) \tag{2.15}$$

and the three-dimensional vector part

$$\boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{pmatrix} = \sin\left(\frac{\varphi}{2}\right)\boldsymbol{n}, \tag{2.16}$$

where $\boldsymbol{n} \in \mathbb{R}^3$ is the rotation axis and $\varphi \in \mathbb{R}$ is the rotation angle, as depicted in Figure 2.6. Using four values for parametrizing a rotation in $\mathbb{R}^3$ leads to a non-minimal attitude parametrization. Hence, a constraint has to be introduced. The constraint for the absolute value of the quaternion

$$||\boldsymbol{q}|| = \sqrt{\boldsymbol{q}^T\boldsymbol{q}} = \sqrt{\eta^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2} \tag{2.17}$$

is

$$\boldsymbol{q}^T\boldsymbol{q} = ||\boldsymbol{n}||\sin^2\left(\frac{\varphi}{2}\right) + \cos^2\left(\frac{\varphi}{2}\right) = 1. \tag{2.18}$$

Quaternions with $\|\boldsymbol{q}\| = 1$ are called unit quaternions. Note that only the rotation axis $\boldsymbol{n}$ has to be normalized to obtain a unit quaternion. Since only unit quaternions are used in this work, they are from now on simply referred to as quaternions. Assume the quaternion $\boldsymbol{q}$ represents the rotation from a frame b to a frame i. In this context, the question how to describe sequences of rotations arises. A quaternion $\boldsymbol{q}$ can be converted to a rotation matrix $\boldsymbol{R}$ using the Euler-Rodrigues formula

$$\boldsymbol{R}_{bi}(\boldsymbol{n},\varphi) = \cos(\varphi)\boldsymbol{E} + (1-\cos(\varphi))\,\boldsymbol{n}\boldsymbol{n}^T + \sin(\varphi)\boldsymbol{S}(\boldsymbol{n}). \tag{2.19}$$

Inserting the definitions (2.15) and (2.16) in (2.19) yields the conversion formula

$$\boldsymbol{R}_{bi}(\boldsymbol{q}) = \left(\eta^2 - \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right)\boldsymbol{E} + 2\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T + 2\eta\boldsymbol{S}(\boldsymbol{\epsilon}). \tag{2.20}$$

Note that $\boldsymbol{R}_{bi}(\boldsymbol{q})$ and $\boldsymbol{q}$ describe the same rotation and the inverse rotation $\boldsymbol{q}^{-1}$ analog to $\boldsymbol{R}_{ib}$ is therefore given by

$$\boldsymbol{q}^{-1} = \begin{bmatrix}\eta & -\epsilon_1 & -\epsilon_2 & -\epsilon_3\end{bmatrix}^T. \tag{2.21}$$

The rotation matrix for two subsequently executed rotations $\boldsymbol{q}$ and $\boldsymbol{q}'$ can be written as

$$\boldsymbol{R}(\boldsymbol{q}'') = \boldsymbol{R}(\boldsymbol{q})\boldsymbol{R}(\boldsymbol{q}'), \tag{2.22}$$

with

$$\boldsymbol{q}'' = \boldsymbol{Q}(\boldsymbol{q})\boldsymbol{q}' = \boldsymbol{q}\otimes\boldsymbol{q}' = \boldsymbol{U}(\boldsymbol{q}')\boldsymbol{q}. \tag{2.23}$$

The matrices $\boldsymbol{Q}$ and $\boldsymbol{U}$ are found by inserting (2.20) in (2.22) and reorganising the result according to (2.23). One obtains

$$\boldsymbol{Q}(\boldsymbol{q}) = \begin{bmatrix}\eta & -\epsilon_1 & -\epsilon_2 & -\epsilon_3\\ \epsilon_1 & \eta & -\epsilon_3 & \epsilon_2\\ \epsilon_2 & \epsilon_3 & \eta & -\epsilon_1\\ \epsilon_3 & -\epsilon_2 & \epsilon_1 & \eta\end{bmatrix} = \begin{bmatrix}\eta & -\boldsymbol{\epsilon}^T\\ \boldsymbol{\epsilon} & \eta\boldsymbol{E}+\boldsymbol{S}(\boldsymbol{\epsilon})\end{bmatrix}, \quad \boldsymbol{U}(\boldsymbol{q}') = \begin{bmatrix}\eta' & -\boldsymbol{\epsilon}'^T\\ \boldsymbol{\epsilon}' & \eta'\boldsymbol{E}-\boldsymbol{S}(\boldsymbol{\epsilon}')\end{bmatrix}. \tag{2.24}$$

The matrices $\boldsymbol{Q}$ and $\boldsymbol{U}$ in (2.24) define the quaternion multiplication denoted by ⊗ in (2.23). Note that the multiplication of two unit quaternions yields again a unit quaternion and that the quaternion multiplication is non-commutative, i.e. $\boldsymbol{q}'\otimes\boldsymbol{q} \neq \boldsymbol{q}\otimes\boldsymbol{q}'$.

Quaternion time derivative

The time derivative of a quaternion $\boldsymbol{q}$, denoted by $\dot{\boldsymbol{q}}$, can be calculated from the angular velocity vector ${}^b\boldsymbol{\omega}_{ib}$ using ([28], p. 265)

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{Q}_q(\boldsymbol{q})\,{}^b\boldsymbol{\omega}_{ib} = \frac{1}{2}\begin{bmatrix}-\epsilon_1 & -\epsilon_2 & -\epsilon_3\\ \eta & -\epsilon_3 & \epsilon_2\\ \epsilon_3 & \eta & -\epsilon_1\\ -\epsilon_2 & \epsilon_1 & \eta\end{bmatrix}\begin{pmatrix}{}^b\omega_x\\ {}^b\omega_y\\ {}^b\omega_z\end{pmatrix}_{bi} = \frac{1}{2}\begin{bmatrix}-\boldsymbol{\epsilon}^T\\ \eta\boldsymbol{E}+\boldsymbol{S}(\epsilon)\end{bmatrix}{}^b\boldsymbol{\omega}_{ib}. \tag{2.25}$$

Note that $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^T$, because $\|\boldsymbol{q}\| = 1$, and that $\boldsymbol{Q}_q(\boldsymbol{q})$ is equal to the matrix $\boldsymbol{Q}(\boldsymbol{q})$ in (2.24) without its first column.

The elements η, ϵ_1, ϵ_2, ϵ_3 of the quaternion $\boldsymbol{q}$ are the so-called Euler parameters, which form the connection between all discussed attitude representations. It can be seen from the definition (2.14) - (2.16) and from Figure 2.6 that the quaternions $\boldsymbol{q}$ and -$\boldsymbol{q}$ represent the same rotation, therefore $\boldsymbol{q}$ is not unique. Nevertheless, quaternions are advantageous compared to rotation matrices, due to the compact form of (2.25) and because only four parameters are used instead of nine, and to Euler angles, which are kinematically singular since the transformation from their time derivative to the angular velocity vector is not globally defined (see (2.13) and [26]).

2.4.2 Synthetic dynamics modeling

A generic rigid multibody modeling approach is chosen to derive the equations of motion of the combined system. In the considered procedure, the so-called Projection Equation [31], every calculation step is implicitly already carried out as far as possible, thus, only problem specific quantities need to be inserted. It allows to conveniently combine single rigid bodies to subsystems which are then combined to obtain the final multibody system. Alternative approaches are for instance the Lagrange formulation ([24], p. 247), the iterative Newton-Euler dynamics ([14], p. 168), or the closed-form computation of the dynamics matrices [32] based on [33, 34]. An implementation of the latter approach is readily available at the DLR Institute of Robotics and Mechatronics and, therefore, used in the simulations presented in Chapter 5. A formal comparison of the different approaches can be found in [31]. The Projection Equation method is the last one in the triangle of methods shown in [31] on page 74, and hence, the approach with minimum modeling effort. The following introduction to the Projection Equation is based on [31, 35] and the brief summary in the author's paper [214].

Projection modeling approach

Using the Projection Equation approach ([31], p. 70), the equations of motion of a rigid multibody system composed of N_s subsystems (cf. Figure 2.7) are derived symbolically in

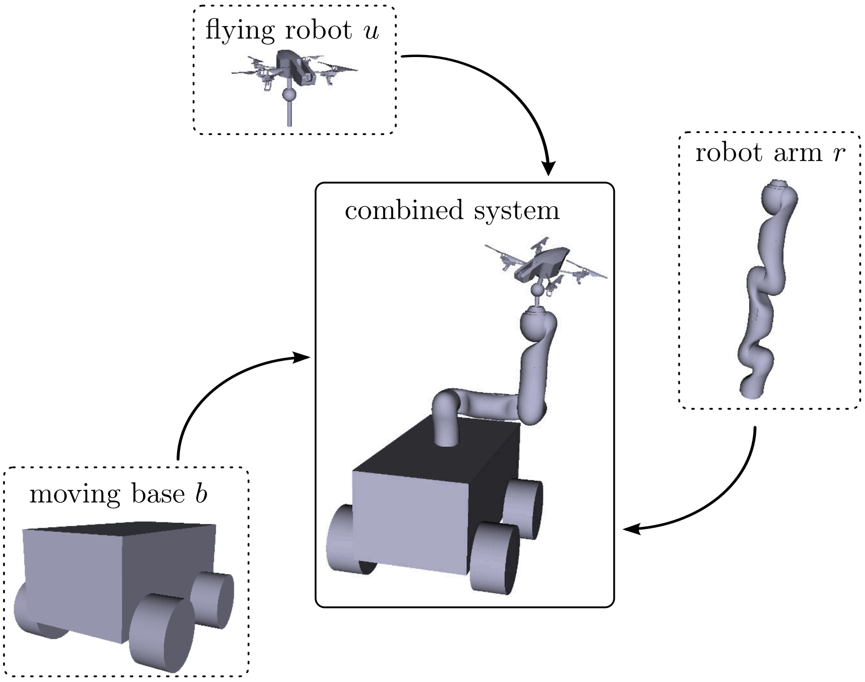


Figure 2.7: Illustration of the modeling approach. The combined system is composed of $N_s = 3$ subsystems, each consisting of N_n, $n \in \{1, 2, 3\}$, single rigid-bodies. Note that a floating base with six DoF is considered in this work, whereas the wheeled base is a special non-holonomic case. As in [36], the robot manipulator is treated as an extension of the kinematics of the floating base. In addition, the flying robot is attached via a ball joint and adds three DoF to the kinematic chain.

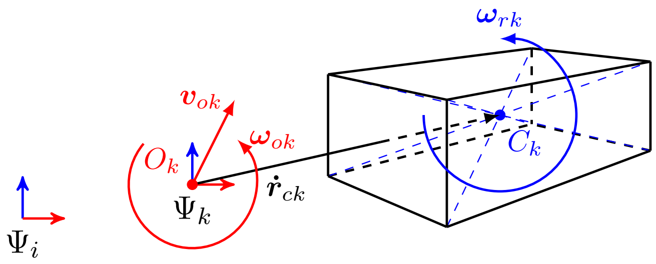

$\boldsymbol{\omega}_{rk}$
$\boldsymbol{v}_{ok}$
$\boldsymbol{\omega}_{ok}$
O_k
C_k
$\dot{\boldsymbol{r}}_{ck}$
Ψ_k
Ψ_i

Figure 2.8: Rigid body k with center of mass C_k, inertial frame Ψ_i, and reference frame Ψ_k with origin O_k. Note that the rigid body can translate and rotate relative to the moving reference frame.

the following. Consider a single rigid body k as depicted in Figure 2.8 and let O_k be the origin of a chosen reference frame. The minimal velocities of body k are collected in a vector as

$$\dot{\boldsymbol{y}}_k = \begin{pmatrix} \boldsymbol{v}_o^T & \boldsymbol{\omega}_o^T & \dot{\boldsymbol{r}}_c^T & \boldsymbol{\omega}_r^T \end{pmatrix}_k^T, \tag{2.26}$$

where $\boldsymbol{v}_{ok}$ is the absolute velocity of O_k, $\boldsymbol{\omega}_{ok}$ is the absolute angular velocity of body k (i.e. w.r.t. the inertial frame i), $\dot{\boldsymbol{r}}_{ck}$ is the translational velocity of the center of mass C_k and $\boldsymbol{\omega}_{rk}$ is the relative rotational velocity w.r.t. O_k, respectively. The equations of motion of a single body k are described using the inertia matrices $\overline{\boldsymbol{M}}_k$, the Coriolis/centrifugal matrices $\overline{\boldsymbol{C}}_k$, and the generalized force vectors $\overline{\boldsymbol{Q}}_k$ as

$$\overline{\boldsymbol{M}}_k \ddot{\boldsymbol{y}}_k + \overline{\boldsymbol{C}}_k \dot{\boldsymbol{y}}_k = \underbrace{\overline{\boldsymbol{Q}}_{ext,k} + \overline{\boldsymbol{Q}}_{ctrl,k}}_{\overline{\boldsymbol{Q}}_k}, \tag{2.27}$$

for which $\overline{\boldsymbol{M}}_k$ and $\overline{\boldsymbol{C}}_k$ can be derived as ([31] p. 80)

$$\overline{\boldsymbol{M}}_k = \begin{bmatrix} m\boldsymbol{E} & m\boldsymbol{S}(\boldsymbol{r}_c^T) & m\boldsymbol{E} & \boldsymbol{0} \\ m\boldsymbol{S}(\boldsymbol{r}_c) & \boldsymbol{I}^o & m\boldsymbol{S}(\boldsymbol{r}_c) & \boldsymbol{I}^c \\ m\boldsymbol{E} & m\boldsymbol{S}(\boldsymbol{r}_c^T) & m\boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}^c & \boldsymbol{0} & \boldsymbol{I}^c \end{bmatrix}_k \tag{2.28}$$

and

$$\overline{\boldsymbol{C}}_k = \begin{bmatrix} m\boldsymbol{S}(\boldsymbol{\omega}_o) & m\boldsymbol{S}(\boldsymbol{\omega}_o)\boldsymbol{S}(\boldsymbol{r}_c^T) & 2m\boldsymbol{S}(\boldsymbol{\omega}_o) & \boldsymbol{0} \\ m\boldsymbol{S}(\boldsymbol{r}_c)\boldsymbol{S}(\boldsymbol{\omega}_o) & \boldsymbol{S}(\boldsymbol{\omega}_o)\boldsymbol{I}^o + \dot{\boldsymbol{I}}^c & 2m\boldsymbol{S}(\boldsymbol{r}_c)\boldsymbol{S}(\boldsymbol{\omega}_o) & \boldsymbol{S}(\boldsymbol{\omega}_o)\boldsymbol{I}^c + \dot{\boldsymbol{I}}^c \\ m\boldsymbol{S}(\boldsymbol{\omega}_o) & m\boldsymbol{S}(\boldsymbol{\omega}_o)\boldsymbol{S}(\boldsymbol{r}_c^T) & 2m\boldsymbol{S}(\boldsymbol{\omega}_o) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}(\boldsymbol{\omega}_o)\boldsymbol{I}^c + \dot{\boldsymbol{I}}^c & \boldsymbol{0} & \boldsymbol{S}(\boldsymbol{\omega}_o)\boldsymbol{I}^c + \dot{\boldsymbol{I}}^c \end{bmatrix}_k . \tag{2.29}$$

Therein, m_k is the mass of the rigid body k, $\boldsymbol{r}_{ck}$ is the vector from O_k to C_k expressed in the reference frame k (see Figure 2.8), and $\boldsymbol{E} \in \mathbb{R}^{3\times3}$ is an identity matrix. The inertia of body k w.r.t. C_k and w.r.t. O_k, both expressed in the reference frame k, are denoted with $\boldsymbol{I}_k^c$ and $\boldsymbol{I}_k^o$, respectively. Note that $\boldsymbol{I}_k^o = \boldsymbol{I}_k^c + m_k\boldsymbol{S}(\boldsymbol{r}_{ck})\boldsymbol{S}(\boldsymbol{r}_{ck})^T$ according to the Huygens-Steiner theorem ([31] p. 80). The general force vector $\overline{\boldsymbol{Q}}_k$ is divided into external forces $\overline{\boldsymbol{Q}}_{ext,k}$ and control inputs $\overline{\boldsymbol{Q}}_{ctrl,k}$.

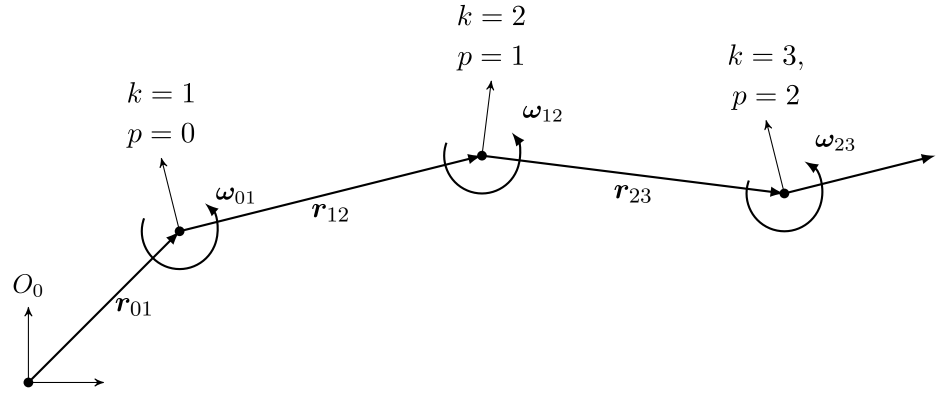


Figure 2.9: Exemplary kinematic chain.

Assembly of the multibody system is performed in two steps. The subsystems (each consisting of N_n rigid bodies) are constructed using the Jacobians $\overline{\boldsymbol{J}}_k = \frac{\partial \dot{\overline{\boldsymbol{y}}}_k}{\partial \dot{\boldsymbol{y}}_n}$, which map single rigid body velocities $\dot{\overline{\boldsymbol{y}}}_k$ to subsystem velocities $\dot{\boldsymbol{y}}_n$:

$$\boldsymbol{M}_n = \sum_{k=1}^{N_n} \overline{\boldsymbol{J}}_k^T \overline{\boldsymbol{M}}_k \overline{\boldsymbol{J}}_k, \tag{2.30}$$

$$\boldsymbol{C}_n = \sum_{k=1}^{N_n} \overline{\boldsymbol{J}}_k^T \left(\overline{\boldsymbol{C}}_k \overline{\boldsymbol{J}}_k + \overline{\boldsymbol{M}}_k \dot{\overline{\boldsymbol{J}}}_k \right), \tag{2.31}$$

$$\boldsymbol{Q}_n = \sum_{k=1}^{N_n} \overline{\boldsymbol{J}}_k^T \overline{\boldsymbol{Q}_k}. \tag{2.32}$$

The kinematic chain (illustrated in Figure 2.9) is described by

$$\dot{\boldsymbol{y}}_k = \boldsymbol{T}_{kp} \dot{\boldsymbol{y}}_p + \boldsymbol{J}_k \dot{\boldsymbol{s}}_k, \tag{2.33}$$

where $\dot{\boldsymbol{y}}_p$ is the minimal velocity of the predecessing element, i.e. $p = k-1$, and $\boldsymbol{s}_k$ is the vector of minimal state coordinates of the rigid body k. If only rotational relative motions are considered, e.g. for a robot with only rotating and twisting joints, the transformation matrices $\boldsymbol{T}_{kp}$ are given by ([35], p. 57)

$$\boldsymbol{T}_{kp} = \begin{bmatrix} \boldsymbol{R}_{kp} & \boldsymbol{R}_{kp}\boldsymbol{S}(\boldsymbol{r}_{pk}^T) & \boldsymbol{R}_{kp}\boldsymbol{S}(\boldsymbol{r}_{pk}^T)\boldsymbol{e}_{\mathrm{ax}} \\ \boldsymbol{0} & \boldsymbol{R}_{kp} & \boldsymbol{R}_{kp}\boldsymbol{e}_{\mathrm{ax}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \tag{2.34}$$

wherein $\boldsymbol{R}_{kp}$ is a rotation matrix that transforms a vector given in predecessor frame p to successor frame k, $\boldsymbol{r}_{pk}$ is the distance vector from O_p to O_k expressed in frame p, and $\boldsymbol{e}_{\mathrm{ax}}$ is a unit vector defining the axis of rotation. Finally, the multibody dynamics are obtained as ([31], p. 92)

$$\underbrace{\begin{bmatrix} \boldsymbol{J}_1^T & \cdots & \boldsymbol{J}_1^T \boldsymbol{T}_{N_s,1}^T \\ & \ddots & \vdots \\ & & \boldsymbol{J}_{N_s}^T \end{bmatrix}}_{\boldsymbol{J}_{tri}^T} \begin{pmatrix} \boldsymbol{M}_1 \ddot{\boldsymbol{y}}_1 + \boldsymbol{C}_1 \dot{\boldsymbol{y}}_1 - \boldsymbol{Q}_1 \\ \vdots \\ \boldsymbol{M}_N \ddot{\boldsymbol{y}}_{N_s} + \boldsymbol{C}_{N_s} \dot{\boldsymbol{y}}_{N_s} - \boldsymbol{Q}_{N_s} \end{pmatrix} = \boldsymbol{0}, \tag{2.35}$$

which leads to the minimal representation

$$\boldsymbol{M}\ddot{\boldsymbol{s}} + \boldsymbol{C}\dot{\boldsymbol{s}} - \boldsymbol{Q} = \boldsymbol{0}, \tag{2.36}$$

with $\boldsymbol{s}$ being the vector of minimal state coordinates of the combined multibody system and

$$\boldsymbol{M} = \boldsymbol{J}_{tri}^T \begin{bmatrix} \boldsymbol{M}_1 & & \\ & \ddots & \\ & & \boldsymbol{M}_{N_s} \end{bmatrix} \boldsymbol{J}_{tri}, \tag{2.37}$$

$$\boldsymbol{C} = \boldsymbol{J}_{tri}^T \begin{bmatrix} \boldsymbol{C}_1 & & \\ & \ddots & \\ & & \boldsymbol{C}_{N_s} \end{bmatrix} \boldsymbol{J}_{tri} + \boldsymbol{J}_{tri}^T \begin{bmatrix} \boldsymbol{M}_1 & & \\ & \ddots & \\ & & \boldsymbol{M}_{N_s} \end{bmatrix} \dot{\boldsymbol{J}}_{tri} \tag{2.38}$$

$$\boldsymbol{Q} = \boldsymbol{J}_{tri}^T \begin{pmatrix} \boldsymbol{Q}_1^T & \cdots & \boldsymbol{Q}_{N_s}^T \end{pmatrix}^T. \tag{2.39}$$

This procedure projects the generalized forces in the direction of unconstrained motion [31].

Special case: A single rigid body

The combined system of floating base, manipulator, and flying robot is of course a multibody system. The flying robot alone is usually described using a single rigid body. If the reference point O coincides with the center of mass C, it holds that $\boldsymbol{r}_c = \boldsymbol{0}$, $\dot{\boldsymbol{r}}_c = \boldsymbol{0}$, and $\boldsymbol{\omega}_r = \boldsymbol{0}$. Here, the index k is dropped for the sake of brevity. The fundamental matrices (2.28) and (2.29) then reduce to

$$\boldsymbol{M} = \begin{bmatrix} m\boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}^o \end{bmatrix}, \qquad \boldsymbol{C} = \begin{bmatrix} m\boldsymbol{S}(\boldsymbol{\omega}_o) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}(\boldsymbol{\omega}_o)\boldsymbol{I}^o \end{bmatrix}, \tag{2.40}$$

where the $\overline{(.)}$ notation is omitted, because no further projection is required. Evaluating (2.36) with $\dot{\boldsymbol{s}} = \begin{pmatrix} \boldsymbol{v}_o^T & \boldsymbol{\omega}_o^T \end{pmatrix}^T$ yields the well-known Newton-Euler equations, which describe the rotational and the translational dynamics of a single rigid body. They can concisely be written in the body-fixed reference frame as

$$\begin{bmatrix} m\boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}^o \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{v}}_o \\ \dot{\boldsymbol{\omega}}_o \end{pmatrix} + \begin{pmatrix} m\boldsymbol{S}(\boldsymbol{\omega}_o)\boldsymbol{v}_o \\ \boldsymbol{S}(\boldsymbol{\omega}_o)\boldsymbol{I}^o\boldsymbol{\omega}_o \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{pmatrix}, \tag{2.41}$$

where $\boldsymbol{f}$ and $\boldsymbol{\tau}$ are forces and torques expressed in the body-fixed frame, respectively.

Recursive algorithm and inner constraint computation

The multibody dynamics model (2.36) can directly be used in simulation, but then the inertia matrix has to be inverted in every iteration step. Since this becomes expensive for higher order systems, a recursive algorithm is used instead, which takes advantage of the triangular form of $\boldsymbol{J}_{tri}^T$ in (2.36). Moreover, it allows to compute the constraint forces acting in each joint explicitly. It is adopted from ([31], pp. 95-99) wherein a detailed proof is

Algorithm 2.1: Recursive dynamics computation

```
while True do
    Computation of forward kinematics:
    for $k \in \{1, \dots, N_s\}$ do
        Set $p = k - 1$;
        Compute $\boldsymbol{T}_{kp}$, $\dot{\boldsymbol{T}}_{kp}$, $\dot{\boldsymbol{y}}_k$, and $\boldsymbol{h}_k = \boldsymbol{C}_k \dot{\boldsymbol{y}}_k - \boldsymbol{Q}_k$
    Set $\boldsymbol{M}^*_{N_s} = \boldsymbol{M}_{N_s}$ and $\boldsymbol{h}^*_{N_s} = \boldsymbol{h}_{N_s}$;
    Computation of modified dynamics matrices:
    for $p \in \{N_s - 1, \dots, 1\}$ do
        Set $k = p + 1$ and compute $\hat{\boldsymbol{M}}_k = \boldsymbol{J}_k^T \boldsymbol{M}_k^* \boldsymbol{J}_k$, $\boldsymbol{N}_k = \boldsymbol{E} - \boldsymbol{M}_k^*(\boldsymbol{J}_k \hat{\boldsymbol{M}}_k^{-1} \boldsymbol{J}_k^T)$,
          $\boldsymbol{M}_p^* = \boldsymbol{M}_p + \boldsymbol{T}_{kp}^T \boldsymbol{N}_k \boldsymbol{M}_k^* \boldsymbol{T}_{kp}$, and $\boldsymbol{h}_p^* = \boldsymbol{h}_p + \boldsymbol{T}_{kp}^T \boldsymbol{N}_k (\boldsymbol{M}_k^* \dot{\boldsymbol{T}}_{kp} \dot{\boldsymbol{y}}_p + \boldsymbol{h}_k^*)$
    Computation of minimal accelerations:
    for $k \in \{1, \dots, N_s\}$ do
        Set $p = k - 1$ and compute $\ddot{\boldsymbol{s}}_k = -\hat{\boldsymbol{M}}_k^{-1} \boldsymbol{J}_k^T (\boldsymbol{M}_k^* (\boldsymbol{T}_{kp} \ddot{\boldsymbol{y}}_p + \dot{\boldsymbol{T}}_{kp} \dot{\boldsymbol{y}}_p) + \boldsymbol{h}_k^*)$,
          $\ddot{\boldsymbol{y}}_k = \boldsymbol{T}_{kp} \ddot{\boldsymbol{y}}_p + \dot{\boldsymbol{T}}_{kp} \dot{\boldsymbol{y}}_p + \boldsymbol{J}_k \ddot{\boldsymbol{s}}_k$, and $\boldsymbol{Q}_k^\lambda = \boldsymbol{N}_k (\boldsymbol{M}_k^* \ddot{\boldsymbol{y}}_k + \boldsymbol{h}_k^*)$
```

sketched, and can be summarized as follows: First, calculate the forward kinematics starting from subsystem $k = 1$. Then, compute the modified dynamics matrices, denoted with a $*$, in a descending order starting with subsystem $p = N_s$. Finally, determine all minimal accelerations $\ddot{\boldsymbol{s}}_k$ for $k \in \{1, \dots, N_s\}$. Algorithm 2.1 comprises the calculation details.

The minimal accelerations are integrated numerically and then enter Algorithm 2.1 via equation (2.33) in every simulation time step. Note that the intermediate velocity $\dot{\boldsymbol{y}}_0$ and acceleration $\ddot{\boldsymbol{y}}_0$ are either zero, or provide a convenient way to account for base movement, which is unaffected by the multibody system motion, e.g. a manipulator on a large ship. The constraint force $\boldsymbol{Q}_k^\lambda$ depends on the dynamics of all its successors and has to fulfill the local orthogonality condition $\boldsymbol{J}_k^T \boldsymbol{Q}_k^\lambda = \boldsymbol{0}$, which holds if relative velocities are used as minimal velocities [31].

2.4.3 Combined model of robot manipulator and flying robot

Applying the modeling approach presented above yields the dynamics of the serial multibody system in configuration space. The minimal representation (2.36) is the same as the dynamics in joint coordinates $\boldsymbol{\phi} \in \mathbb{R}^n$ of a serial robotic manipulator with n joints ([37], p. 29)

$$\boldsymbol{M}(\boldsymbol{\phi})\ddot{\boldsymbol{\phi}} + \boldsymbol{C}(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}})\dot{\boldsymbol{\phi}} + \boldsymbol{g}(\boldsymbol{\phi}) = \boldsymbol{\tau}_j + \boldsymbol{\tau}_{ext}, \tag{2.42}$$

where $\boldsymbol{M}_{n \times n}$, $\boldsymbol{C}_{n \times n}$, and $\boldsymbol{g}_{n \times 1}$ are the inertia matrix, the Coriolis and centrifugal matrix, and the vector of gravity terms, respectively. For conciseness, from now on the argument $\boldsymbol{\phi}$ will be mostly dropped. Different representations of the combined model composed of floating base b, robot manipulator r, and flying robot (or VTOL UAV) u are discussed in

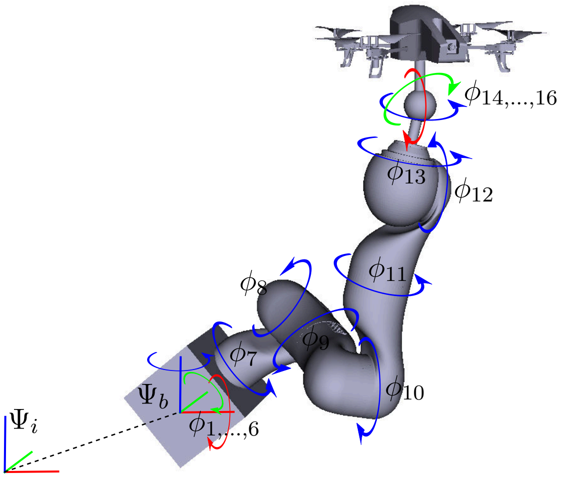


Figure 2.10: Illustration of the manipulator on a floating base with flying robot attached at the end-effector.

the following. All couplings between b, r, and u become apparent, if the matrices are written in block form, i.e.

$$\begin{bmatrix} \boldsymbol{M}_{bb} & \boldsymbol{M}_{br} & \boldsymbol{M}_{bu} \\ \boldsymbol{M}_{br}^T & \boldsymbol{M}_{rr} & \boldsymbol{M}_{ru} \\ \boldsymbol{M}_{bu}^T & \boldsymbol{M}_{ru}^T & \boldsymbol{M}_{uu} \end{bmatrix} \begin{pmatrix} \ddot{\boldsymbol{\phi}}_b \\ \ddot{\boldsymbol{\phi}}_r \\ \ddot{\boldsymbol{\phi}}_u \end{pmatrix} + \begin{bmatrix} \boldsymbol{C}_{bb} & \boldsymbol{C}_{br} & \boldsymbol{C}_{bu} \\ \boldsymbol{C}_{rb} & \boldsymbol{C}_{rr} & \boldsymbol{C}_{ru} \\ \boldsymbol{C}_{ub} & \boldsymbol{C}_{ur} & \boldsymbol{C}_{uu} \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{\phi}}_b \\ \dot{\boldsymbol{\phi}}_r \\ \dot{\boldsymbol{\phi}}_u \end{pmatrix} - \begin{pmatrix} \boldsymbol{Q}_b \\ \boldsymbol{Q}_r \\ \boldsymbol{Q}_u \end{pmatrix} = \boldsymbol{0}, \tag{2.43}$$

where $\boldsymbol{M}_{ij}$ are the blocks of the inertia matrix, which is positive definite and symmetric. Furthermore, $\boldsymbol{C}_{ij}$ are the blocks of the Coriolis/centrifugal matrix and $\boldsymbol{Q}_k$ comprises external forces and actuator forces acting on the three different subsystems.

Here, $\boldsymbol{\phi}_b \in \mathbb{R}^6$ are the states of the floating base, $\boldsymbol{\phi}_r \in \mathbb{R}^n$ are the states of the n-DoF robot manipulator, and $\boldsymbol{\phi}_u \in \mathbb{R}^3$ are the states of the UAV. Therefore, in the most general case, the combined system model has $6+n+3$ DoF. Note, that the three DoF of the flying robot are the result of a rigid connection to the manipulator which leaves the rotational DoF open, e.g. a ball joint (or universal hinge). Moreover, three states for describing the orientation of base or UAV are only sufficient, if a minimal attitude representation, e.g. Euler angles, is used. Since the latter have singularities, a non-minimal representation like quaternions is advantageous, resulting in $7+n+4$ states of the general combined system model. The above model is suitable for cases, where the motion of the floating base is influenced by the motion of manipulator and flying robot, e.g. if a small wheeled mobile robot is used for robot-assisted landing.

Large floating base case

For a robot manipulator operating on a large base, for example a heavy vehicle or a ship, it is reasonable to assume that the manipulator has no effect on the motion of the base, but vice versa. Under this assumption, (2.43) becomes

$$\begin{bmatrix} \boldsymbol{M}_{br}^T & \boldsymbol{M}_{rr} & \boldsymbol{M}_{ru} \\ \boldsymbol{M}_{bu}^T & \boldsymbol{M}_{ru}^T & \boldsymbol{M}_{uu} \end{bmatrix} \begin{pmatrix} \ddot{\boldsymbol{\phi}}_b \\ \ddot{\boldsymbol{\phi}}_r \\ \ddot{\boldsymbol{\phi}}_u \end{pmatrix} + \begin{bmatrix} \boldsymbol{C}_{rb} & \boldsymbol{C}_{rr} & \boldsymbol{C}_{ru} \\ \boldsymbol{C}_{ub} & \boldsymbol{C}_{ur} & \boldsymbol{C}_{uu} \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{\phi}}_b \\ \dot{\boldsymbol{\phi}}_r \\ \dot{\boldsymbol{\phi}}_u \end{pmatrix} - \begin{pmatrix} \boldsymbol{Q}_r \\ \boldsymbol{Q}_u \end{pmatrix} = \boldsymbol{0}. \tag{2.44}$$

This model allows to directly include measured or simulated base motions through $\ddot{\boldsymbol{\phi}}_b$ and $\dot{\boldsymbol{\phi}}_b$.

Planar motion case

A simplified version of (2.43) can be derived, if only planar motions of base, manipulator, and aerial vehicle are considered. In the planar case, the structure of (2.43) and (2.44) remains the same, but the dimensions of the state vectors reduce to $\boldsymbol{\phi}_b \in \mathbb{R}^3$, $\boldsymbol{\phi}_r \in \mathbb{R}^n$, and $\boldsymbol{\phi}_u \in \mathbb{R}^1$. This model neglects cross couplings in the angular velocities, but allows to derive simplified controllers, which need less model knowledge. The planar model is used in Section 4.4.1 and in Section 4.4.5 for a simulation case study.

Fixed base case

Lastly, a combined system is considered where the base is not moving. This allows to solely study the interaction between flying robot and manipulator. Then, (2.43) reduces to

$$\begin{bmatrix} \boldsymbol{M}_{rr} & \boldsymbol{M}_{ru} \\ \boldsymbol{M}_{ru}^T & \boldsymbol{M}_{uu} \end{bmatrix} \begin{pmatrix} \ddot{\boldsymbol{\phi}}_r \\ \ddot{\boldsymbol{\phi}}_u \end{pmatrix} + \begin{bmatrix} \boldsymbol{C}_{rr} & \boldsymbol{C}_{ru} \\ \boldsymbol{C}_{ur} & \boldsymbol{C}_{uu} \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{\phi}}_r \\ \dot{\boldsymbol{\phi}}_u \end{pmatrix} - \begin{pmatrix} \boldsymbol{Q}_r \\ \boldsymbol{Q}_u \end{pmatrix} = \boldsymbol{0}. \tag{2.45}$$

This model is considered mostly for the controller designs and experiments carried out at DLR with a LWR 4+ on a fixed base.

2.4.4 Task space formulation

For a given task, e.g. robot-assisted take-off or landing of flying robots, one is interested in the motion of a predefined point on the robotic system, e.g. the TCP of the manipulator, in a Cartesian reference frame. One possible representation of the dynamics in Cartesian coordinates is given by the task space (or operational space [38]) formulation. A pragmatic reason for using the task space formulation, is that it does not require to compute the inverse kinematics, which is prone to singularities and can be quite difficult to solve ([14], p. 102). On the contrary, the forward (or direct) kinematics $\boldsymbol{\chi} : \mathbb{R}^n \to \mathbb{R}^m$, which maps joint angles $\boldsymbol{\phi} \in \mathbb{R}^n$ to Cartesian end-effector coordinates $\boldsymbol{x} \in \mathbb{R}^m$, is straightforward to compute ([14], p. 62). The mapping between joint velocities $\dot{\boldsymbol{\phi}} \in \mathbb{R}^n$ and task velocities $\dot{\boldsymbol{x}} \in \mathbb{R}^m$ is given by the Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ ([14], pp. 135) as

$$\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{\chi}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \dot{\boldsymbol{\phi}} = \boldsymbol{J}(\boldsymbol{\phi}) \dot{\boldsymbol{\phi}}, \tag{2.46}$$

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{\phi}) \ddot{\boldsymbol{\phi}} + \dot{\boldsymbol{J}}(\boldsymbol{\phi}) \dot{\boldsymbol{\phi}}. \tag{2.47}$$

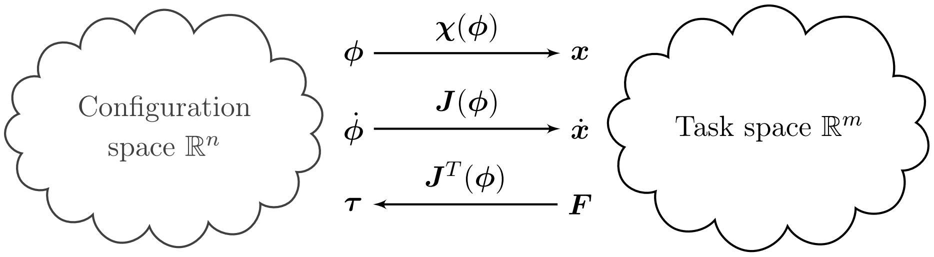

Configuration space $\mathbb{R}^n$
$\boldsymbol{\chi}(\boldsymbol{\phi})$
$\boldsymbol{\phi}$
$\boldsymbol{x}$
$\boldsymbol{J}(\boldsymbol{\phi})$
$\dot{\boldsymbol{\phi}}$
$\dot{\boldsymbol{x}}$
$\boldsymbol{J}^T(\boldsymbol{\phi})$
$\boldsymbol{\tau}$
$\boldsymbol{F}$
Task space $\mathbb{R}^m$

Figure 2.11: Illustration of transformations between configuration and task space.

Using (2.46) and (2.47), velocities and accelerations in task space can be computed from velocities and accelerations in joint space. For linear velocities $\boldsymbol{v}$, it is straightforward to calculate the partial derivatives of $\boldsymbol{\chi}$ w.r.t. $\boldsymbol{\phi}$, but there is no 3×1 orientation vector whose derivative is the angular velocity $\boldsymbol{\omega}$, i.e. $\boldsymbol{\omega}$ is not integrable. Therefore, it is more convenient in practice to compute the Jacobian matrix $\boldsymbol{J}$ from

$$\boldsymbol{J} = \frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{\phi}} = \frac{\partial \frac{\mathrm{d}\boldsymbol{\chi}}{\mathrm{d}t}}{\partial \frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t}} = \frac{\partial \dot{\boldsymbol{x}}}{\partial \dot{\boldsymbol{\phi}}}. \tag{2.48}$$

Note that because of $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{\phi} \in \mathbb{R}^n$, the Jacobian matrix (2.48) has m rows and n columns. It is clear that for $n > m$, i.e. a redundant robotic system with joint space greater than task space, the relation (2.46) is not directly invertible. Hence, for a redundant system the inverse relation

$$\dot{\boldsymbol{\phi}}_x - \dot{\boldsymbol{\phi}}_{nsp} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}} \tag{2.49}$$

requires a generalized pseudoinverse $\boldsymbol{J}^{\#}$ (see Appendix A3). A general nullspace projector[1] is given by [39]

$$\boldsymbol{N} = \boldsymbol{E} - \boldsymbol{J}^{\#}\boldsymbol{J}. \tag{2.50}$$

Hence, in the redundant case one has to distinguish between joint velocities $\dot{\boldsymbol{\phi}}_x = \boldsymbol{J}^{\#}\boldsymbol{J}\dot{\boldsymbol{\phi}}$, which affect the motion of the end-effector, and nullspace velocities $\dot{\boldsymbol{\phi}}_{nsp} = \boldsymbol{N}\dot{\boldsymbol{\phi}}$, which do not affect the motion of the end-effector.

Exploiting the principle of virtual work, one can show that for static forces $\boldsymbol{F} \in \mathbb{R}^m$ and joint torques $\boldsymbol{\tau} \in \mathbb{R}^n$ a relation similar to (2.46) exists ([14], pp. 156):

$$\begin{aligned} \boldsymbol{F}^T\delta\boldsymbol{x} &= \boldsymbol{\tau}^T\delta\boldsymbol{\phi}, \\ \boldsymbol{F}^T\boldsymbol{J}(\boldsymbol{\phi})\delta\boldsymbol{\phi} &= \boldsymbol{\tau}^T\delta\boldsymbol{\phi}. \end{aligned}$$

Therein, $\delta\boldsymbol{x}$ and $\delta\boldsymbol{\phi}$ denote infinitesimal deviations in $\boldsymbol{x}$ and $\boldsymbol{\phi}$, respectively. Thus, for the redundant case $n > m$, it follows that

$$\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{F} + \boldsymbol{N}^T\boldsymbol{\tau}, \tag{2.51}$$

where $\boldsymbol{\tau}_{nsp} = \boldsymbol{N}^T\boldsymbol{\tau}$ are nullspace torques that do not affect the end-effector motion. Applying the nullspace projector (2.50) and the pseudoinverse $\boldsymbol{J}^{\#T}$ to (2.51) yields $\boldsymbol{F} = \boldsymbol{J}^{\#T}\boldsymbol{\tau}$.

[1] For the derivation of the nullspace projector $\boldsymbol{N}$, recall that $\boldsymbol{J}\boldsymbol{J}^{\#} = \boldsymbol{E}$, but $\boldsymbol{J}^{\#}\boldsymbol{J} \neq \boldsymbol{E}$, and similarly for $\boldsymbol{N}^T$ that $\boldsymbol{J}^{\#T}\boldsymbol{J}^T = \boldsymbol{E}$, but $\boldsymbol{J}^T\boldsymbol{J}^{\#T} \neq \boldsymbol{E}$.

The dynamics in the task space coordinates $\boldsymbol{x} \in \mathbb{R}^m$ follow from the equivalence of the kinetic energy in configuration and in task space. Rearranging (2.42) and inserting in (2.47) yields

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{\phi})\boldsymbol{M}^{-1}(\boldsymbol{\phi})\left(\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{\phi},\dot{\boldsymbol{\phi}})\dot{\boldsymbol{\phi}} - \boldsymbol{g}(\boldsymbol{\phi})\right) + \dot{\boldsymbol{J}}(\boldsymbol{\phi},\dot{\boldsymbol{\phi}})\dot{\boldsymbol{\phi}}. \tag{2.52}$$

The task space dynamics can be written in the form

$$\boldsymbol{\Lambda}(\boldsymbol{\phi})\ddot{\boldsymbol{x}} + \boldsymbol{\mu}(\boldsymbol{\phi},\dot{\boldsymbol{\phi}}) + \boldsymbol{F}_g(\boldsymbol{\phi}) = \boldsymbol{F}_\tau, \tag{2.53}$$

where $\boldsymbol{\Lambda}(\boldsymbol{\phi})$ is referred to as pseudo kinetic energy matrix [40] and $\boldsymbol{\mu}(\boldsymbol{\phi},\dot{\boldsymbol{\phi}})$ is a vector of Coriolis and centrifugal terms[2].

Proposition 2.1. *Dropping arguments, the inertia matrix, Coriolis/centrifugal vector, the gravity terms, and the control input in task space are given by*

$$\boldsymbol{\Lambda} = \left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\right)^{-1}, \tag{2.54}$$

$$\boldsymbol{\mu} = \left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\right)^{-1}\left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{C}\dot{\boldsymbol{\phi}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{\phi}}\right), \tag{2.55}$$

$$\boldsymbol{F}_g = \boldsymbol{J}^{\#T}\boldsymbol{g}, \tag{2.56}$$

$$\boldsymbol{F}_\tau = \boldsymbol{J}^{\#T}\boldsymbol{\tau}, \tag{2.57}$$

with the generalized pseudoinverse of the transposed Jacobian matrix defined as

$$\boldsymbol{J}^{\#T} = \left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\right)^{-1}\boldsymbol{J}\boldsymbol{M}^{-1}. \tag{2.58}$$

Proof. Inserting (2.54) - (2.58) in (2.53) yields (2.52). □

Equations (2.54) - (2.58) are directly applicable to redundant systems. In the non-redundant case, it follows that $\boldsymbol{J}^{\#T} = \boldsymbol{J}^{-T}$ and, hence,

$$\boldsymbol{\mu} = \boldsymbol{J}^{-T}\left(\boldsymbol{C} - \boldsymbol{M}\boldsymbol{J}^{-1}\dot{\boldsymbol{J}}\right)\boldsymbol{J}^{-1}\dot{\boldsymbol{x}}. \tag{2.59}$$

The structure of the configuration space dynamics (2.42) and of the task space dynamics (2.53) is used to describe serial dynamical systems throughout this thesis.

2.4.5 Decomposition of first principles two body model

The problem of a robot manipulator interacting with a flying robot can be modeled specifically for a given manipulator and UAV. This model will contain the exact geometry and inertia of both systems and may be used to design a model-based controller especially designed for the underlying systems. However, in order to study a model decomposition based on the inertia of the subsystems, an analogous model consisting of two rigid bodies, one representing the manipulator and one representing the UAV, is discussed first.

Consider a system composed of two bodies as depicted in Figure 2.12. It does not contain linear (or prismatic) joints and each body has three relative degrees of freedom represented by $\boldsymbol{\omega}_{ri}$. The motion of the origin of the first reference frame (i.e. of the manipulators floating base b) w.r.t. the inertial frame i is given by $\boldsymbol{v}_{ib}$ and $\boldsymbol{\omega}_{ib}$ (measured in the base frame b).

[2] Note that $\boldsymbol{\mu}$ is defined different than in [37] on p. 32 and, therefore, $\dot{\boldsymbol{\Lambda}} \neq \boldsymbol{\mu} + \boldsymbol{\mu}^T$ (cf. [37], p. 33).

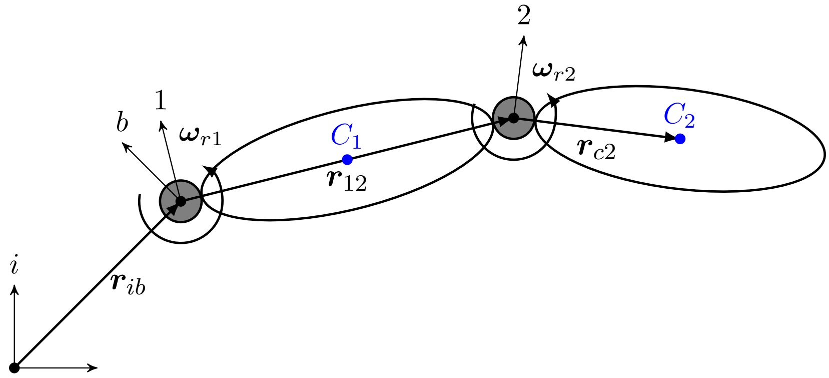


Figure 2.12: First principles two body model, e.g. of robot manipulator and flying robot.

Derivation of configuration space dynamics

The projection of the dynamics of the two rigid bodies is carried out using the upper triangular Jacobian (2.35) introduced in Section 2.4.2

$$\boldsymbol{J}_{tri}^T = \begin{bmatrix} \boldsymbol{J}_1^T & \boldsymbol{J}_1^T \boldsymbol{T}_{2,1}^T \\ \boldsymbol{0}_{3\times 9} & \boldsymbol{J}_2^T \end{bmatrix}_{12\times 18} . \tag{2.60}$$

The minimal representation is obtained using (2.39)

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{J}_1^T(\overline{\boldsymbol{M}}_1 + \boldsymbol{T}_{2,1}^T \overline{\boldsymbol{M}}_2 \boldsymbol{T}_{2,1})\boldsymbol{J}_1 & \boldsymbol{J}_1^T \boldsymbol{T}_{2,1}^T \overline{\boldsymbol{M}}_2 \boldsymbol{J}_2 \\ \boldsymbol{J}_2 \overline{\boldsymbol{M}}_2 \boldsymbol{T}_{2,1} \boldsymbol{J}_1 & \boldsymbol{J}_2^T \overline{\boldsymbol{M}}_2 \boldsymbol{J}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}_{11} & \boldsymbol{M}_{12} \\ \boldsymbol{M}_{12}^T & \boldsymbol{M}_{22} \end{bmatrix} , \tag{2.61}$$

and

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{J}_1^T(\overline{\boldsymbol{C}}_1 + \boldsymbol{T}_{2,1}^T \overline{\boldsymbol{C}}_2 \boldsymbol{T}_{2,1})\boldsymbol{J}_1 & \boldsymbol{J}_1^T \boldsymbol{T}_{2,1}^T \overline{\boldsymbol{C}}_2 \boldsymbol{J}_2 \\ \boldsymbol{J}_2 \overline{\boldsymbol{C}}_2 \boldsymbol{T}_{2,1} \boldsymbol{J}_1 & \boldsymbol{J}_2^T \overline{\boldsymbol{C}}_2 \boldsymbol{J}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}_1^T(\overline{\boldsymbol{C}}_1 + \boldsymbol{T}_{2,1}^T \overline{\boldsymbol{C}}_2 \boldsymbol{T}_{2,1})\dot{\boldsymbol{J}}_1 & \boldsymbol{J}_1^T \boldsymbol{T}_{2,1}^T \overline{\boldsymbol{C}}_2 \dot{\boldsymbol{J}}_2 \\ \boldsymbol{J}_2 \overline{\boldsymbol{C}}_2 \boldsymbol{T}_{2,1} \dot{\boldsymbol{J}}_1 & \boldsymbol{J}_2^T \overline{\boldsymbol{C}}_2 \dot{\boldsymbol{J}}_2 \end{bmatrix} , \tag{2.62}$$

with $\dot{\boldsymbol{s}} = \begin{pmatrix} \dot{\boldsymbol{s}}_1^T & \dot{\boldsymbol{s}}_2^T \end{pmatrix}^T = \begin{pmatrix} \boldsymbol{v}_{ib}^T & \boldsymbol{\omega}_{ib}^T & \boldsymbol{\omega}_{r1}^T & \boldsymbol{\omega}_{r2}^T \end{pmatrix}^T$, i.e. $n = 12$ DoF in configuration space.

Some important properties are summarized below, which follow from the construction of the serial chain dynamical system.

Important properties of serial chain dynamical systems

1. The elements of the projected inertia and Coriolis matrices $\boldsymbol{M}$ and $\boldsymbol{C}$ are linear in the inertia matrices of the subsystems (cf. (2.61) and (2.62), respectively).

2. The elements of inertia matrix $\boldsymbol{M}$ and Coriolis matrix $\boldsymbol{C}$ do not contain the inertia of the predecessing bodies in the chain. On the other hand, the elements of inertia and Coriolis matrix depend on the inertia of all its successors.

3. If, as shown, relative velocities are used and the last element in the chain is a single rigid body (which is always the case here), then the lower right 3×3 elements in inertia and Coriolis matrix are simply $\boldsymbol{M}_{22} = \boldsymbol{I}_2^c$ and $\boldsymbol{C}_{22} = \boldsymbol{S}(\boldsymbol{\omega}_{i2})\boldsymbol{I}_2^c + \dot{\boldsymbol{I}}_2^c$.

4. Neither the inertia nor the Coriolis/centrifugal matrix depend on the orientation of base b w.r.t. the inertial frame i.

Property 3 matches the observations by Khatib in [40] on page 27 for the kinetic energy matrix of macro/mini structures.

Model decomposition in configuration space

Taking advantage of the linearity of $\boldsymbol{M}$ and $\boldsymbol{C}$ in $\overline{\boldsymbol{M}}_1$, $\overline{\boldsymbol{M}}_2$ and $\overline{\boldsymbol{C}}_1$, $\overline{\boldsymbol{C}}_2$, respectively, inertia and Coriolis matrices are decomposed such that

$$\boldsymbol{M} = \boldsymbol{M}_1(\overline{\boldsymbol{M}}_1) + \boldsymbol{M}_2(\overline{\boldsymbol{M}}_2) = \begin{bmatrix} \boldsymbol{J}_1^T \overline{\boldsymbol{M}}_1 \boldsymbol{J}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}_1^T \boldsymbol{T}_{2,1}^T \overline{\boldsymbol{M}}_2 \boldsymbol{T}_{2,1} \boldsymbol{J}_1 & \boldsymbol{J}_1^T \boldsymbol{T}_{2,1}^T \overline{\boldsymbol{M}}_2 \boldsymbol{J}_2 \\ \boldsymbol{J}_2 \overline{\boldsymbol{M}}_2 \boldsymbol{T}_{2,1} \boldsymbol{J}_1 & \boldsymbol{J}_2^T \overline{\boldsymbol{M}}_2 \boldsymbol{J}_2 \end{bmatrix} \tag{2.63}$$

and $\boldsymbol{C} = \boldsymbol{C}_1(\overline{\boldsymbol{C}}_1) + \boldsymbol{C}_2(\overline{\boldsymbol{C}}_2)$ follows analogously. The decomposed dynamics read

$$\boldsymbol{M}_1 \ddot{\boldsymbol{s}} + \boldsymbol{C}_1 \dot{\boldsymbol{s}} + \boldsymbol{g}_1 + \boldsymbol{M}_2 \ddot{\boldsymbol{s}} + \boldsymbol{C}_2 \dot{\boldsymbol{s}} + \boldsymbol{g}_2 = \boldsymbol{\tau}. \tag{2.64}$$

Obviously, now a control law $\boldsymbol{\tau} = \boldsymbol{\tau}_1(\boldsymbol{M}_1, \boldsymbol{C}_1, \boldsymbol{g}_1) + \boldsymbol{\tau}_2(\boldsymbol{M}_2, \boldsymbol{C}_2, \boldsymbol{g}_2)$ could be designed that individually takes the properties of subsystems one and two into account. However, $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ have to be designed, such that the interconnected two body model reaches a stable equilibrium.

Model decomposition in task space

Using the Jacobian (2.60), the dynamics can be transformed to task space using the method described in Section 2.4.4. Recall the definition of the inertia matrix in task space

$$\boldsymbol{\Lambda} = \left(\boldsymbol{J} \boldsymbol{M}^{-1} \boldsymbol{J}^T \right)^{-1}. \tag{2.65}$$

To identify the contribution of the subsystems, $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$ are sought-after, such that

$$\boldsymbol{\Lambda}_1 + \boldsymbol{\Lambda}_2 = \left(\boldsymbol{J} \left(\boldsymbol{M}_1 + \boldsymbol{M}_2 \right)^{-1} \boldsymbol{J}^T \right)^{-1}. \tag{2.66}$$

To solve equation (2.66) for $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$ individually, the following Lemma from [41] is applied.

Lemma 2.1. *For the inverse of the sum of two matrices it holds that*

$$(\boldsymbol{M}_1 + \boldsymbol{M}_2)^{-1} = \boldsymbol{M}_2^{-1} - \boldsymbol{M}_2^{-1} \boldsymbol{M}_1 (\boldsymbol{M}_1 + \boldsymbol{M}_2)^{-1}. \tag{2.67}$$

Proof. Multiplying $(\boldsymbol{M}_1 + \boldsymbol{M}_2)$ from the right yields

$$\boldsymbol{E} = \boldsymbol{M}_2^{-1} (\boldsymbol{M}_1 + \boldsymbol{M}_2) - \boldsymbol{M}_2^{-1} \boldsymbol{M}_1 = \boldsymbol{M}_2^{-1} \boldsymbol{M}_1 + \boldsymbol{E} - \boldsymbol{M}_2^{-1} \boldsymbol{M}_1 = \boldsymbol{E}.$$

Multiplying $(\boldsymbol{M}_1 + \boldsymbol{M}_2)$ from the left leads to the same result. This proves the lemma. □

Applying Lemma 2.1 to (2.66) yields

$$\left(\boldsymbol{J}\left(\boldsymbol{M}_1+\boldsymbol{M}_2\right)^{-1}\boldsymbol{J}^T\right)^{-1}=\left(\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{J}^T-\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{M}_1(\boldsymbol{M}_1+\boldsymbol{M}_2)^{-1}\boldsymbol{J}^T\right)^{-1}, \tag{2.68}$$

to which Lemma 2.1 can be applied once more, resulting in

$$\boldsymbol{\Lambda}_1(\boldsymbol{M}_1,\boldsymbol{M}_2)=(\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{J}^T)^{-1}\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{M}_1(\boldsymbol{M}_1+\boldsymbol{M}_2)^{-1}\boldsymbol{J}^T\left(\boldsymbol{J}\left(\boldsymbol{M}_1+\boldsymbol{M}_2\right)^{-1}\boldsymbol{J}^T\right)^{-1}, \tag{2.69}$$

$$\boldsymbol{\Lambda}_2(\boldsymbol{M}_2)=(\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{J}^T)^{-1}. \tag{2.70}$$

Equations (2.69) and (2.70) reveal that $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$ can not be completely separated in the redundant case. However, the portion of the last element in the chain can directly be deduced. Here, it is given by subsystem two as $\boldsymbol{\Lambda}_2(\boldsymbol{M}_2)=(\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{J}^T)^{-1}$. The first element in the chain $\boldsymbol{\Lambda}_1(\boldsymbol{M}_1,\boldsymbol{M}_2)$ depends on the inertias $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ as well as on the configuration of the system defined by $\boldsymbol{J}$. However, for non-redundant systems complete decomposition can be accomplished as shown below.

Proposition 2.2. *In the non-redundant and fully actuated case, i.e. with $n=m$, the solution (2.69), (2.70) reduces to the complete and trivial decomposition*

$$\boldsymbol{\Lambda}_1+\boldsymbol{\Lambda}_2=\boldsymbol{J}^{-T}\boldsymbol{M}_1\boldsymbol{J}^{-1}+\boldsymbol{J}^{-T}\boldsymbol{M}_2\boldsymbol{J}^{-1}. \tag{2.71}$$

Proof. The proof directly follows from the invertibility of $\boldsymbol{J}$ in the non-redundant case, which applied to (2.69) results in

$$\boldsymbol{\Lambda}_1(\boldsymbol{M}_1,\boldsymbol{M}_2)=\boldsymbol{J}^{-T}\boldsymbol{M}_2\boldsymbol{J}^{-1}\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{M}_1(\boldsymbol{M}_1+\boldsymbol{M}_2)^{-1}\boldsymbol{J}^T\boldsymbol{J}^{-T}(\boldsymbol{M}_1+\boldsymbol{M}_2)\boldsymbol{J}^{-1} \tag{2.72}$$

$$=\boldsymbol{J}^{-T}\boldsymbol{M}_1\boldsymbol{J}^{-1}, \tag{2.73}$$

which is the kinetic energy matrix of subsystem one. □

Next, the decomposition of the Coriolis/centrifugal vector is treated. Recall the definition

$$\boldsymbol{\mu}=\boldsymbol{\Lambda}(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{C}-\dot{\boldsymbol{J}})\dot{\boldsymbol{q}}, \tag{2.74}$$

which should be solved for $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, such that

$$\boldsymbol{\mu}_1+\boldsymbol{\mu}_2=(\boldsymbol{\Lambda}_1+\boldsymbol{\Lambda}_2)\left(\boldsymbol{J}(\boldsymbol{M}_1+\boldsymbol{M}_2)^{-1}(\boldsymbol{C}_1+\boldsymbol{C}_2)-\dot{\boldsymbol{J}}\right)\dot{\boldsymbol{q}}. \tag{2.75}$$

Again, applying (2.67) and rearranging leads to

$$\begin{aligned}\boldsymbol{\mu}_1(\boldsymbol{M}_1,\boldsymbol{M}_2)&=\boldsymbol{\Lambda}_1\left(\boldsymbol{J}(\boldsymbol{M}_1+\boldsymbol{M}_2)^{-1}(\boldsymbol{C}_1+\boldsymbol{C}_2)-\dot{\boldsymbol{J}}\right)\dot{\boldsymbol{q}}+\boldsymbol{\Lambda}_2(\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{C}_1-\dot{\boldsymbol{J}})\dot{\boldsymbol{q}}\\&\quad-\boldsymbol{\Lambda}_2(\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{M}_1(\boldsymbol{M}_1+\boldsymbol{M}_2)^{-1}(\boldsymbol{C}_1+\boldsymbol{C}_2)-\dot{\boldsymbol{J}})\dot{\boldsymbol{q}},\\\boldsymbol{\mu}_2(\boldsymbol{M}_2)&=\boldsymbol{\Lambda}_2(\boldsymbol{J}\boldsymbol{M}_2^{-1}\boldsymbol{C}_2-\dot{\boldsymbol{J}})\dot{\boldsymbol{q}}.\end{aligned} \tag{2.76}$$

The Coriolis/centrifugal vector $\boldsymbol{\mu}_2$ of subsystem two can be identified directly, but the Coriolis/centrifugal vector $\boldsymbol{\mu}_1$ of subsystem one does only simplify in the non-redundant case.

Proposition 2.3. *In the non-redundant and fully actuated case $n = m$, equation (2.76) reduces to the complete and trivial decomposition*

$$\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2 = \boldsymbol{J}^{-T}(\boldsymbol{C}_1 - \boldsymbol{M}_1\boldsymbol{J}^{-1}\dot{\boldsymbol{J}})\dot{\boldsymbol{q}} + \boldsymbol{J}^{-T}(\boldsymbol{C}_2 - \boldsymbol{M}_2\boldsymbol{J}^{-1}\dot{\boldsymbol{J}})\dot{\boldsymbol{q}}. \tag{2.77}$$

Proof. Inserting the relation $\boldsymbol{\Lambda} = \boldsymbol{J}^{-T}\boldsymbol{M}\boldsymbol{J}^{-1}$ in (2.74) yields

$$\boldsymbol{\mu} = \boldsymbol{J}^{-T}(\boldsymbol{C} - \boldsymbol{M}\boldsymbol{J}^{-1}\dot{\boldsymbol{J}})\dot{\boldsymbol{q}}, \tag{2.78}$$

which is linear in $\boldsymbol{M}$ and $\boldsymbol{C}$ and therefore the decomposition is found as

$$\boldsymbol{\mu} = \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2 = \boldsymbol{J}^{-T}\left((\boldsymbol{C}_1 + \boldsymbol{C}_2) - (\boldsymbol{M}_1 + \boldsymbol{M}_2)\boldsymbol{J}^{-1}\dot{\boldsymbol{J}}\right)\dot{\boldsymbol{q}}. \tag{2.79}$$

Rearranging the above equation results in (2.77). □

Equations (2.70) and (2.76) lead to a partial decomposition of the task space dynamics of redundant systems

$$\boldsymbol{\Lambda}\ddot{\boldsymbol{x}}+\boldsymbol{\mu}+\boldsymbol{F}_g = \boldsymbol{\Lambda}_1(\boldsymbol{M}_1,\boldsymbol{M}_2)\ddot{\boldsymbol{x}}+\boldsymbol{\Lambda}_2(\boldsymbol{M}_2)\ddot{\boldsymbol{x}}+\boldsymbol{\mu}_1(\boldsymbol{M}_1,\boldsymbol{M}_2,\boldsymbol{C}_1,\boldsymbol{C}_2)+\boldsymbol{\mu}_2(\boldsymbol{M}_2,\boldsymbol{C}_2)+\boldsymbol{F}_g. \tag{2.80}$$

The terms $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\mu}_1$ contain the influence of subsystem two on the dynamics of subsystem one. As can be seen from (2.69) and (2.70), $\boldsymbol{\Lambda}_1$ scales with $\boldsymbol{M}_2$. Moreover, equation (2.76) shows, that the contribution of the Coriolis and centrifugal forces of system one is reduced indirectly proportional to the diagonal elements of the product $\boldsymbol{M}_1\boldsymbol{M}_2^{-1}$ (or directly proportional to the diagonal elements of the product $\boldsymbol{M}_1^{-1}\boldsymbol{M}_2$). These insights are intuitive, because they simply show that a heavy subsystem contributes more to the overall dynamics than a light subsystem.

The presented decompositions can be used to design controllers that take advantage of the underlying dynamic structure of the robotic systems. For example, the derived inertia and Coriolis/centrifugal matrices in task space allow to unilaterally compensate the coupling between subsystem one and two. Furthermore, they allow to derive design principles for robotic assistance systems. Most notably, non-redundant and fully actuated systems allow a simpler modeling and control approach. However, redundant designs may result in better performance, for example in terms of dexterity and workspace coverage. They require either complete model knowledge or have to be robust against model uncertainties. Models and suitable control laws for both the redundant and the non-redundant case are presented in Chapter 4 and Chapter 5.

2.5 State-of-the-art control of redundant and distributed systems

As explained above, in general the combined system is redundant meaning that it has more DoF than are actually necessary to fulfill the task. Both flying robot and robot manipulator use their own control computer resulting in a distributed system. Coordinated control of the system requires a communication channel. The latter is subject to time delay which needs to be considered in the controller design and in the stability analysis.

Robot-assisted take-off and landing of flying robots has not been considered in the literature so far. Nevertheless, state-of-the-art approaches exist for coordinated and distributed control under time delay, for redundancy resolution, and for physical interaction control.

Coordinated and distributed control

Research considering coordinated control of a flying robot and a robot manipulator is mostly devoted to aerial manipulation[3] [5, 42, 43, 44, 45], where the manipulator is attached to the aerial vehicle. This poses a challenge due to the dynamical coupling between the two systems. Control of a flying robot equipped with a Light Weight Robot manipulator with seven DoF is addressed in [46] (regulation case) and [47] (tracking case). Dynamics and nonlinear control of a system composed of a quadrocopter and a manipulator (with multiple DoF) are presented in [48, 49, 50].

Visually coordinated landing or docking of a flying robot on a mobile robot manipulator is considered in [51] and [52], respectively, but only evaluated in simulation. Vision-based autonomous landing of flying robots on a moving base is presented in [53] and [54]. Modeling and control of manipulators on large moving platforms, such as ships, is treated in [36, 55, 56]. A recent adaptive control approach for networked cooperative mobile manipulators can be found in [57].

Consideration of time delay

In the literature, time delay is mostly considered in telemanipulation, where a human operator controls a robot in a remote environment. Passivity, and therefore stability, may be preserved using time domain passivity control (TDPC) initially introduced in [58] and generalized to multi-DoF haptic devices in [59] and to time-varying communication delay in [60]. TDPC adjusts a variable damping such that the energy generated by the time delay is exactly dissipated. The input and output energies are constantly monitored using a passivity observer [61]. An alternative approach is to use so-called wave variables [62], which are applied to coordinated control of robot manipulators with distributed control architecture in [63]. The survey [64] provides an overview of the wave variable approach for telemanipulation with (constant) time delays. A general treatment of stability analysis and robust control of systems under time delay can be found in [65, 66]. A stability analysis considering time delay for the linear dynamics model introduced in Section 4.2 is presented in Appendix A9.

[3] Note that aerial or mobile manipulation are fundamentally different to robot-assisted take-off and landing of flying robots. The aim of the latter is to move the flying robot along a desired trajectory by means of the manipulator. Both systems may contribute to fulfill the task. In contrast to that, the aim of aerial or mobile manipulation is to apply a desired wrench to the environment. There, the flying or mobile robot has to compensate the counterforce arising from the interaction of the manipulator with the environment.

Time delay also affects the state estimation [67]. For instance, the delay may be directly included in a Kalman filter, as done in [68, 69]. On the other hand, if the time delay of the communication channel is lower than the update rate of the Kalman filter, it may be neglected. However, due to varying time delay some measurements may arrive too late. They would corrupt the estimate and need to be rejected. This is considered in Chapter 6, where in-flight capturing of the flying robot by means of the robot manipulator is addressed.

Redundancy resolution

Redundancy occurs for robotic systems where the joint space is larger than the task space. In [70], redundancy is exploited for impedance control of an aerial manipulator system. A control approach for a seven DoF manipulator is presented in [71]. The approach is based on potential fields and keeps the elbow of the manipulator between the landing skids of an unmanned helicopter used for aerial manipulation. This approach is also implemented for the experiments presented in Chapter 4.

Other examples are mobile manipulators or legged and wheeled humanoids [72]. The redundancy also enables to fulfill secondary objectives in addition to the main task. Multiple objectives may be formulated hierarchically as done in [73]. In [74], passivation of the hierarchy is achieved via virtual energy tanks. A secondary objective is for example the generation of a desired nullspace behaviour [75, 76]. The redundancy can also be resolved through torque optimization [77]. An optimization allows to directly consider actuator constraints. In Chapter 5, distribution of the control effort between flying robot and robot manipulator is realized via a heuristic or by solving a quadratic optimization problem.

Physical interaction control

Physical interaction inevitably occurs in robot-assisted take-off and landing of flying robots. Thus, another relevant line of research is interaction control of robotic systems. Commonly, interaction control is realized using direct force control ([78], p. 162), which requires to measure the contact force, or indirect force control, such as impedance control ([78], p. 167). The latter is based on the seminal work [79] and has since then found wide acceptance in the scientific community. In impedance control, the robot appears to the environment as a mass-spring-damper system for which passivity and, therefore stability, can be shown ([37], p. 104).

In [19, 80], a unified framework for external wrench estimation and interaction control of flying robots under wind influence is presented. The wrench is estimated using a generalized momentum observer [81, 82] and included in the general impedance controller ([78], p. 167). An alternative approach can be found in [83], where a nonlinear Lyapunov-based disturbance observer for estimating the external wrench is presented and the flying robot is controlled via interconnection and damping assignment passivity-based control (IDA-PBC). Physical interaction of a human with a flying quadrocopter by means of admittance control is treated in [84], where external forces and torques are estimated by an Unscented Kalman Filter (UKF) [85].

2.6 Generalized task space control in a nutshell

Applying state transformations and feedback control to a nonlinear dynamical system in order to obtain a linear system, is commonly referred to as feedback linearization [86, 87], inverse dynamics control ([88], pp. 269), or nonlinear decoupling [89]. Feedback linearizing the task space dynamics enables to utilize a whole set of linear and nonlinear control methods. It is assumed that the joint torques $\boldsymbol{\tau}$ are the input to the robotic system, which is reasonable for a torque-controlled robot like the LWR manipulator presented in Section 2.2. An overview of the considered control scheme is depicted in Figure 2.13. Specifics about the low-level torque controller of the LWR can be found in [22]. The task space controllers introduced in the following can be transferred to position-controlled robot manipulators by using an inner position and orientation control loop. This results in admittance control ([78], p. 170).

Proposition 2.4. *Applying feedback linearization to the task space dynamics (2.53), transforms (2.53) to a single double integrator system*

$$\ddot{\boldsymbol{x}} = \boldsymbol{u}. \tag{2.81}$$

Proof. Introducing the virtual control $\boldsymbol{u}$ and inserting the control law

$$\boldsymbol{F}_\tau = \boldsymbol{\Lambda}\boldsymbol{u} + \boldsymbol{\mu}_d + \boldsymbol{F}_g, \tag{2.82}$$

with $\boldsymbol{\mu}_d = \boldsymbol{\mu}$ in the dynamical system (2.53), yields (2.81). □

The linear and angular accelerations $\ddot{\boldsymbol{x}} \in \mathbb{R}^m$ are now decoupled and can be controlled independently. Possible redundancy of the robotic system needs to be resolved via the nullspace torque $\boldsymbol{\tau}_{nsp}$, e.g. using joint damping, keeping the manipulator close to a desired configuration, or using an elbow field [71]. The resulting control torque is

$$\boldsymbol{\tau} = \boldsymbol{J}^T(\boldsymbol{\Lambda}\boldsymbol{u} + \boldsymbol{\mu}_d) + \boldsymbol{\tau}_{nsp} + \boldsymbol{g}. \qquad \textit{(feedback linearization)} \tag{2.83}$$

Next, different choices of the virtual control input $\boldsymbol{u}$ are discussed. A straight-forward control law to track a desired pose $\boldsymbol{x}_d$ with $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_d$ is given by

$$\boldsymbol{u} = \ddot{\boldsymbol{x}}_d - \boldsymbol{D}\dot{\tilde{\boldsymbol{x}}} - \boldsymbol{K}\tilde{\boldsymbol{x}}, \tag{2.84}$$

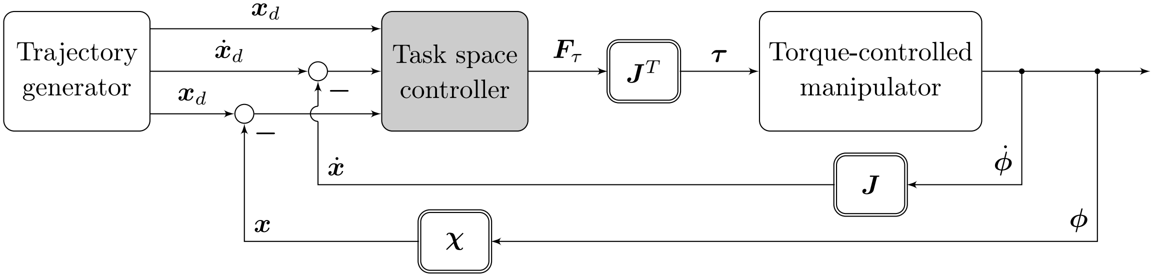


Figure 2.13: Task space control scheme considered in this work.

with positive definite damping matrix $\boldsymbol{D}$ and positive definite stiffness matrix $\boldsymbol{K}$. This results in the so-called inverse dynamics controller

$$\boldsymbol{\tau} = \boldsymbol{J}^T \left(\boldsymbol{\Lambda}(\ddot{\boldsymbol{x}}_d - \boldsymbol{D}\dot{\tilde{\boldsymbol{x}}} - \boldsymbol{K}\tilde{\boldsymbol{x}}) + \boldsymbol{\mu}_d \right) + \boldsymbol{\tau}_{nsp} + \boldsymbol{g}. \qquad \textit{(inverse dynamics control)} \tag{2.85}$$

In terms of stability and robustness, the closed-loop dynamics under inverse dynamics control are sensitive to dynamic uncertainties, such as viscous friction or joint damping ([24], p. 332). Also, a requirement for stability is that $\boldsymbol{\Lambda D}$ and $\boldsymbol{\Lambda K}$ in (2.85) are positive definite, which cannot be guaranteed, because the product of two positive definite matrices is not positive definite in general [90]. Hence, a slightly different control law $\boldsymbol{u}$ can be found with

$$\boldsymbol{u} = \ddot{\boldsymbol{x}}_d - \boldsymbol{\Lambda}^{-1}(\boldsymbol{D}\dot{\tilde{\boldsymbol{x}}} + \boldsymbol{K}\tilde{\boldsymbol{x}}). \tag{2.86}$$

This leads to classical task space impedance control ([37], p. 36)[4] without inertia shaping

$$\boldsymbol{\tau} = \boldsymbol{J}^T \left(\boldsymbol{\Lambda}\ddot{\boldsymbol{x}}_d - \boldsymbol{D}\dot{\tilde{\boldsymbol{x}}} - \boldsymbol{K}\tilde{\boldsymbol{x}} + \boldsymbol{\mu}_d \right) + \boldsymbol{\tau}_{nsp} + \boldsymbol{g}. \qquad \textit{(impedance control)} \tag{2.87}$$

The term impedance control stems from the electrical and mechanical analogon. A mechanical impedance takes position and velocity and generates a force, as opposed to an admittance, which takes a force and results in a velocity and change in position. In his seminal work [79], Hogan states that for safety reasons a robotic system in contact with the environment (admittance) should always appear as an impedance.

Note that for stability and passivity (in the redundant case), the desired Coriolis/centrifugal vector needs to be $\boldsymbol{\mu}_d = \boldsymbol{\mu} - \frac{1}{2}\dot{\boldsymbol{\Lambda}}\dot{\tilde{\boldsymbol{x}}}$. This can be shown using the (time-varying) Lyapunov function ([37], p. 35)

$$V = \frac{1}{2}\left(\dot{\tilde{\boldsymbol{x}}}^T \boldsymbol{\Lambda} \dot{\tilde{\boldsymbol{x}}} + \tilde{\boldsymbol{x}}^T \boldsymbol{K} \tilde{\boldsymbol{x}} \right), \tag{2.88}$$

whose derivative w.r.t. time follows with (2.87) as

$$\dot{V} = -\dot{\tilde{\boldsymbol{x}}}^T \boldsymbol{D} \dot{\tilde{\boldsymbol{x}}} + \dot{\tilde{\boldsymbol{x}}}^T \left(\boldsymbol{\mu}_d - \boldsymbol{\mu} + \frac{1}{2}\dot{\boldsymbol{\Lambda}}\dot{\tilde{\boldsymbol{x}}} \right). \tag{2.89}$$

In the non-redundant case, the Coriolis/centrifugal vector can be (cf. (2.59) in Section 2.4.4)

$$\boldsymbol{\mu}_d = \boldsymbol{J}^{-T} \left(\boldsymbol{C} - \boldsymbol{M}\boldsymbol{J}^{-1}\dot{\boldsymbol{J}} \right) \boldsymbol{J}^{-1}\dot{\boldsymbol{x}}_d, \tag{2.90}$$

which is shown in [37] on page 34 - 36.

An inherent problem of all task space control approaches, is that the Jacobian matrix can become singular, e.g. at a singular configuration of the robotic system. Therefore, kinematic singularities have to be avoided by the trajectory generator or motion planner. Furthermore, so far external forces are not explicitly considered in the controller design, but implicitly treated as unknown disturbances. If an external force acts on the robot, the dynamics (2.53) become

$$\boldsymbol{\Lambda}\ddot{\boldsymbol{x}} + \boldsymbol{\mu} + \boldsymbol{F}_g = \boldsymbol{F}_\tau + \boldsymbol{F}_{ext} \tag{2.91}$$

[4] The formulation of the impedance controller is different to [37], because here $\boldsymbol{\mu}_d$ is a vector instead of a matrix and it can either contain the velocity error (redundant case) or the desired velocity (non-redundant case).

or under the feedback linearizing control (2.85) introduced above

$$\boldsymbol{\Lambda}(\ddot{\boldsymbol{x}} - \boldsymbol{u}) = \boldsymbol{F}_{ext}. \tag{2.92}$$

Considering the aim of this thesis, a robot manipulator that physically interacts with a flying robot will perceive its real inertia. That might be disadvantageous, because of large contact forces, especially for heavy UAVs. To address this fact, the inertia can be shaped via

$$\boldsymbol{F}_{\tau} = \boldsymbol{\Lambda u} + \boldsymbol{\mu}_d + \boldsymbol{F}_g + \left(\boldsymbol{\Lambda \Lambda}_d^{-1} - \boldsymbol{E}\right) \boldsymbol{F}_{ext}, \tag{2.93}$$

wherein $\boldsymbol{\Lambda}_d$ is the desired inertia and $\boldsymbol{F}_{ext}$ are the measured external forces. The inertia of the double integrator system (2.92) then becomes

$$\boldsymbol{\Lambda}_d(\ddot{\boldsymbol{x}} - \boldsymbol{u}) = \boldsymbol{F}_{ext}, \tag{2.94}$$

and the computed torque control law is obtained as

$$\boldsymbol{\tau} = \boldsymbol{J}^T \left(\boldsymbol{\Lambda u} + \boldsymbol{\mu}_d\right) + \boldsymbol{\tau}_{nsp} + \boldsymbol{g} + \boldsymbol{J}^T \left(\boldsymbol{\Lambda \Lambda}_d^{-1} - \boldsymbol{E}\right) \boldsymbol{F}_{ext}. \qquad \textit{(inertia shaping)} \tag{2.95}$$

Example: Inertia mimicking for flying robot fixed to robot manipulator

Assume that a flying robot consisting of a single rigid body is attached at the end-effector of a robot manipulator. The inertia of the flying robot (or UAV, index u) can be preserved ($m_d = m_u$, $\boldsymbol{I}_d = \boldsymbol{I}_u$) or shaped ($m_d < m_u$, $\boldsymbol{I}_d < \boldsymbol{I}_u$), such that

$$\boldsymbol{\Lambda}_d = \begin{bmatrix} m_d \boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_d \end{bmatrix}, \tag{2.96}$$

where m_d is the desired mass and $\boldsymbol{I}_d$ is the desired inertia. Inserting (2.96) in (2.94) gives

$$\begin{bmatrix} m_d \boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_d \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{v}} - \boldsymbol{u}_v \\ \dot{\boldsymbol{\omega}} - \boldsymbol{u}_\omega \end{pmatrix} = \boldsymbol{F}_{ext}. \tag{2.97}$$

Thus, applying the control law (2.95) with (2.96) to the combined system dynamics (2.53) results in decoupled translational and rotational dynamics equal to those of the flying robot, but without velocity couplings. Now, any suitable control laws can be applied via the virtual control inputs $\boldsymbol{u}_v$ and $\boldsymbol{u}_\omega$ without the need to account for the real inertia $\boldsymbol{\Lambda}$, but for the desired inertia $\boldsymbol{\Lambda}_d$.

"Once you have tasted flight, you will forever walk the earth with your eyes turned skyward, for there you have been, and there you will always long to return."
— Leonardo da Vinci

3

Modeling and control of flying robots

In this chapter, independent modeling and control of flying robots is presented. The aim is to highlight the crucial factors in flying robot control. Moreover, a control approach is proposed that enables the flying robot to accurately follow a desired flight path even in the presence of model uncertainties or external disturbances, such as wind or interaction forces. The latter arise for example in contact with the robotic support system during take-off and landing. Figure 3.1 shows a roadmap of this chapter.

Commonly, attitude and position controllers of flying robots only use a rigid body model (see Section 3.2). The nonlinear dynamics of the rotor (or propulsion) system are usually neglected and rotor thrust and torque are approximated using constant quadratic functions.

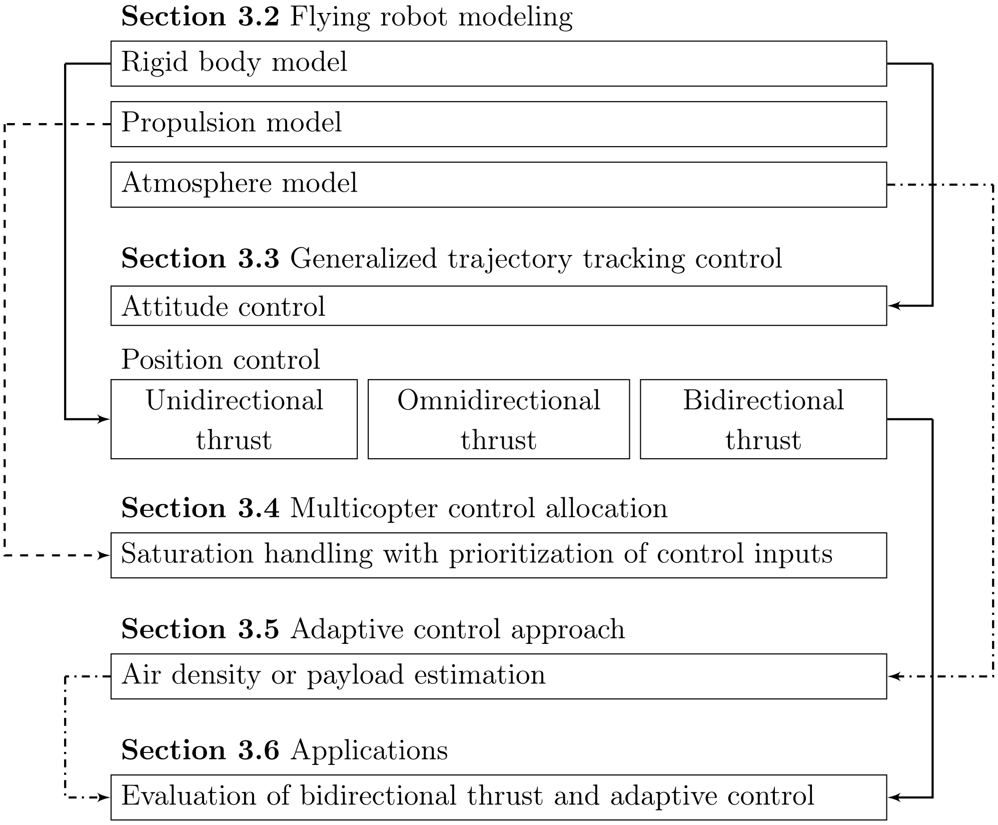

Section 3.2 Flying robot modeling
Rigid body model
Propulsion model
Atmosphere model
Section 3.3 Generalized trajectory tracking control
Attitude control
Position control
Unidirectional thrust
Omnidirectional thrust
Bidirectional thrust
Section 3.4 Multicopter control allocation
Saturation handling with prioritization of control inputs
Section 3.5 Adaptive control approach
Air density or payload estimation
Section 3.6 Applications
Evaluation of bidirectional thrust and adaptive control

Figure 3.1: Structural overview of Chapter 3.

However, rotor thrust and torque also linearly depend on the air density, which changes depending on the local atmosphere. This uncertainty affects control performance, since thrust and torque are not produced as expected, as well as the accuracy of disturbance observers, which assume that applied thrust and torque are known. To achieve increased performance, existing model-based control and external wrench estimation approaches are extended using a novel adaptive control approach in Section 3.5. It enables trajectory tracking, with formal passivity and convergence guaranty, and simultaneous air density estimation. Alternatively, it can be used for payload estimation, because the mass of the vehicle acts in a multiplicative way similar to the air density. In Section 3.3, position control is presented from a generic perspective, i.e. in addition to common aerial vehicles with unidirectional thrust vector, fully actuated vehicles (i.e. with omnidirectional thrust vector) are considered and bidirectional thrust is introduced.

In Section 3.4, the distribution of generalized control forces, e.g. computed by position and attitude controllers, to the actuators is treated. This is referred to as control allocation in the aerospace and marine literature ([91], [92], p. 398). If the number of actuators N is greater than the number of generalized control forces M, it is an overactuated control problem and for $N < M$ it is an underactuated control problem. Up to date, the control allocation for flying robots is almost always solved without considering constraints. However, neglecting the actuator limits can lead to actuator saturation and, in the worst case, to loss of control. Therefore, multiple control allocation methods that consider different propulsion models and actuator saturation are presented in Section 3.4.

On the considered flying robots at DLR, the rotor speeds can be measured. This is a matter of the practical implementation and usually not possible on conventional unmanned aerial vehicles. However, it has some major benefits. For instance, online rotor speed measurements can be used to detect rotor saturation or failure, to perform an online parameter identification (e.g. of the thrust and torque coefficients), and to implement closed-loop rotor speed control.

This chapter is based substantially on the author's own article [209], with the introduction of bidirectional thrust and incremental control allocation being based on the author's conference paper [215]. The remainder of this chapter is structured as follows. First, an overview of the state-of-the-art in the respective fields is given in Section 3.1. Then, the considered dynamical model of a flying robot and a parameter identification procedure are presented in Section 3.2. Based on the rigid body dynamics, a generalized trajectory tracking controller is introduced in Section 3.3 considering aerial vehicles with uni-, bi-, and omnidirectional thrust vectors. In Section 3.4, control allocation for flying robots with multiple rotors is presented and extended to polynomial propulsion models. Moreover, novel prioritized saturation handling methods are discussed. In order to handle parameter variations, the aforementioned adaptive controller is designed in Section 3.5. It specifically accounts for varying air density or payload and is used to augment a hybrid external wrench estimator used for disturbance rejection. Finally, the presented methods are evaluated in simulations and experiments in Section 3.6.

3.1 State-of-the-art flying robot control

In this section, state-of-the-art trajectory tracking, external wrench estimation, and saturation handling approaches for flying robots are reviewed briefly and summarized in Table 3.1, Table 3.2, and Table 3.3. Since flying robots are subject to wind, different environmental conditions, and interaction forces, the emphasis is on approaches that account for external disturbances as well as model uncertainties, such as a changing air density.

3.1.1 Trajectory tracking control

Many researchers have treated trajectory tracking control so far. A straightforward approach is to use Proportional Integral Derivative (PID) control [93], with linearization about hover [94] or feedback linearization [95]. The latter is sensitive to noise and modeling errors and in general the convergence and stability of the integral part are difficult to assure. Other approaches are the geometric controllers presented by Lee et al. [29] and by Mellinger et al. [96], the Nonlinear Dynamic Inversion (NDI) based controller presented by Achtelik et al. [97], and the integral sliding mode controller presented by Tomić et al. [98], wherein also a comparison to linear Proportional Derivative (PD), linear PID, and adaptive integral backstepping control [94] is given. In [94], a disturbance observer is used, which compensates for model uncertainties.

In most of the above approaches, model uncertainties and parameter variations are neglected, which lead to the development of adaptive controllers. Faessler et al. show in [99] that the dynamics of a quadrocopter are flat[1], also if first order (i.e. linear) drag effects are considered, and use the identified drag coefficients as feed-forward compensation at high velocities. Though, the quadrocopter system is not flat, as soon as higher order drag terms are considered[2]. Fernando et al. [101] treat adaptive attitude control of a quadrocopter with unknown inertia and achieve either asymptotic convergence of the attitude tracking error or boundedness and robustness against disturbances via a leakage term. Though, they neglect the influence of the air density on the propulsion wrench (i.e. rotor forces and torques) of the quadrocopter. Demircioglu et al. [102] present adaptive attitude and altitude control for a quadrocopter. They consider wind disturbances, but do not address the air density explicitly and only provide simulation results. Schreier et al. [103] propose the two classical adaptive control approaches Model Identification Adaptive Control (MIAC) and Model Reference Adaptive Control (MRAC) for position and attitude control of a quadrocopter. They consider a linearized quadrocopter model and a propulsion model with lumped constant thrust and torque coefficients, i.e. which contain a constant air density.

An adaptive estimation of dynamical system and propulsion model parameters in conjunction with Incremental Nonlinear Dynamic Inversion (INDI) is presented by Smeur et al.

[1] In control theory, flatness means that the control inputs may be written as functions of the so-called flat outputs (e.g. for flying robots the position and heading angle) and their derivatives. Flatness is used in practice to simplify the trajectory generation problem by considering only the kinematics of the rigid body [100].

[2] Please note that blade flapping [100] and drag models [99] are not considered in this work, because robot-assisted take-off and landing are assumed to be performed at moderate velocities.

Table 3.1: State-of-the-art trajectory tracking control of flying robots.

Reference	Method	Considered uncertainty / disturbances
[93, 94, 95]	PID	✗/ ✗
[29, 96]	Geometric, PD	✗/ ✗
[99]	PD	Rotor drag / ✗
[94]	Integral backstepping	✗/ Ground effect
[98]	Sliding mode	✗/ Acceleration-based observer
[101, 102, 103]	MIAC, MRAC	Plant model / Dryden gust model
[97]	NDI	✗/ ✗
[104, 105]	INDI	Plant model / External force observer
[106]	MPC	✗/ ✗
[107]	MPC	Perception system / ✗

[104]. This approach can be interpreted as a feedback linearization controller with a disturbance observer. INDI allows to reject disturbances directly on acceleration level. However, angular acceleration measurements can only be obtained numerically on a flying robot (hence, are noisy or delayed due to filtering) and the Least Mean Squares algorithm used for parameter identification provides no guarantee for convergence of the tracking error and does not yield the air density explicitly.

INDI is also used in combination with the differential flatness property of multicopters by Tal et al. [105] to realize accurate tracking of aggressive quadrocopter trajectories. They use a motion capture system for position estimation and neglect the influence of the air density on the propulsion wrench, i.e. it will appear like a disturbance. Interestingly, they derive an expression for the external forces similar to the external wrench estimators treated in Section 3.5.4. A more recent trend in flying robot control is to include awareness of the perception system [107] by defining a suitable cost function and using Model Predictive Control (MPC) [106].

3.1.2 External wrench and wind estimation

Flying robots are subject to disturbances, e.g. close to the ground or close to structures, and may be subject to wind disturbances if they operate outdoors. One of the first approaches considering the influence of wind on flying robot control is the method by Waslander et al. [108], who use a rigid body model and measured linear acceleration for wind estimation and rejection. More accurate wind estimation is achieved by Tomić et al. [109] via an aerodynamic power model and measured rotor speeds. General external wrench estimation for flying robots is studied for instance in [110], [111], and [112]. The estimator by Tomić et al. [112] is based on the original observer by de Luca et al. [81, 82]. In [112], it is assumed that contact and wind forces can be distinguished, as shown in [113], and that the commanded thrust and torques are exactly applied by the propulsion system (i.e. it is assumed that the air density is known). This is, due to the varying air density, not true per se. Thus, an additional online air density estimator is required.

Table 3.2: State-of-the-art external wrench and wind estimation for flying robots.

Reference	Method	Considered air density
[108]	Acceleration-based	Known
[109]	Aerodynamic power-based	Known
[110]	Constant mapping of controls to wrench	Lumped
[111]	Momentum-based	Lumped
[112], [113]	Acceleration-/momentum-based	Known
[83]	Lyapunov-based	Lumped, constant
[85]	Unscented quaternion-based	Lumped, constant
This work	Acceleration-/momentum-based	Unknown, variable

The passivity-based control approach presented recently by Yüksel et al. [83] either requires a direct measurement of the external wrench or uses a nonlinear Lyapunov-based wrench observer. The latter depends heavily on the choice of the observer gain, shows poor performance compared to sensor measurements, and neglects the influence of the air density on the propulsion wrench. The external force and torque estimation for quadrocopters proposed by McKinnon et al. [85] is based on the well-known unscented quaternion estimator and also uses a lumped thrust coefficient with constant air density.

3.1.3 Prioritized control allocation under saturation

The thrust of a flying robot is limited and highly depends on the incoming airflow. The relationship between propeller forces and airspeed is highly nonlinear [114] and no control allocation algorithm actually considers this. However, the maximum thrust mainly depends on the maximum rotor speed, which in turn depends on the motor/propeller combination and on the available battery voltage.

Since attitude control is realized via differential thrust, control authority in roll, pitch, and yaw is reduced, as soon as all rotors reach their maximum or minimum speed. Also, position control performance in a cascaded loop highly depends on accurate attitude tracking. Going into torque saturations will therefore also negatively impact position control performance. This fact is crucial for robustness and can be accounted for on different levels in flying robot control. It is important to notice, that saturation of the actuators means that the vehicle is not able to follow the precomputed trajectory, for instance because of wind disturbances or interaction forces. Any saturation handling will lead to a deviation from the desired trajectory.

In the standard literature, control allocation with prioritization of the generalized forces is achieved using a weighted pseudoinverse [115]. Though, the weighting is usually constant and this method does not consider actuator limits. Raffler et al. [116] present pseudo control hedging for multicopter path following. Pseudo control hedging is introduced by Johnson et al. [117] for adaptive control of fixed-wing UAVs. It uses a virtual control input on velocity or acceleration level and alters the input trajectory such that the limits of the control surfaces (or actuators) are not exceeded. In other words, it changes the reference trajectory in response to saturation. It is straightforward to implement and applicable for

Table 3.3: State-of-the-art prioritized control allocation for flying robots.

Reference	Method	Measured rotor speed
[92], p. 404	Weighted pseudoinverse	✗
[116, 117]	Pseudo control hedging	✗
[118]	Iterative root finding	✗
[120], [119]	Active set	✗
This work	Direct or iterative	✓

rate limiting and if a precomputed flight trajectory is available. Faessler et al. [118] present a trivial solution to control allocation under saturation by iteratively reducing the thrust of saturated rotors until the rotor speed limits are met. For this, an algebraic relation between rotor thrust and torque based on the established quadratic model is derived. Experimental results are presented only for saturation of the yaw torque. Smeur et al. [119] formulate the control allocation as a constrained least squares problem and use weighted least squares and an active set method [120] to solve the problem. It is a computationally expensive but real-time capable approach. Similar to the above approach, only experimental results for the yaw direction are presented. Note that all of the above approaches do not consider the actual rotor speed due to a lack of appropriate sensors, but use the commanded rotor speeds. Online recomputation or replanning of the trajectory [121, 122] is a different (and usually computationally expensive) approach, which has to be implemented on the motion planning level, rather than on the position and attitude control level. It is therefore not treated in detail here. In contrast to state-of-the-art approaches, the direct and iterative saturation handling procedures presented in this work consider a hierarchical prioritization instead of a fixed weighting matrix. They do not require (but can make use of) a numerical solution of the constrained control allocation problem.

3.2 Flying robot modeling

The components that constitute the flying robot plant model are summarized in Figure 3.2. In general, the model is split into rigid body dynamics, actuator (motor) dynamics, and aerodynamics, which are all presented in detail in the next sections. Figure 3.3 depicts a quadrocopter which serves as an example of a flying robot.

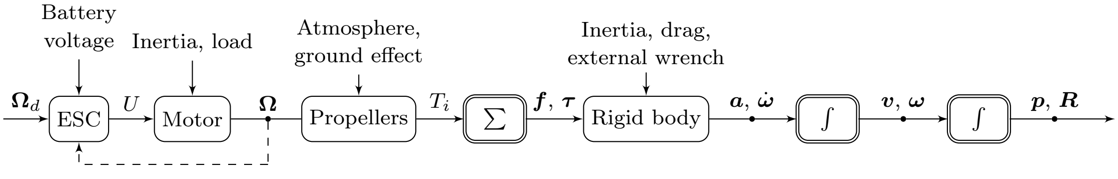

Battery voltage
Inertia, load
Atmosphere, ground effect
Inertia, drag, external wrench
$\boldsymbol{\Omega}_d$
ESC
U
Motor
$\boldsymbol{\Omega}$
Propellers
T_i
$\sum$
$\boldsymbol{f}$, $\boldsymbol{\tau}$
Rigid body
$\boldsymbol{a}$, $\dot{\boldsymbol{\omega}}$
$\int$
$\boldsymbol{v}$, $\boldsymbol{\omega}$
$\int$
$\boldsymbol{p}$, $\boldsymbol{R}$

Figure 3.2: Flying robot plant model with external influences and uncertainties.

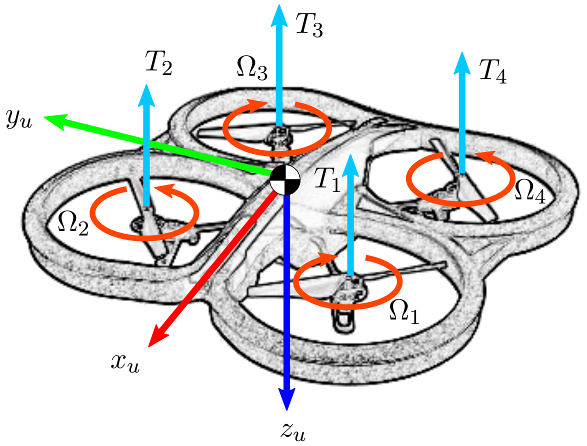


Figure 3.3: Quadrocopter with rotation directions of rotors Ω_i, rotor thrust forces T_i, and body coordinate frame Ψ_u. The origin of the body-fixed frame is the center of gravity of the rigid body. The first axis points to the front of the vehicle, the second axis points to the right, and the third axis points downwards, i.e. the three axes form a right-handed orthonormal base. Traditionally, the pose of the vehicle is then described w.r.t. a North-East-Down (NED) world frame. For convenience, the third axes is sometimes defined upwards [100] which means that the second axis also changes direction and which results in the Forward-Left-Up (FLU) body frame and the East-North-Up (ENU) world reference frame [123].

3.2.1 Considered rigid body model

The rigid body of a flying robot is composed of the fuselage (or frame) and the rotors. The model employed by [124] is directly taken from [125] and considers the inertia of N rotors

$$\boldsymbol{I}_u\dot{\boldsymbol{\omega}} + \sum_{i=1}^{N}\boldsymbol{I}_p\dot{\boldsymbol{\omega}}_{pi} + \boldsymbol{\omega} \times \left(\boldsymbol{I}_u\boldsymbol{\omega} + \sum_{i=1}^{N}\boldsymbol{I}_p(\boldsymbol{\omega} + \boldsymbol{\omega}_{pi})\right) = \boldsymbol{\tau}, \tag{3.1}$$

where $\boldsymbol{I}_u$ and $\boldsymbol{\omega}$ are inertia and angular velocity of the fuselage, $\boldsymbol{\omega}_{pi}$ are the rotor speeds relative to the fuselage with $i \in \{1, \ldots, N\}$. The inertia $\boldsymbol{I}_p$ is assumed equal for all N propellers. This model is especially applicable to helicopter UAVs, where the main rotor is relatively large compared to the fuselage and therefore contributes more to the overall inertia. It is shown in [126], that modeling a helicopter as two rigid bodies increases control performance. Instead of rotating blades, it is sufficient to treat the rotor as a static, solid disk.

Usually, a multicopter is sufficiently well approximated by a single rigid body. Hence, its dynamics are given by the Newton Euler equations and can concisely be written as [112]

$$m\ddot{\boldsymbol{p}} = -mg\boldsymbol{e}_3 + T\boldsymbol{R}_{iu}\boldsymbol{e}_3 + \boldsymbol{f}_{ext}, \tag{3.2}$$

$$\boldsymbol{I}_u\dot{\boldsymbol{\omega}} = \boldsymbol{S}(\boldsymbol{I}_u\boldsymbol{\omega})\boldsymbol{\omega} + \boldsymbol{\tau}_{cog} + \boldsymbol{\tau} + \boldsymbol{\tau}_{ext}, \tag{3.3}$$

$$\dot{\boldsymbol{R}}_{iu} = \boldsymbol{R}_{iu}\boldsymbol{S}(\boldsymbol{\omega}). \tag{3.4}$$

Therein, m is the mass, $\boldsymbol{I}_u$ is the overall inertia tensor of the flying robot, g is the acceleration of gravity, $\boldsymbol{e}_3 = (0 \quad 0 \quad 1)^T$ is a unit vector, T is the scalar cumulative thrust, and $\boldsymbol{\tau}$ is the input torque. The angular velocity of body w.r.t. inertial frame is denoted with $\boldsymbol{\omega}$, $\boldsymbol{R}_{iu}$ is the rotation matrix from body to inertial frame, and $\boldsymbol{S}(\cdot)$ is the skew-symmetric matrix representation of the cross product (2.5). Different representations of $\boldsymbol{R}_{iu}$ are presented in Section 2.4.1. The geometric center of the multicopter frame is chosen as reference point. An offset $\boldsymbol{r}_g$ between the origin of the body-fixed frame and the center-of-gravity of the fuselage is accounted for by the term $\boldsymbol{\tau}_{cog} = mg\boldsymbol{S}(\boldsymbol{r}_g)\boldsymbol{R}_{iu}^T\boldsymbol{e}_3$. Furthermore, $\boldsymbol{f}_{ext}$ and $\boldsymbol{\tau}_{ext}$ are external forces and torques, arising from physical interaction with the environment, from external disturbances like wind, or from parametric uncertainties.

3.2.2 Considered propulsion model

The propulsion system of a flying robot consists of two main parts: the motor and the rotor blades (or propellers).

The simplified dynamics of a standard brushless direct current (BLDC) motor are [127]

$$U_a = K_e\Omega + i_aR_a + L_a\frac{\mathrm{d}i_a}{\mathrm{d}t}, \tag{3.5}$$

$$\tau_m = (K_{q0} - K_{q1}i_a)i_a, \tag{3.6}$$

$$I_r\dot{\Omega} = \tau_m - \tau. \tag{3.7}$$

Therein, U_a, i_a are voltage and current through the motor and Ω is the rotor speed. Furthermore, R_a is the motor resistance and L_a is the motor coil inductance. The constant K_e is related to the motor's K_v-rating [127] defined as $\Omega = K_vU_a$ and K_{q0} and K_{q1} are specific constants that need to be identified for a given BLDC motor. Finally, I_r is the motor inertia, τ_m is the motor torque, and τ is the aerodynamic drag torque defined below in (3.10).

For constant rotor speed and constant current and if the quadratic term $K_{q1}i_a^2$ (that accounts for degrading torque efficiency associated with high currents) is neglected, it follows that

$$K_{q0}\left(\frac{U_a}{R_a} - \frac{K_e}{R_a}\Omega\right) = \tau. \tag{3.8}$$

From the above the following observations can be drawn:

- For a fixed load τ, the motor speed Ω is only affected by the applied voltage U_a. An increase in voltage results in an increase in speed.
- For a fixed voltage U_a, the motor speed Ω is inversely affected by the load τ. An increase in load torque results in a decrease in speed.

The exact load on the motor is unknown in practice. It depends on the aerodynamics of the propeller and changes with varying rotor inflow conditions. In addition to the unknown load on the motor, the rotor speed is affected by changes in battery voltage U_{bat}. The voltage supplied to the motor is controlled using an electronic speed controller (ESC). The ESC supplies a defined voltage $U = kU_{bat}$ to generate a desired rotor speed. Modern ESCs for BLDC motors use either six-step commutation or field oriented control (FOC) [128, 129]. Six-step commutation involves a zero-crossing detection to adjust the switching time and to acquire a motor speed estimate. Closed-loop speed control is typically not implemented. FOC uses an estimate of the electric speed and angle to commutate the motor. This results in better torque control and increased efficiency [128]. Other ESCs, like ESC32 [130], or the approaches presented in [127] and [131] also directly implement rotor speed control on the microcontroller of the ESC. For ESCs without rotor speed feedback control, battery voltage can be measured and the drop in thrust force can be compensated as done by [118, 132]. However, a model-based voltage compensation is still prone to modeling errors and sensor noise.

The second component of the rotor system is the propeller, which is spun by the motor and which produces a force depending on the airfoil of the propeller blades. In principle, the rotor blades produce thrust by moving air through the rotor disk plane which creates a flowfield. This induced flowfield contributes to the local blade incidence and the dynamic pressure which results in a lifting force [114]. Today, highly accurate aerodynamics models including simulation of turbulence can be derived with the aid of computational fluid dynamics (CFD) [133]. However, this is computationally demanding and not suitable for real-time control and therefore not considered in this work. For flight dynamics around hover, it is sufficient to consider only the normal component of the inflow [134], which is called the induced downwash. In classical helicopter aerodynamics, rotor thrust is modeled using momentum theory, blade element theory, or combinations thereof [114]. For flight control, momentum theory and a steady-state solution of the rotor inflow are usually utilized [100]. It is assumed that the flying robot is hovering, i.e. not translating horizontally or vertically, which leads to

$$T_i = \rho C_T A R^2 \Omega_i^2, \tag{3.9}$$

where T_i is the thrust produced by a single rotor i, C_T is the thrust coefficient, ρ is the air density, A is the rotor disk area, R is the rotor radius, and Ω_i is the rotor speed. For convenience, rotor specific parameters, such as rotor radius and rotor disk area, may be lumped in the thrust and torque coefficients [100]. Then, rotor thrust T_i and torque τ_i are given by

$$\begin{aligned} T_i &= \rho c \Omega_i^2, \\ \tau_i &= \rho k \Omega_i^2, \end{aligned} \tag{3.10}$$

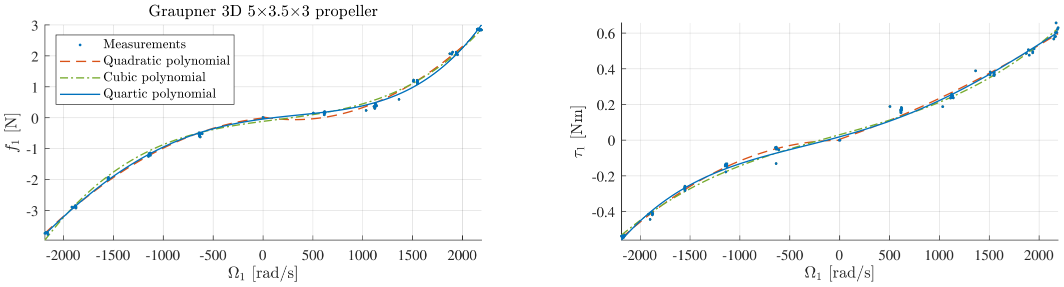
where c and k are the thrust and torque coefficients, respectively. They are identified experimentally as depicted in Figure 3.4. In contrast to [100], here the air density ρ is explicitly not lumped in c and k.

It is shown in [135], that the assumption $T_i \propto \Omega_i^2$ does not hold for flights at high forward velocities. The propeller identification in Section 3.2.5 also reveals, that bidirectional thrust under static conditions follows a polynomial with order ≥ 2 (see Figure 3.4), where offset and linear term are due to losses at high rotor speeds, while a cubic term is not physically motivated, but convenient because it naturally covers the transition from positive to negative thrust. In the literature, more accurate thrust (or torque) to rotor speed mappings are reported using a second order polynomial (cf. Figure 3.4) of the form [136, 118, 215]

$$T_i = a_{T0} + a_{T1}\Omega_i + a_{T2}\Omega_i^2, \tag{3.11}$$

$$\tau_i = a_{Q0} + a_{Q1}\Omega_i + a_{Q2}\Omega_i^2. \tag{3.12}$$

The offset and the linear term in (3.11) are due to losses at high rotor speeds [114]. Polynomials with order > 2 are not considered by standard control allocation procedures. Thus,



(a) Force profile.

(b) Torque profile.

Figure 3.4: Comparison of quadratic (2nd order), cubic (3rd order), and quartic (4th order) polynomial fitting of rotor thrust and torque measurements.

the control allocation is generalized to polynomial propulsion models in Section 3.4.3 and Section 3.4.4.

3.2.3 Ground effect models

Rotorcraft operating near ground experience the so-called ground effect, which essentially increases the net rotor thrust because of increased inflow due to rotor downwash reflection on the ground (see [137] and the references therein). It is clear that this effect needs to be accounted for especially during take-off and landing.

Empirical ground effect models for helicopters from the literature [138] are

$$\frac{T_{IGE}}{T_{OGE}} = \left(1 - \frac{1}{(4h/R)^2}\right)^{-1} \qquad \text{Cheeseman and Bennet} \tag{3.13}$$

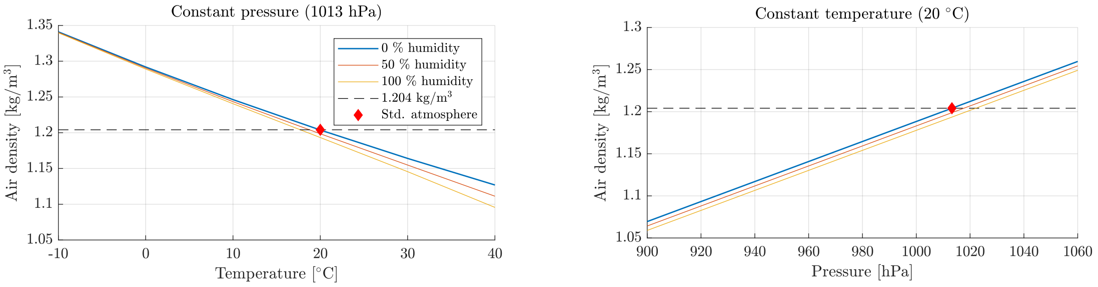
$$\frac{T_{IGE}}{T_{OGE}} = \left(0.9926 + \frac{0.03794}{(h/2R)^2}\right)^{2/3} \qquad \text{Hayden} \tag{3.14}$$

where h is the height above ground, R is the rotor radius, and the indices IGE and OGE describe the states "In Ground Effect" and "Out of Ground Effect", respectively. For multicopters, several definitions of the effective rotor radius R exist. For example, the distance from the outer tip of one rotor to the vehicles center may be used [138]. In [139], it is shown that the ground effect can also be modeled as a spring. The predominant opinion in the literature is that the ground effect is measurable up to $h = 6R$. Note that the above formulas are not explicitly considered for control. Instead, the adaptive control approach presented in Section 3.5 implicitly accounts for the ground effect, since it decreases (or increases) the thrust in response to a pose error induced by the ground effect.

3.2.4 Atmosphere model

To assess the variation of the air density w.r.t. to the local atmosphere and, hence, the variation of the thrust of a multirotor, an established atmosphere model shall be used. The air density ρ may be calculated using the well-known gas equation ([140], p. 63)

$$\rho = \frac{p}{R_f T_K}, \tag{3.15}$$



(a) Air density ρ w.r.t. air temperature T_C at constant pressure p.

(b) Air density ρ w.r.t. pressure p at constant air temperature T_C.

Figure 3.5: Variation of the air density computed using the atmosphere model presented in Section 3.2.4. The sea level standard atmosphere is defined as $p_{std} = 1013\,\text{hPa}$, $T_{std} = 20\,°\text{C}$, and $\delta = 0$, which yields $\rho_{std} = 1.204\,\text{kg/m}^3$ (red diamond). Copyright © 2020 IEEE [209].

where p is the air pressure in Pascal (Pa), T_K is the temperature in Kelvin (K) and R_f is the gas constant. Note that the air density scales linearly with air temperature and air pressure. Its dependency on the relative humidity δ is considered via

$$R_f = \frac{R_d}{1 - \delta \frac{p_{sat}}{p}(1 - \frac{R_d}{R_s})}, \tag{3.16}$$

with

$$\begin{aligned} R_d &= 287.058\,\text{J} \;/\; (\text{kg K}) \quad \text{dry air}, \\ R_s &= 461.523\,\text{J} \;/\; (\text{kg K}) \quad \text{steam}. \end{aligned}$$

Therein, p_{sat} is the saturation vapor pressure which is obtained using the empirical Magnus formula [141]

$$p_{sat} = p_0 \exp \frac{C_1 T_C}{C_2 + T_C}, \tag{3.17}$$

where

$$\begin{cases} C_1 = 17.08085, & C_2 = 234.175\,°\text{C}, \quad \text{if} \quad T_C \geq 0\,°\text{C}, \\ C_1 = 17.84362, & C_2 = 245.425\,°\text{C}, \quad \text{if} \quad T_C < 0\,°\text{C}, \end{cases}$$

with $p_0 = 610.78\,\text{Pa}$ and T_C being the air temperature in degrees Celsius (°C). Figure 3.5 visualizes the influence of absolute air pressure p, temperature T_C, and relative humidity δ on the air density ρ. Figure 3.5a and Figure 3.5b reveal a difference in the air density of approximately 20 % in the considered temperature and pressure range. That means a difference in thrust and torque of up to 20 %, since both scale linearly with ρ (cf. (3.10)). This variation has to be compensated by both the position and the attitude controller. Furthermore, it becomes clear that the relative humidity has minor influence on the air density (between 0.11 % and 2.86 % in Figure 3.5a and between 0.99 % and 0.84 % in Figure 3.5b).

3.2.5 Parameter identification procedure

Motivated by the approach presented in [19], the following hierarchical parameter identification procedure is proposed: First, the parameters of the quadratic (3.10) or polynomial propulsion model (3.11) are identified in a static experiment using a force-torque sensor and

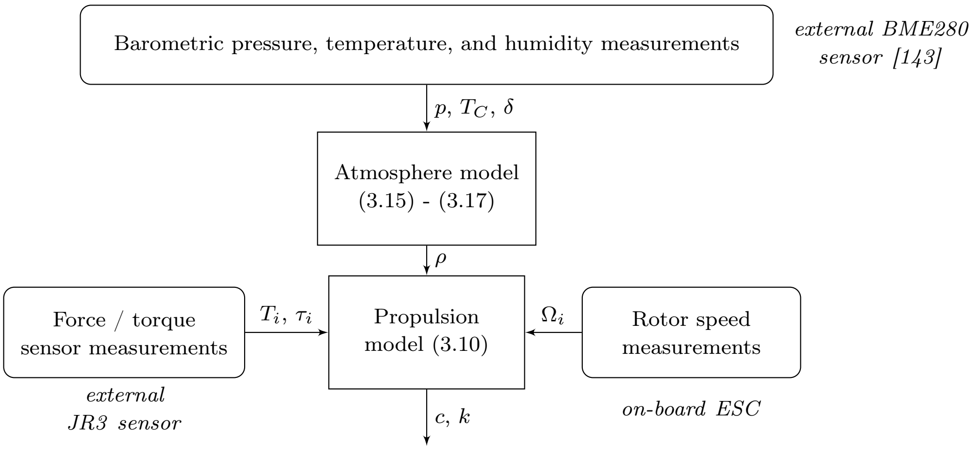

Barometric pressure, temperature, and humidity measurements
external BME280 sensor [143]
p, T_C, δ
Atmosphere model (3.15) - (3.17)
ρ
Force / torque sensor measurements
external JR3 sensor
T_i, τ_i
Propulsion model (3.10)
Ω_i
Rotor speed measurements
on-board ESC
c, k

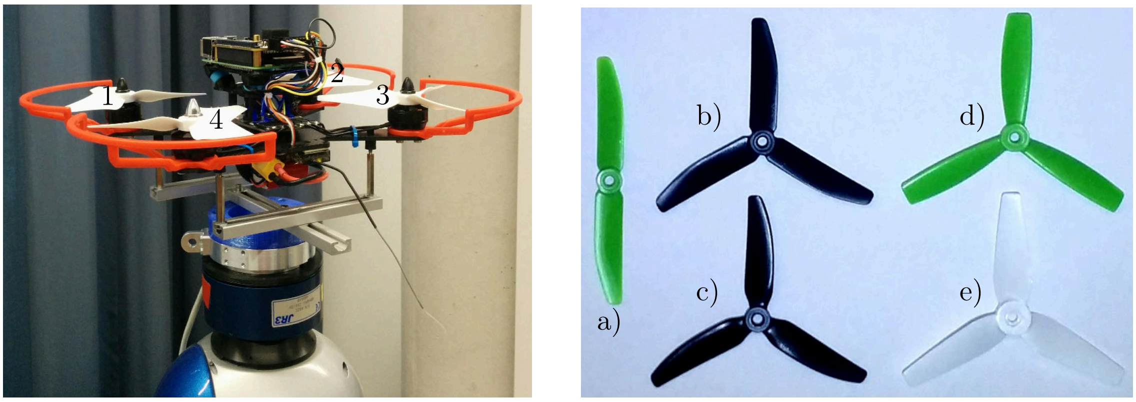
Figure 3.6: Offline identification of thrust and torque coefficients assuming a constant atmosphere and no disturbances.

by measuring force, torque, and rotor speed (cf. Figure 3.6). An exemplary propeller identification is described in detail below. The motor model (3.7) can be identified independently on a test bench via voltage, current, motor speed, and motor torque measurements. The air density ρ can be calculated using an atmosphere model and temperature, barometric pressure, and humidity sensor measurements. A possible uncertainty of the air density due to changing weather or altitude requires continuous on-board sensor measurements or an adaptive control approach (see Section 3.5). The mass m of the flying robot is straightforward to measure using a scale. To determine the location of the center-of-gravity (CoG), multiple scales can be used to measure the forces at four corners of the aerial vehicle. From a moment equilibrium and known forces and locations of the force measurements, two coordinates of the CoG may be determined, i.e. two sets of measurements are necessary to determine all three coordinates of the CoG. To identify the inertia, dynamic experiments with a gravitational, a torsional, or a multi-filar pendulum [142] can be performed. Assuming that the other parameters mentioned above are already known, the CoG and the inertia can also be identified from flight data using the dynamical model (3.2), (3.3), (3.4) and acceleration and rotor speed measurements. That leads to a system of equations which can be solved in a least squares sense [19]. Note that for the parameters to converge, the three rotational DoF must be sufficiently excited in the flight experiment.

Propeller identification

A propeller can be characterized by its diameter, slope, and blade number, but the main influence is its airfoil [144]. To study the latter in detail and to find the most suitable propeller for thrust inversion, accurate aerodynamic models and wind tunnel measurements are needed [135]. This is beyond the scope of this thesis. Instead, the propellers for considered quadrocopter Sparrow[3] are identified via a static thrust test. For this, the quadrocopter is mounted on a JR3 force-torque sensor, as shown in Figure 3.7a.

[3] The system identification of the DLR hexacopter Ardea is treated in detail in [19].



(a) Quadrocopter MAV mounted on the JR3 force-torque sensor (maximum rating 40 N/4 N m@1 kHz) for static thrust tests with rotor numbers 1 to 4. The rotation directions are 1: CW, 2: CCW, 3: CW, 4: CCW.

(b) Selection of 5 inch propellers: a) HQ 5x4x2, b) HQ 5x4x3, c) HQ 5x4x3V1S, d) HQ 5x4.5x3DP, e) Graupner 3D 5x3.5x3, where the first number is the diameter in inch, second number is the slope in inch, and the third parameter is the number of blades.

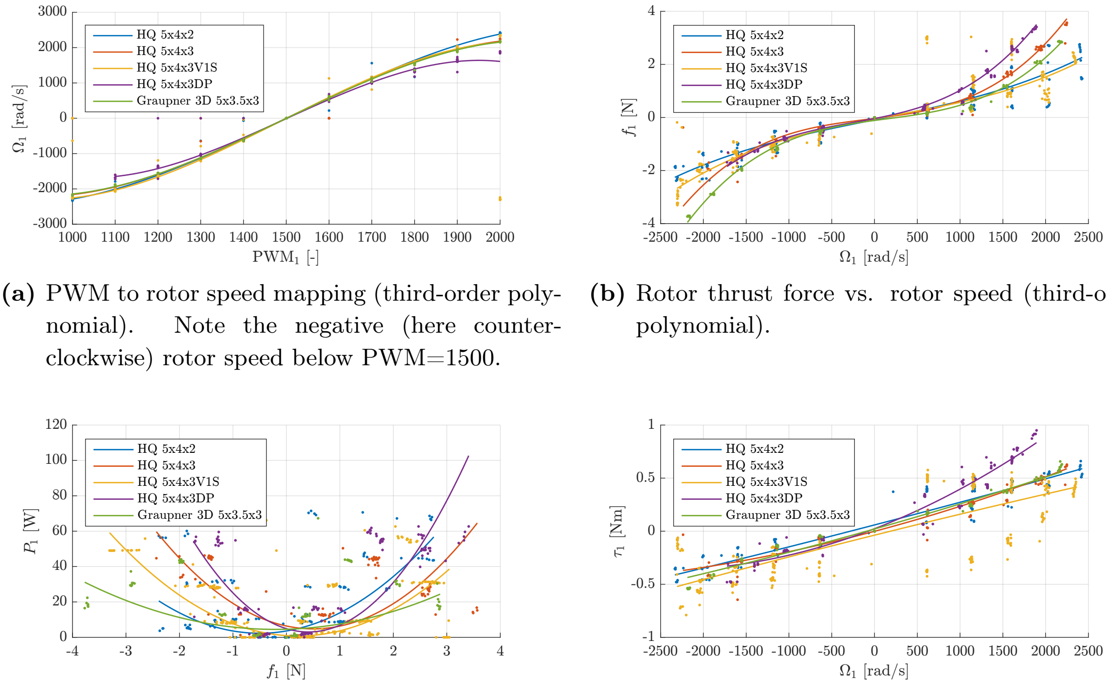
Figure 3.7: Propeller identification performed using a force-torque sensor. Copyright © 2018 IEEE [215].

Table 3.4: Propeller names, materials, and masses.

	Name	Material	Mass [g]
a)	HQ 5x4x2	Glass fiber composite	2.7
b)	HQ 5x4x3	Glass fiber composite	3.7
c)	HQ 5x4x3V1S	Poly carbonate	3.5
d)	HQ 5x4.5x3DP	Poly carbonate	5.8
e)	Graupner 3D 5x3.5x3	Polyamid glass fiber	5.8

Because of the size of the quadrocopter, only 5 inch propellers (see Figure 3.7b) are used. From the commercially available ones, a set of five propellers is selected with different blade numbers and slopes, different materials, and comparable mass (c.f. Table 3.4). For all tests a RHD D2204 2300kv brushless DC motor and a Floureon Lithium-polymer battery (3S, 11.1 V, 1500 mA h, 35C) are used. The on-board computer of the quadrocopter is a Raspberry Pi 3 with Raspbian Linux and real-time patch, an emlid Navio2 sensor hat, as well as completely custom-built autopilot soft- and middleware.

The KISS 24 A ESC accepts PWM input up to 800 Hz and sends serial telemetry messages at up to 1 kHz. A telemetry rate of 100 Hz was chosen and found sufficient due to the static nature of the tests. Starting from 1500 PWM (motor stop), steps of 100 PWM until 2000 PWM (full forward) and then down to 1000 PWM (full backward) are commanded. The measurements of the JR3 sensor are logged as well as the telemetry provided by the ESC, i.e. temperature, voltage, current, and revolutions per minute. Due to high frequency noise induced by the propellers spinning at up to 23000 1/min, the force and torque data is low-pass filtered resulting in measurements at 10 Hz. The ESC telemetry is manually synchronized with the force measurements.



(a) PWM to rotor speed mapping (third-order polynomial). Note the negative (here counter-clockwise) rotor speed below PWM=1500.

(b) Rotor thrust force vs. rotor speed (third-order polynomial).

(c) Power consumption vs. produced thrust force (second-order polynomial).

(d) Rotor torque vs. rotor speed (second-order polynomial).

Figure 3.8: Summarized test results for rotor $i = 1$ and all five propellers. Copyright © 2018 IEEE [215].

Figure 3.8 summarizes the outcome of the static thrust test for all five propellers (see Figure 3.7b) mounted at rotor $i = 1$, which rotates clockwise for upward and counter-clockwise for downward thrust. Besides the mapping of PWM$_1$ to rotor speed Ω_1 in Figure 3.8a, it shows the thrust f_1 w.r.t. Ω_1 (Figure 3.8b), the torque τ_1 w.r.t. Ω_1 (Figure 3.8d), as well as the electrical power $P = UI$ w.r.t. f_1 (Figure 3.8c).

Because of its small mass (cf. Table 3.4) and drag, the tested two-blade propeller HQ 5x4x2 spins the fastest, but due to its smaller blade area it does not produce as much upward and downward thrust as the other tested propellers. The propellers HQ 5x4x3 and HQ 5x4x3V1S almost share the same blade geometry, but the HQ 5x4x3V1S is much more flexible, which results in blade oscillations. That became evident in noisy force-torque sensor measurements at all rotor speeds. Instead, the propeller HQ 5x4x3 is stiffer, has less undercamber, and produces much more up- and downward thrust compared to the HQ 5x4x3V1S (cf. 3.8b). The HQ 5x4.5x3DP and the Graupner 3D propellers are the heaviest and stiffest propellers under test. The first one has the highest slope tested and therefore produces the highest thrust at maximum rotor speed, as can be seen in Figure 3.8b. However, it also has the highest power consumption (Figure 3.8c). Because of its unsymmetric undercambered blade geometry and the small downward thrust shown in Figure 3.8b, is is unsuitable for realizing bidirectional thrust. As expected, the Graupner 3D propeller produces the maximum downward thrust. To the best of the author's knowledge, it is currently the only available 5 inch propeller with a fully symmetric profile and neutral blade geometry. It has also the lowest slope of the tested propellers, which makes it suitable for flying at low velocities and which explains the low power consumption shown in Figure 3.8c. It is selected for further flight experiments with the quadrocopter depicted in Figure 3.7a. The identified coefficients of the propulsion model (3.10) or of the polynomials (3.11) and (3.12), respectively, are used in the multicopter control allocation described in Section 3.4.

3.3 Generalized trajectory tracking control

A flying robot has six degrees of freedom (three rotational and three translational degrees of freedom), but usually less control inputs, i.e. the system is underactuated. A demonstrative illustration of this fact is, that the vehicle needs to tilt its thrust vector and therefore change its orientation in order produce a force in a desired direction. This leads to the cascaded control structure as shown in Figure 3.9. The outer position controller computes a desired thrust force and a desired orientation, which are then forwarded to the inner attitude controller and to the control allocation, respectively [112]. Due to the flatness property of multicopter aerial vehicles with linear drag model [96, 99], it is also possible do derive the controls T in (3.2) and $\boldsymbol{\tau}$ in (3.3) or the angular rates $\boldsymbol{\omega}$ directly from the desired flat outputs (position $\boldsymbol{p}$ and yaw ψ) [97, 219, 99, 105]. Nevertheless, the cascaded structure is advantageous because the attitude control can be used even if a position controller is not present, e.g. if the flying robot is manually piloted.

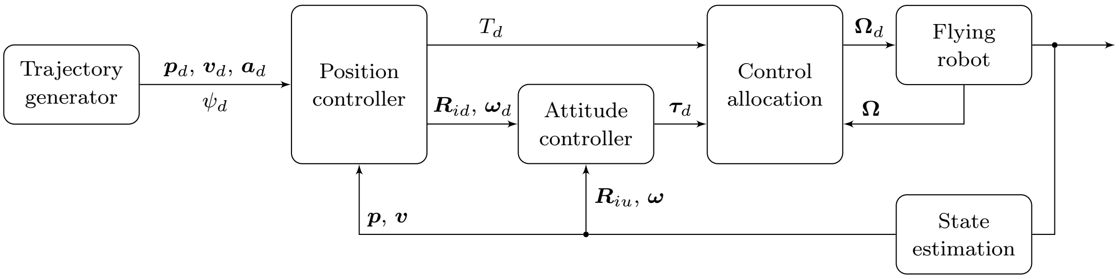

Trajectory generator
$\boldsymbol{p}_d$, $\boldsymbol{v}_d$, $\boldsymbol{a}_d$
ψ_d
Position controller
T_d
$\boldsymbol{R}_{id}$, $\boldsymbol{\omega}_d$
Attitude controller
$\boldsymbol{\tau}_d$
Control allocation
$\boldsymbol{\Omega}_d$
Flying robot
$\boldsymbol{\Omega}$
$\boldsymbol{p}$, $\boldsymbol{v}$
$\boldsymbol{R}_{iu}$, $\boldsymbol{\omega}$
State estimation

Figure 3.9: Cascaded attitude and position control loop.

In contrast to the underactuated flying robots mentioned above, there also exist fully actuated vehicles. Thus, in order to generalize trajectory tracking control, aerial vehicles with uni-, bi-, and omnidirectional thrust vector (i.e. a fully actuated control allocation matrix) are treated in the following.

3.3.1 Attitude controller

The inner loop of the cascaded controller depicted in Figure 3.9 is the attitude controller. For attitude (orientation) tracking control, the geometric control law from [29] is employed

$$\boldsymbol{\tau}_d = \boldsymbol{J}\left(\boldsymbol{K}_\omega(\boldsymbol{\omega}_d - \boldsymbol{\omega}) - \frac{1}{2}k_R\boldsymbol{e}_R\right), \tag{3.18}$$

but without accounting for the desired angular acceleration for simplicity, because otherwise reference jerk, reference snap, and the first derivative of the commanded thrust would be required [105]. In (3.18),

$$\boldsymbol{e}_R = (\boldsymbol{R}_{id}^T\boldsymbol{R}_{iu} - \boldsymbol{R}_{iu}^T\boldsymbol{R}_{id})^\vee \tag{3.19}$$

is the geometric attitude error, whose properties are provided in [29], $\boldsymbol{R}_{id}$ is the desired orientation, and $(\cdot)^{\vee}$ denotes the vee map, i.e. the inverse of the cross product operation $\boldsymbol{S}(\cdot)$. The positive definite matrix $\boldsymbol{K}_{\omega}$ and $k_R > 0$ are design parameters. The attitude controller (3.18) may be used for all flying robots considered in this work. An estimate $\hat{\boldsymbol{\tau}}_{ext}$ of the external disturbances can be added to (3.18) as shown in Section 3.5.4.

3.3.2 Upright and inverted equilibrium

In this thesis, the ability of a multicopter UAV to produce up- and downward thrust is demonstrated (see Figure 3.10). This is realized by using almost or fully symmetric propellers as well as modern ESCs, which are able to invert the motor's direction of rotation during flight (referred to as 3D mode). Basically, bidirectional thrust increases (and in the case of fully symmetric propellers it even doubles) the possible envelope of the control wrench compared to state-of-the-art approaches. A multicopter with bidirectional thrust differs from common unidirectional thrust vehicles, because it is able to stabilize its position and orientation at an upright and additionally at an inverted equilibrium. This allows to realize novel applications that are discussed in Section 3.6.1.

To derive the equilibria, recall the translational dynamics (3.2). A static equilibrium exists, if all derivatives of the state $\boldsymbol{p}$ vanish, hence, $\ddot{\boldsymbol{p}} = \boldsymbol{0}$ and

$$\boldsymbol{R}_{iu}(\boldsymbol{q})\boldsymbol{T} = mg\boldsymbol{e}_3. \tag{3.20}$$

Clearly, for vehicles with unidirectional thrust the only equilibrium, from now on referred to as upright equilibrium, is at

$$\boldsymbol{T} = \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix}, \quad \boldsymbol{R}_{iu} = \boldsymbol{R}_{\text{up}} = \begin{bmatrix} \cdot & \cdot & 0 \\ \cdot & \cdot & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.21}$$

where $\cdot$ denotes an arbitrary element in the rotation matrix. Note that for fully actuated vehicles [145, 146], there exists a region of attraction, whose size depends on the maximum applicable thrust force in $\boldsymbol{x}_b$- and $\boldsymbol{y}_b$-direction (cf. Figure 3.12). Because of the ability to

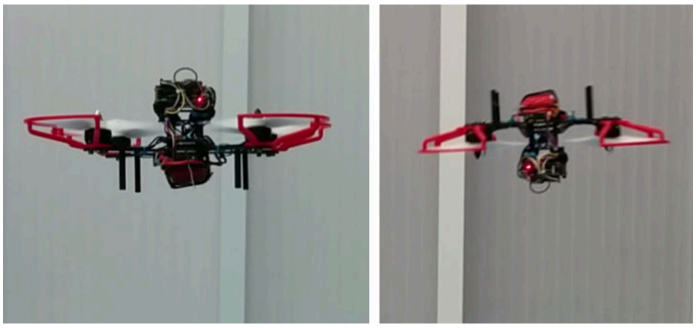


Figure 3.10: Quadrocopter with symmetric fixed-pitch propellers and bidirectional thrust flying upright (left) and inverted (right). Copyright © 2018 IEEE [215].

produce positive and negative thrust, flying robots with bidirectional thrust vector have a second stable equilibrium, the inverted equilibrium, at

$$\boldsymbol{T} = \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix}, \quad \boldsymbol{R}_{iu} = \boldsymbol{R}_{\text{inv}} = \begin{bmatrix} \cdot & \cdot & 0 \\ \cdot & \cdot & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{3.22}$$

For underactuated multicopter UAVs the equilibria correspond to an arbitrary yaw angle ψ and zero (upright) or π (inverted) tilt angle ϕ:

$$\cos(\phi) = \boldsymbol{e}_3^T \boldsymbol{R}_{iu} \boldsymbol{e}_3 = r_{3,3} = 1 - 2q_x^2 - 2q_y^2, \tag{3.23}$$

where $r_{3,3}$ is the lower right element of $\boldsymbol{R}_{iu}$ and the quaternion representation $\boldsymbol{q} = (\eta \quad \boldsymbol{\epsilon})^T$ follows from the Euler-Rodrigues formula (see Appendix A7). Hence, the transformation from upright to inverted equilibrium $\boldsymbol{R}_{\text{inv}} = \boldsymbol{R}_{\text{flip}} \boldsymbol{R}_{\text{up}}$ is obtained from (A7.1) in Appendix A7 and $q_w = q_z = 0$ as

$$\boldsymbol{R}_{\text{flip}} = \begin{bmatrix} 1 - 2q_y^2 & 2q_x q_y & 0 \\ 2q_x q_y & 1 - 2q_x^2 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \tag{3.24}$$

where the flip axis is defined as $\boldsymbol{\epsilon} = (q_x \quad q_y \quad 0)^T$ with $||\boldsymbol{\epsilon}|| = 1$.

In order to hold a constant altitude, the maximum tilt angle of a multicopter is given by

$$\cos(\phi_{\text{max}}) = \frac{mg}{T_{\text{max}}}. \tag{3.25}$$

Therefore, to transition from the upright to the inverted equilibrium and back, a suitable trajectory has to be generated (see Appendix A6). From an energetic point of view, there is no necessity to switch between the upright and the inverted equilibrium under normal flight conditions, e.g. hovering or path following, except if one is more efficient due to the propeller profile. Depending on the task of the flying robot, it might be sufficient to stabilize either the upright or the inverted equilibrium. The decision which one to choose can be made based on the actual tilt angle and the shorter distance to the respective equilibrium.

3.3.3 Common unidirectional thrust

Commonly, the thrust of each fixed-pitch propeller of a flying robot with multiple rotors is modeled as unidirectional [100]. This has mainly two reasons: Firstly, common electronic speed controllers (ESCs), which are used to control the rotational speed of the motors [131], are programmed to drive the motor either clockwise (CW) or counter-clockwise (CCW) and the direction of rotation can not be reversed during flight. Secondly, the propellers used in flying robots are usually fixed-pitch, i.e. can not change the blade pitch as e.g. conventional helicopters do, and are unsymmetric. These propellers produce less or even no thrust, if spun in the opposite direction. For common flying robot tasks, the described state-of-the-art setup is the most efficient one. However, as will be shown in the control allocation in Section 3.4, it restricts the control wrench to half of the manifold achievable with bidirectional thrust rotors.

A control law to track a desired position $\boldsymbol{p}_d(t) = (x_d(t) \quad y_d(t) \quad z_d(t))^T$ is given by

$$\boldsymbol{f}_i = m\left(\ddot{\boldsymbol{p}}_d + \boldsymbol{K}_d\dot{\tilde{\boldsymbol{p}}} + \boldsymbol{K}_p\tilde{\boldsymbol{p}}\right) + mg\boldsymbol{e}_3, \tag{3.26}$$

where $\ddot{\boldsymbol{p}}_d$ is the desired linear acceleration, $\boldsymbol{K}_d \in \mathbb{R}^{3\times3}$ is a diagonal positive damping, $\boldsymbol{K}_p \in \mathbb{R}^{3\times3}$ is a diagonal positive proportional gain, and $\tilde{\boldsymbol{p}} = \boldsymbol{p}_d - \boldsymbol{p}$ is the position error.

The force $\boldsymbol{f}_i$ in the inertial frame has to be mapped to the vertical axis of the vehicle's body frame. Define $\boldsymbol{x}_u = \boldsymbol{R}_{iu}\boldsymbol{e}_1$, $\boldsymbol{y}_u = \boldsymbol{R}_{iu}\boldsymbol{e}_2$, and $\boldsymbol{z}_u = \boldsymbol{R}_{iu}\boldsymbol{e}_3$, with $\boldsymbol{e}_1 = (1 \quad 0 \quad 0)^T$, $\boldsymbol{e}_2 = (0 \quad 1 \quad 0)^T$, and $\boldsymbol{e}_3 = (0 \quad 0 \quad 1)^T$, respectively. It is clear, that $\boldsymbol{R}_{iu} = [\boldsymbol{x}_u \quad \boldsymbol{y}_u \quad \boldsymbol{z}_u]$ and $||\boldsymbol{x}_u|| = ||\boldsymbol{y}_u|| = ||\boldsymbol{z}_u|| = 1$. Then, the projected desired thrust is given by

$$T_d = \boldsymbol{z}_u^T\boldsymbol{f}_i = \cos(\phi)||\boldsymbol{f}_i||. \tag{3.27}$$

If the angle ϕ between $\boldsymbol{z}_u$ and $\boldsymbol{f}_i$ is zero, i.e. $\boldsymbol{f}_i$ directly points in the direction of $\boldsymbol{z}_u$, the projected thrust is equal to the Euclidean norm of $\boldsymbol{f}_i$.

From $\boldsymbol{f}_i$, the desired attitude of the flying robot is computed, e.g. as recently described in [99], or more efficiently as done in [98, 147] via the quaternion[4]

$$\boldsymbol{q}_d = \frac{1}{\sqrt{2(1 + \boldsymbol{f}_b^T\boldsymbol{f}_i)}}\begin{pmatrix} 1 + \boldsymbol{f}_b^T\boldsymbol{f}_i \\ \boldsymbol{f}_b \times \boldsymbol{f}_i \end{pmatrix}, \tag{3.28}$$

where the sign of the third component of $\boldsymbol{f}_b = (0 \quad 0 \quad \pm1)^T$ is chosen such that $\boldsymbol{f}_b^T\boldsymbol{f}_i \geq 0$. For $\boldsymbol{f}_i = \boldsymbol{0}$ it is impossible to derive the orientation using (3.28). This is uncommon, but for a flip trajectory it occurs during a very short period of time (see Figure 3.20d). Hence, at $\boldsymbol{f}_i = \boldsymbol{0}$ the orientation is hold constant. Subsequently, $\boldsymbol{q}_d$ is converted to a rotation matrix using (A7.1). The desired yaw angle ψ_d of the multicopter can be selected freely as needed for the task. The resulting rotation matrix $\boldsymbol{R}_{id} = \boldsymbol{R}(\boldsymbol{q}_d)\boldsymbol{R}_z(\psi_d)$ is then passed to the attitude controller (3.18).

The position and the attitude controller are PD controllers. Using Lyapunov theory, it is straight forward to show, that with this control laws and without external disturbances or model uncertainties, the position and attitude equilibria are asymptotically stable. Though, as mentioned before, parameter uncertainties in the model and in the produced thrust as well as external disturbances, like wind, will degrade trajectory tracking performance. Hence, in order to increase robustness, an adaptive control approach is presented in Section 3.5.

3.3.4 Extension to bidirectional thrust

If a flying robot is able to produce negative thrust in the body frame, i.e. if the thrust vector is bidirectional, an ambiguity exists between upright and inverted flight, because the force $\boldsymbol{f}_i$ is the same in both cases. Therefore, in practice upright or inverted thrust and the corresponding attitude equilibrium are selected using the binary switches

$$\sigma = \begin{cases} 1, & \text{upright}, \\ -1, & \text{inverted}, \end{cases} \qquad \boldsymbol{R}_{\text{switch}} = \begin{cases} \boldsymbol{E}_{3\times3}, & \text{upright}, \\ \boldsymbol{R}_{\text{flip}}, & \text{inverted}, \end{cases} \tag{3.29}$$

[4] The derivation of this formula can be found in Appendix A5.

with $T_d = \sigma \boldsymbol{z}_u^T \boldsymbol{f}_i$ as before and $\boldsymbol{R}_{\text{flip}}$ defined in (3.24). Again, the desired yaw angle ψ_d of the multicopter can be selected independently. The resulting rotation matrix

$$\boldsymbol{R}_{id} = \boldsymbol{R}_{\text{switch}} \boldsymbol{R}(\boldsymbol{q}_d) \boldsymbol{R}_z(\psi_d)$$

is then passed to the attitude controller (3.18).

3.3.5 Extension to omnidirectional thrust

More recent research on multicopter UAVs focuses on full actuation in six degrees of freedom, which is especially useful for disturbance rejection and physical interaction with the environment. This is achieved by passively tilting the rotors in a hexagonal configuration [145, 146, 148]. However, even in this configuration there is no downward thrust and, hence, the maximum downward acceleration is given by gravity. Recently, quadrocopters [149] and hexacopters [150] with actuated lever arms are developed. They are able to apply forces in any direction while maintaining an arbitrary orientation, at the cost of increased weight and complexity due to the additional actuators.

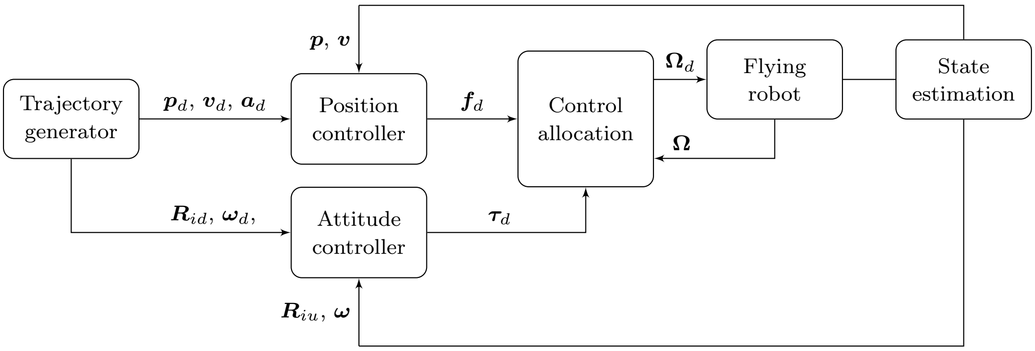


Figure 3.11: Attitude and position control loop for fully actuated aerial vehicles.

Researchers have also tried to overcome the limitations of fixed-pitch propellers by using variable-pitch propellers [147, 151, 152], like the ones used on helicopters, which include a swashplate, a pitch link, and a servo motor [153]. This results in a more complex controller design (rotor pitch has to be controlled in addition to rotor speed) and the need for additional mechanical parts. The latter is indeed crucial on flying robots, because their take-off-weight is limited by the maximum available thrust force. The resulting multicopters with non-parallel thrust vectors are fully actuated, i.e. position and orientation can be controlled independently. Hence, cascaded position and orientation control is superfluous, leading to the controller structure depicted in Figure 3.11.

3.4 Multicopter control allocation under saturation

The mapping of desired thrust T_d and torque $\boldsymbol{\tau}_d$ computed by position and attitude controller, respectively, to rotor speed commands Ω_i (or more general to so-called control surface deflections[5]) is referred to as control allocation ([92], p. 398). During fast maneuvers or due to external disturbances, it may occur in practice that the rotor speeds are saturated, i.e. that the attitude and position commands can not be realized. Therefore, a general control allocation procedure and different saturation handling strategies are discussed in this section.

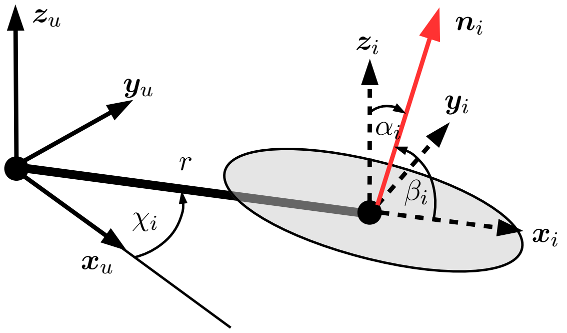


Figure 3.12: Rotor coordinate system and transformations of rotor i w.r.t. multicopter body frame u.

3.4.1 General control allocation for quadratic propulsion models

In the following, a general control allocation is presented considering flying robots with even number of propellers $N \geq 4$ and generic rotor configurations as shown in Figure 3.13. The position of propeller $i \in [1, N]$ in the flying robot's body frame u as shown in Figure 3.12 is

$$\boldsymbol{r}_i = r \cdot \begin{pmatrix} \cos(\chi_i) & \sin(\chi_i) & 0 \end{pmatrix}^T \tag{3.30}$$

and the rotor normal vector is

$$\boldsymbol{n}_i = \boldsymbol{R}_z(\chi_i)\boldsymbol{R}_y(\beta_i)\boldsymbol{R}_x(\alpha_i)\boldsymbol{e}_3. \tag{3.31}$$

From the propulsion model (3.10) it follows, that the thrust force vector $\boldsymbol{T}_i$ produced by each propeller i is derived as

$$\boldsymbol{T}_i = \rho c \boldsymbol{n}_i \Omega_i^2, \tag{3.32}$$

assuming equal thrust coefficients c for all rotors. Therein, ρ is the air density and $\boldsymbol{n}_i$ is the normal vector defined as above. The torque $\boldsymbol{\tau}_i$ generated by propeller i is given by

$$\boldsymbol{\tau}_i = \rho(c\boldsymbol{r}_i \times \boldsymbol{n}_i + P_i k \boldsymbol{n}_i)\Omega_i^2, \tag{3.33}$$

where k is the equal torque coefficient of all propellers and $P_i \in \{\text{-}1, 1\}$ denotes the propeller's direction of rotation (positive for clockwise and negative for counter clockwise revolutions). Finally, the mapping from rotor speeds Ω_i to forces $\boldsymbol{f} = \sum_{i=1}^{N} \boldsymbol{T}_i$ and torques $\boldsymbol{\tau} = \sum_{i=1}^{N} \boldsymbol{\tau}_i$, i.e. the control allocation matrix $\boldsymbol{B}$ of size $6 \times N$ with

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{pmatrix} = \rho \boldsymbol{B} \begin{pmatrix} \Omega_1^2 \\ \vdots \\ \Omega_N^2 \end{pmatrix} = \rho \boldsymbol{B} \boldsymbol{\varpi} \tag{3.34}$$

[5] A helicopter with main and tail rotor can be considered a special type of multicopter. However, on helicopters the blade pitch angle is actuated using a servo mechanism [154], in contrast to multicopters where the blade pitch angle is fixed.

is found as

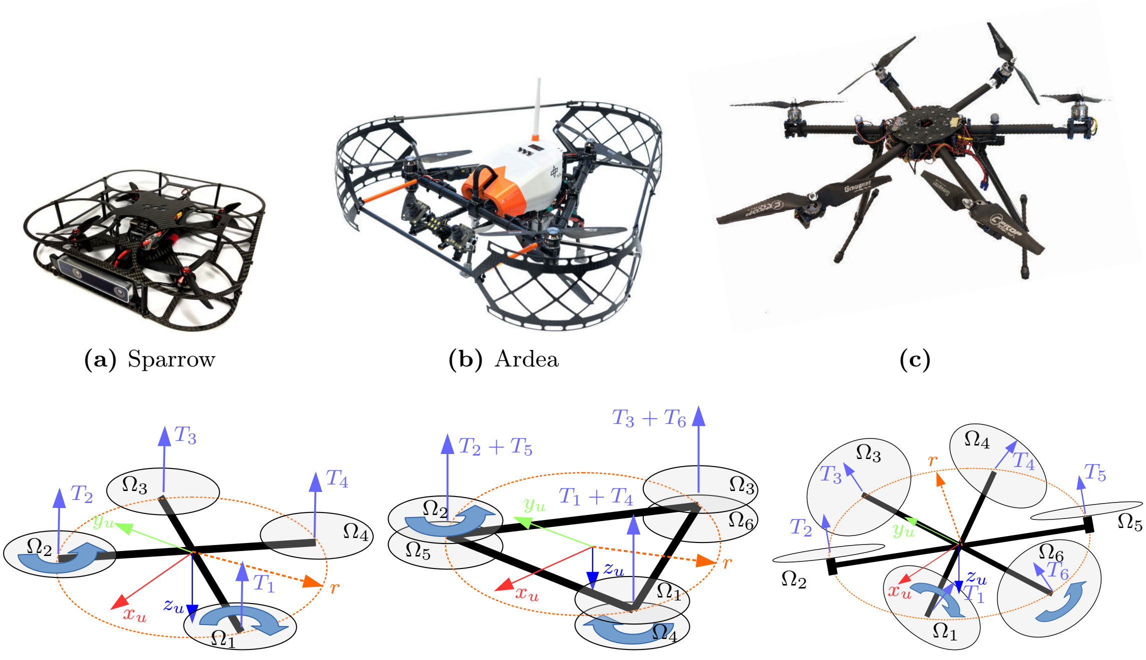
$$\boldsymbol{B} = \begin{bmatrix} c\boldsymbol{n}_1 & \cdots & c\boldsymbol{n}_N \\ c\boldsymbol{S}(\boldsymbol{r}_1)\boldsymbol{n}_1 + P_1 k\boldsymbol{n}_1 & \cdots & c\boldsymbol{S}(\boldsymbol{r}_N)\boldsymbol{n}_N + P_N k\boldsymbol{n}_N \end{bmatrix}, \tag{3.35}$$

where $\boldsymbol{\varpi} = (\Omega_1^2 \quad \cdots \quad \Omega_N^2)^T$ is a vector containing the squared rotor speeds. The multicopter is fully actuated for rank$(\boldsymbol{B}) = 6$, underactuated for rank$(\boldsymbol{B}) < 6$, and overactuated for rank$(\boldsymbol{B}) > 6$. Furthermore, a multicopter with more rotors then actuated degrees of freedom, i.e. $N >$ rank$(\boldsymbol{B})$, is referred to as redundant. Note that full actuation is associated and, hence, also verifiable with nonzero elements in the second and third row of the vector on the left side of (3.34).

If all N rotors lie in a plane, as for instance on most common multicopters, the cumulative thrust is a unidirectional vector through the center of mass and perpendicular to this plane. The above mapping is then defined using the scalar thrust T and the allocation matrix $\boldsymbol{B}(c, k, r) \in \mathbb{R}^{4\times N}$ as

$$\begin{pmatrix} T \\ \boldsymbol{\tau} \end{pmatrix} = \rho \boldsymbol{B}\boldsymbol{\varpi}. \tag{3.36}$$

In this work, a quadrocopter (subscript Q) as shown in Figure 3.13a and a coaxial hexacopter (subscript H) as depicted in Figure 3.13b are treated exemplarily. It holds for both configurations that all rotors lie on a circle with radius r. The coaxial rotor configuration of the hexacopter is characterized by different thrust and torque coefficients for upper (c_u, k_u) and lower propeller (c_l, k_l). This difference is due to the fact that the lower propeller operates in the downstream of the upper propeller and therefore has a different inflow velocity. The allocation matrices $\boldsymbol{B}_Q$ of the quadrocopter and $\boldsymbol{B}_H$ of the hexacopter are provided in Table 3.5.



(a) Sparrow **(b)** Ardea **(c)**

(d) Quadrocopter **(e)** Coaxial hexacopter **(f)** Fully actuated hexacopter

Figure 3.13: Examples of flying robots with multiple rotors used for the experiments.

Table 3.5: Exemplary control allocation matrices.

$\boldsymbol{B}_Q$	$\boldsymbol{B}_H$
$\begin{bmatrix} -c & -c & -c & -c \\ \frac{\sqrt{2}cr}{2} & -\frac{\sqrt{2}cr}{2} & -\frac{\sqrt{2}cr}{2} & \frac{\sqrt{2}cr}{2} \\ \frac{\sqrt{2}cr}{2} & \frac{\sqrt{2}cr}{2} & -\frac{\sqrt{2}cr}{2} & -\frac{\sqrt{2}cr}{2} \\ -k & k & -k & k \end{bmatrix}$	$\begin{bmatrix} -c_u & -c_u & -c_u & -c_l & -c_l & -c_l \\ \frac{\sqrt{3}\,c_u\,r}{2} & -\frac{\sqrt{3}\,c_u\,r}{2} & 0 & \frac{\sqrt{3}\,c_l\,r}{2} & -\frac{\sqrt{3}\,c_l\,r}{2} & 0 \\ \frac{c_u\,r}{2} & \frac{c_u\,r}{2} & -c_u\,r & \frac{c_l\,r}{2} & \frac{c_l\,r}{2} & -c_l\,r \\ k_u & k_u & k_u & -k_l & -k_l & -k_l \end{bmatrix}$

In order to obtain the required rotor speeds Ω_i for a desired thrust T_d and torque $\boldsymbol{\tau}_d$ commanded by position and attitude controller, respectively, the relation (3.36) needs to be inverted. For a quadrocopter, i.e. a quadratic allocation matrix $\boldsymbol{B}_Q \in \mathbb{R}^{4\times4}$, the general inverse $\boldsymbol{B}_Q^{\#} = \boldsymbol{B}_Q^{-1}$ is straightforward to compute. For a hexacopter with $\boldsymbol{B}_H \in \mathbb{R}^{4\times6}$, the Moore-Penrose pseudoinverse may be used

$$\boldsymbol{B}_H^{\#} = \boldsymbol{B}_H^T(\boldsymbol{B}_H\boldsymbol{B}_H^T)^{-1}, \tag{3.37}$$

which minimizes the Euclidean norm $\boldsymbol{\varpi}^T\boldsymbol{\varpi}$ (see Appendix A3 for a proof). The hexacopter is redundant. Under the assumption that the collective thrust is sufficient even with less than six rotors, the control allocation can be recomputed by deleting the column corresponding to the lost rotor in $\boldsymbol{B}_H$. However, an equilibrium of rotor forces and moments is only obtained for a finite set of remaining rotors that satisfies $\boldsymbol{B}_H\boldsymbol{B}_H^{\#} = \boldsymbol{E}$, i.e. for which $\boldsymbol{B}_H^{\#}$ exists. Otherwise, the vehicle will rotate with a defined steady-state angular velocity [124].

Finally, the general control allocation is given by

$$\boldsymbol{\varpi} = \frac{1}{\rho_m}\boldsymbol{B}^{\#}\begin{pmatrix} T_d \\ \boldsymbol{\tau}_d \end{pmatrix}, \tag{3.38}$$

where ρ_m is the model air density (usually considering the sea level standard atmosphere) and Ω_i is found by taking the square root of the corresponding element of $\boldsymbol{\varpi}$. Inserting (3.38) in (3.36) yields the following relation between actually applied and desired thrust

$$T = \frac{\rho}{\rho_m}T_d, \tag{3.39}$$

which will be used later on in Section 3.5 for the development of an adaptive controller.

3.4.2 Numerical comparison of uni- and bidirectional thrust

It is stated above, that bidirectional thrust increases the control wrench compared to unidirectional thrust. This is intuitively true for the scalar thrust T. For the torque $\boldsymbol{\tau}$ a numerical comparison is provided in Figure 3.14, generated by sampling the admissible rotor thrust. The numerical values in Table 3.6 apply to the quadrocopter Sparrow (see Figure 3.13a).

The maximum torques lie within a cuboid with slope $\Delta\tau = \frac{l}{\kappa}$. Note that the maxima of τ_x and τ_y are equal, thus, τ_y is not shown in Figure 3.14. Areas where the cumulative thrust

Table 3.6: Numerical comparison between uni- and bidirectional thrust for quadrocopter Sparrow.

Numerical example (Sparrow)
$\begin{pmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ l & -l & -l & l \\ l & l & -l & -l \\ -\kappa & \kappa & -\kappa & \kappa \end{bmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix}$ $\qquad \begin{matrix} l = 0.08\,\mathrm{m} \\ \kappa = 0.25 \\ m = 0.52\,\mathrm{kg} \\ g = 9.81\,\mathrm{m/s^2} \end{matrix}$
unidirectional thrust: $0 \leq T_i \leq 3.5\,\mathrm{N}$ bidirectional thrust: $-3.5\,\mathrm{N} \leq T_i \leq 3.5\,\mathrm{N}$

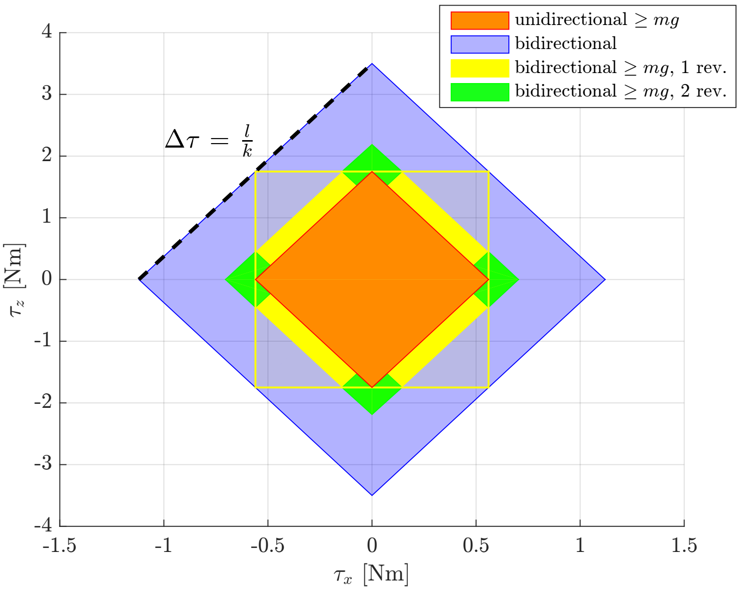


Figure 3.14: Comparison of available torques τ_x and τ_z with unidirectional and bidirectional thrust. The plot is generated by sampling the complete admissible rotor thrust range given in Table 3.6. Note that the actual torque limits depend on the thrust set-point and that the figure shows a cut at $\tau_y = 0$ through an octahedron.

exceeds gravity and where one or two propellers rotate in the reverse direction are highlighted using different colors in Figure 3.14. It can be concluded, that in this particular case the torque produced with bidirectional thrust (blue) indeed covers twice the area compared to the torque produced with unidirectional thrust (red).

3.4.3 Nonlinear control allocation for polynomial propulsion models

The control allocation can be extended to account for polynomial propulsion models. For this, the mapping of rotor forces and torques (3.32) - (3.35) is stated differently. The thrust vector $\boldsymbol{T}_i$ of each propeller in the flying robot's body frame can be written concisely as

$$\boldsymbol{T}_i = T_i \boldsymbol{n}_i \qquad (3.40)$$

and the torque $\boldsymbol{\tau}_i$ on the multicopter generated by thrust and torque of propeller i is

$$\boldsymbol{\tau}_i = T_i \boldsymbol{r}_i \times \boldsymbol{n}_i + P_i \tau_i \boldsymbol{n}_i. \tag{3.41}$$

Let $\boldsymbol{C}$ describe the mapping from rotor to cumulative force and torque, as

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{pmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{n}_1 \cdots \boldsymbol{n}_N & \boldsymbol{0} \cdots \boldsymbol{0} \\ \boldsymbol{S}(\boldsymbol{r}_1)\boldsymbol{n}_1 \cdots \boldsymbol{S}(\boldsymbol{r}_N)\boldsymbol{n}_N & P_1 \boldsymbol{n}_1 \cdots P_N \boldsymbol{n}_N \end{bmatrix}}_{\boldsymbol{C}} \boldsymbol{h}, \tag{3.42}$$

with $\boldsymbol{h} = (T_1 \quad \cdots \quad T_N \quad \tau_1 \quad \cdots \quad \tau_N)^T$. Now, instead of assuming $T_i \propto \Omega_i^2$, the vector $\boldsymbol{h}$ on the right side of (3.42) allows to consider arbitrary thrust and torque profiles of each rotor i, e.g. higher order polynomials. Inserting polynomials (3.11) and (3.12) in (3.42) results in a nonlinear system of equations. It can be formulated as

$$\boldsymbol{e}(\Omega_i) = \boldsymbol{C}\boldsymbol{h}(\Omega_i) - \begin{pmatrix} \boldsymbol{f}_d \\ \boldsymbol{\tau}_d \end{pmatrix}, \tag{3.43}$$

where $\boldsymbol{f}_d$ and $\boldsymbol{\tau}_d$ are desired force and torque, respectively, and $\boldsymbol{e}$ is the residual. Using a quadratic cost function $\Gamma = \frac{1}{2}\boldsymbol{e}^T \boldsymbol{W} \boldsymbol{e}$ with positive definite weighting matrix $\boldsymbol{W}$ ([92], p. 404), this may be solved by applying gradient (or steepest) descent[6] ([155], p. 21)

$$\boldsymbol{\Omega}^{k+1} = \boldsymbol{\Omega}^k - \boldsymbol{J}_e^{\#}(\boldsymbol{\Omega}^k)\boldsymbol{e}^k, \tag{3.44}$$

using a generalized pseudoinverse $\boldsymbol{J}_e^{\#}(\boldsymbol{\Omega})$ (see Appendix A3) of the Jacobian

$$\boldsymbol{J}_e(\boldsymbol{\Omega}) = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\Omega}}. \tag{3.45}$$

Then,

$$\boldsymbol{e}^k = \boldsymbol{C}\boldsymbol{h}(\boldsymbol{\Omega}^k) - \begin{pmatrix} \boldsymbol{f}_d \\ \boldsymbol{\tau}_d \end{pmatrix} \tag{3.46}$$

is the residual at iteration k and the optimization loop can run until Γ is below a given threshold. For example, the Jacobian matrix for the quadrocopter Sparrow is given by

$$\boldsymbol{J}_e = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -a_{T1} - 2a_{T2}\Omega_1 & -a_{T1} - 2a_{T2}\Omega_2 & -a_{T1} - 2a_{T2}\Omega_3 & -a_{T1} - 2a_{T2}\Omega_4 \\ \frac{\sqrt{2}r(a_{T1}+2a_{T2}\Omega_1)}{2} & -\frac{\sqrt{2}r(a_{T1}+2a_{T2}\Omega_2)}{2} & -\frac{\sqrt{2}r(a_{T1}+2a_{T2}\Omega_3)}{2} & \frac{\sqrt{2}r(a_{T1}+2a_{T2}\Omega_4)}{2} \\ \frac{\sqrt{2}r(a_{T1}+2a_{T2}\Omega_1)}{2} & \frac{\sqrt{2}r(a_{T1}+2a_{T2}\Omega_2)}{2} & -\frac{\sqrt{2}r(a_{T1}+2a_{T2}\Omega_3)}{2} & -\frac{\sqrt{2}r(a_{T1}+2a_{T2}\Omega_4)}{2} \\ -a_{Q1} - 2a_{Q2}\Omega_1 & a_{Q1} + 2a_{Q2}\Omega_2 & -a_{Q1} - 2a_{Q2}\Omega_3 & a_{Q1} + 2a_{Q2}\Omega_4 \end{pmatrix},$$

[6] One could also use the Levenberg-Marquardt method, which should result in faster convergence ([155], p. 258). Alternatively, (3.43) could be solved directly for $\boldsymbol{h}$ in a least squares sense using the generalized pseudoinverse (cf. Appendix A3). Then, the speed of each rotor can be found from $\boldsymbol{h}$, again using a line search method ([155], p. 30). In order to incorporate constraints on force and torque level, the solution for $\boldsymbol{h}$ could also be obtained using a quadratic programming solver, such as qpOASES [156]. However, since the constraints are actually on rotor speed level, the approaches in Section 3.4.7 and 3.4.6 seem more appropriate.

under the assumption, that thrust and torque w.r.t. rotor speeds Ω_i of all four rotors follow the quadratic polynomials (3.11) and (3.12), respectively. Since the quadrocopter is obviously not able to produce forces in the body-fixed x- and y-directions, it is sufficient and also more efficient to implement the symbolic inverse $\boldsymbol{J}_e^{-1}$ of the 4×4 submatrix of $\boldsymbol{J}_e$ with nonzero elements. The Jacobian matrices for higher order polynomials and multicopters with a larger number of rotors N follow analogously, but are omitted for brevity.

3.4.4 Incremental control allocation for polynomial propulsion models

Solving the nonlinear system of equations (3.43) in real-time is possible, but the computational complexity grows for higher order polynomials and an increasing number of rotors N. Therefore, a different and straight-forward control allocation algorithm is introduced in the following for comparison. Linearizing the polynomial propulsion model (3.11) and (3.12) at current (estimated or measured) rotor speeds Ω_{i0} yields

$$T_i(\Omega_i) \approx T_i(\Omega_{i0}) + \left.\frac{\partial T_i}{\partial \Omega_i}\right|_{\Omega_{i0}} (\Omega_i - \Omega_{i0}) + \mathcal{O}(2) \tag{3.47}$$

and

$$\tau_i(\Omega_i) \approx \tau_i(\Omega_{i0}) + \left.\frac{\partial \tau_i}{\partial \Omega_i}\right|_{\Omega_{i0}} (\Omega_i - \Omega_{i0}) + \mathcal{O}(2). \tag{3.48}$$

For convenience, one can define

$$\boldsymbol{A}_{2N\times N}(\Omega_{i0}) = \begin{bmatrix} \left.\frac{\partial T_1}{\partial \Omega_1}\right|_{\Omega_{10}} & & \\ & \ddots & \\ & & \left.\frac{\partial T_N}{\partial \Omega_N}\right|_{\Omega_{N0}} \\ \left.\frac{\partial \tau_1}{\partial \Omega_1}\right|_{\Omega_{10}} & & \\ & \ddots & \\ & & \left.\frac{\partial \tau_N}{\partial \Omega_N}\right|_{\Omega_{N0}} \end{bmatrix}. \tag{3.49}$$

Hence, it follows that

$$\boldsymbol{h} = \boldsymbol{h}_0(\Omega_{i0}) + \boldsymbol{A}(\Omega_{i0})\,(\boldsymbol{\Omega} - \boldsymbol{\Omega}_0) \tag{3.50}$$

and inserting the linearized $\boldsymbol{h}$ in (3.42) yields

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{pmatrix} = \boldsymbol{C}\boldsymbol{h}_0(\Omega_{i0}) + \boldsymbol{C}\boldsymbol{A}(\Omega_{i0})\,(\boldsymbol{\Omega} - \boldsymbol{\Omega}_0)\,. \tag{3.51}$$

The control allocation matrix can now be interpreted as $\boldsymbol{B} = \boldsymbol{C}\boldsymbol{A}$, which yields

$$\boldsymbol{\Omega} = \boldsymbol{\Omega}_0 + \boldsymbol{B}^{\#}\left(\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{pmatrix} - \boldsymbol{C}\boldsymbol{h}_0(\Omega_{i0})\right). \tag{3.52}$$

This is comparable to an incremental control law [104], since it computes the desired rotor speeds based on current (estimated or measured) rotor speeds. The rotor speeds Ω_{i0} can be obtained from ESC measurements, while $\boldsymbol{h}_0(\Omega_{i0})$ follows from the polynomial thrust and torque models. The advantage compared to the iterative approach in Section 3.4.3 is that (3.52) gives a direct result, i.e. no iterations are required. On the downside, the incremental approach involves the inversion of the matrix $\boldsymbol{CA}$ in each time step and it is highly sensitive to packet loss in the rotor speed feedback. Also, due to the linearization about the current rotor speed, it will produce inaccurate results, if large and abrupt changes in thrust and torque are commanded.

3.4.5 Prioritization of control inputs

The rotor speeds of a flying robot may saturate during fast maneuvers or if external disturbances are present. Even if the rotor speed limits of the vehicle are taken into account for motion planning, it is not guaranteed that these limits will not be exceeded in flight, for instance due to unconsidered aerodynamics, low battery, or wind disturbances. On the other hand, using too conservative limits in the trajectory design will prevent the vehicle from exploiting its full agility. Thus, it is important to incorporate the rotor speed limits $\Omega_{\max}$ and to handle saturation of Ω_i properly.

As soon as the rotor speeds of a multicopter saturate, the control authority is partially lost. For example, at full thrust no moments can be produced any more. Depending on the task, not all desired states of a multicopter are equally important. Therefore, it makes sense to prioritize the control inputs and to handle actuator saturation online. Note that the following applies to both the direct and the iterative control allocation approaches presented above.

Because of the symmetry of a multicopter, its heading has the least priority in most applications. In addition, the yaw angle has the least control authority, i.e. will be the first control input to saturate. In accordance with [118] and [119], roll and pitch have the highest priority, because thrust can only be applied in the correct direction, if roll and pitch are controlled accurately. Decreasing the magnitude of the thrust vector while maintaining its direction enables the vehicle to follow the desired trajectory. Summing up, the priorities are depicted in Figure 3.15.

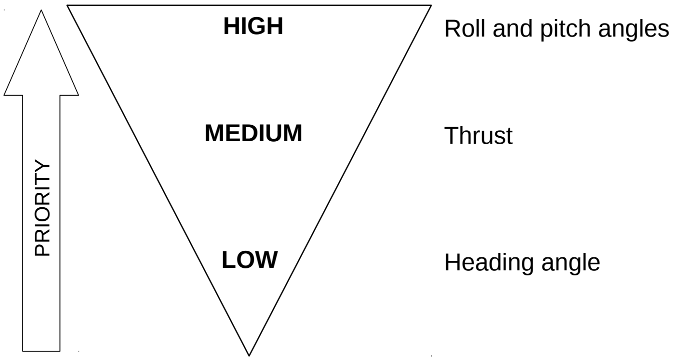


Figure 3.15: Priorities of the flying robot states.

Alternatively, if the vehicle should track an object, e.g. using an on-board camera, the full attitude can be prioritized higher than the thrust. Decreasing the thrust magnitude will inevitably lead to a deviation from the desired trajectory in all three translational degrees of freedom. Hence, for the saturation handling to work, the generated trajectory should not wind up, if the rotors of the multicopter are saturated. For this, instead of a temporal, a spatial tracking scheme [157] can be used. In the following, in contrast to [118], saturation handling is applied on rotor speed level, instead of on thrust level. This is due to the fact that rotor speed is directly measurable on a flying robot, but rotor thrust is not.

3.4.6 Direct maximum input allocation

In the following, it is shown that the maximum attainable inputs can be computed directly via the control allocation. Based on the prioritization of the control inputs, as defined in Section 3.4.5, and if $|\tau_{z,d}| > 0$, first compute a feasible yaw torque $\tau_{z,\max}$. Inserting the control allocation matrix $\boldsymbol{B}_Q$ of the quadrocopter with $l = \frac{\sqrt{2}r}{2}$ in (3.38) and solving for τ_z leads to four different conditions for the maximum yaw torque at maximum rotor speed $\Omega_{i,\max}$:

$$\tau_{z,\max} = \begin{cases} \frac{k}{cl}(\tau_{x,d} + \tau_{y,d} + lT_d - 4\rho_m cl\Omega_{1,\max}^2), \\ \frac{k}{cl}(\tau_{x,d} - \tau_{y,d} - lT_d + 4\rho_m cl\Omega_{2,\max}^2), \\ \frac{k}{cl}(-\tau_{x,d} - \tau_{y,d} + lT_d - 4\rho_m cl\Omega_{3,\max}^2), \\ \frac{k}{cl}(-\tau_{x,d} + \tau_{y,d} - lT_d + 4\rho_m cl\Omega_{4,\max}^2). \end{cases} \tag{3.53}$$

Note that applying the same approach to the control allocation matrix $\boldsymbol{B}_H$ of the hexacopter yields six conditions, which are not shown here for brevity. A sufficient condition for feasibility of the calculated maximum yaw torque is

$$\operatorname{sign}(\tau_{z,\max}) = \operatorname{sign}(\tau_{z,d}). \tag{3.54}$$

If condition (3.54) is fulfilled, it means that a smaller but equally directed yaw torque can be produced for the desired thrust force T_d, desired roll torque $\tau_{x,d}$, and desired pitch torque $\tau_{y,d}$. In this case $\boldsymbol{\Omega}$ can be reallocated using $\tau_{z,d} = \tau_{z,\max}$. Otherwise, set $\tau_{z,d} = 0$ and compute $T_{\max}$. The four conditions for $T_{\max}$ follow accordingly as

$$T_{\max} = \begin{cases} \frac{1}{kl}(-k\tau_{x,d} - k\tau_{y,d} + cl\tau_{z,d} + 4\rho_m ckl\Omega_{1,\max}^2), \\ \frac{1}{kl}(k\tau_{x,d} - k\tau_{y,d} - cl\tau_{z,d} + 4\rho_m ckl\Omega_{2,\max}^2), \\ \frac{1}{kl}(k\tau_{x,d} + k\tau_{y,d} + cl\tau_{z,d} + 4\rho_m ckl\Omega_{3,\max}^2), \\ \frac{1}{kl}(-k\tau_{x,d} + k\tau_{y,d} - cl\tau_{z,d} + 4\rho_m ckl\Omega_{4,\max}^2). \end{cases} \tag{3.55}$$

For vehicles with unidirectional thrust vector, the resulting thrust is only feasible if

$$0 < T_{\max} < \sum_{i=1}^{N} T_{i,\max}, \tag{3.56}$$

because with zero cumulative thrust, no torques can be produced. With bidirectional thrust, the admissible region of $T_{\max}$ increases to

$$\sum_{i=1}^{N} T_i(\overline{\Omega}_i) < T_{\max} < \sum_{i=1}^{N} T_i(\hat{\Omega}_i), \tag{3.57}$$

where $\hat{\Omega}_i$ denotes the upper (positive) limit and $\overline{\Omega}_i$ denotes the lower (negative) limit of the rotor speeds, with $\overline{\Omega}_i = -\hat{\Omega}_i$ for symmetric and $\overline{\Omega}_i \neq -\hat{\Omega}_i$ for unsymmetric bidirectional thrust. If the above conditions are fulfilled, $\boldsymbol{\Omega}$ is reallocated using $T_d = T_{\max}$. Otherwise, all rotor speeds need to be clipped at the maximum rotor speeds, which means that none of the desired control inputs are achieved.

Extension to polynomial propulsion models

It holds for all nonlinear propulsion models, that at any given (i.e. estimated or measured) rotor speed Ω_i a linear relation between thrust and torque of the form

$$\tau_i = \kappa_i(\Omega_i)T_i \tag{3.58}$$

can be computed [118]. Hence, it follows for all multicopter configurations that

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{pmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{n}_1 & \cdots & \boldsymbol{n}_N \\ \boldsymbol{S}(\boldsymbol{r}_1)\boldsymbol{n}_1 + P_1\boldsymbol{n}_1\kappa_1(\Omega_1) & \cdots & \boldsymbol{S}(\boldsymbol{r}_N)\boldsymbol{n}_N + P_N\boldsymbol{n}_N\kappa_N(\Omega_N) \end{bmatrix}}_{\boldsymbol{C}^*} \begin{pmatrix} T_1 \\ \vdots \\ T_N \end{pmatrix}. \tag{3.59}$$

It is straightforward to show, that equation (3.59) may be inverted analytically (or symbolically) for the multicopter configurations treated in this work. Then the exact same approach as above can be applied to allocate the maximum admissible inputs at the current rotor speeds Ω_i, where $i \in \{1, \dots, N\}$.

Proof of the direct input allocation approach

The feasibility of the direct maximum input allocation approach is proven in the following.

Theorem 3.1. *Without loss of generality, consider $N_{\text{sat}} \leq N$ saturated motors. Let motor j be the motor that violates the rotor speed constraint the most. If (3.53) and (3.55) for motor j, or appropriate variations thereof, yield a feasible result for $\tau_{z,\max}$ or $T_{\max}$ at $\Omega_{j,\max}$, then scaling down the control inputs $\tau_{z,d}$ or T_d to $\tau_{z,\max}$ and $T_{\max}$, respectively, leads for motors $1 \dots N$ to rotor speeds within the maximum rotor speed constraints.*

Proof. For motor j it holds that $T_j > T_i$, $i \in \{1, \dots, N\} \setminus \{j\}$. If the cumulative thrust is scaled down via $T_d^* = \sigma_j T_d = T_{\max}$, it can be seen from (3.59) that still $\sigma_j T_j > \sigma_j T_i$. Recall that the scaling σ_j is computed, such that $\Omega_j = \Omega_{j,\max}$. Therefore, $\Omega_i < \Omega_{j,\max}$, $i \in \{1, \dots, N\} \setminus \{j\}$, because the polynomials (3.11) and (3.12) are strictly monotonically decreasing for decreasing rotor speeds. The argumentation for scaling the control torque $\tau_{z,d}$ follows analogously due to the linearity of (3.58) and, hence, the linear relation between τ_z and $T_1 \dots T_N$ in (3.59). □

3.4.7 Iterative saturation handling

Directly allocating a maximum input as described above may lead to discontinuities in the control input. This can be overcome by iteratively increasing or decreasing the control input until the constraints are met. The iterative approach can be summarized as follows: In every iteration step, the desired thrust and torques are mapped to rotor speeds using any of the unconstrained control allocation procedures presented above (for example (3.38), the iterative approach (3.44), or the linearized result (3.52)). The obtained speeds are checked for consistency with the maximum possible rotor speeds. If the allocated rotor speeds are infeasible, thrust and torques are decreased iteratively.

Pseudo code of the proposed iterative saturation handling procedure is provided in Algorithm 3.1. Therein, $\boldsymbol{\Omega}(t-1)$ are the measured rotor speeds, which ideally should be equal to the previously commanded rotor speeds, and μ and ε are tuning parameters. Those parameters are the step size of the iterative algorithm and determine how fast it will converge. The algorithm lines 2 to 7 reduce the yaw torque by a fixed amount until the maximum rotor speeds are not exceeded any more. If line 8 of the algorithm is reached, it means that the yaw torque is reduced to zero, but the rotor speeds are still saturated. Then, the thrust is decreased until it reaches the lower boundary ε. If that happens, it means that torques are still to high to be allocated with the desired thrust. Then, the thrust is reset, the roll and pitch torques are decreased by one increment μ, and the thrust is decreased iteratively again. This loop continues until the rotor speed limits are met. According to Theorem 3.1 in Section 3.4.6, this will eventually lead to $\boldsymbol{\Omega} < \boldsymbol{\Omega}_{\max}$ and will not end up in an infinite loop. In order to decrease the running time of Algorithm 3.1, τ_z and T can be initialized using the solutions of the direct input allocation approach from Section 3.4.6, since these represent the maximum attainable values. The maximum rotor speeds can be adjusted according to the momentarily available battery voltage U_{bat} via $\Omega_{\max} = K_v U_{bat}$ (cf. the motor model in Section 3.2.2). Note that the case $\boldsymbol{\Omega} < \boldsymbol{\Omega}_{\min}$ is not shown for brevity. It can be realized

Algorithm 3.1: Iterative saturation handling

```
1  $\boldsymbol{\Omega} \leftarrow \textbf{allocate}(\boldsymbol{\Omega}(t-1), T, \boldsymbol{\tau})$        Implements e.g. (3.38), (3.44), or (3.52).
2  if $\boldsymbol{\Omega} > \boldsymbol{\Omega}_{\max}$ then
3      while $|\tau_z| > 0$ do
4          $\tau_z \leftarrow \tau_z - \mathrm{sign}(\tau_z)\mu$
5          $\boldsymbol{\Omega} \leftarrow \textbf{allocate}(\boldsymbol{\Omega}(t-1), T, \boldsymbol{\tau})$
6          if $\boldsymbol{\Omega} < \boldsymbol{\Omega}_{\max}$ then
7              return
8      $T^* \leftarrow T$
9      while $\boldsymbol{\Omega} > \boldsymbol{\Omega}_{\max}$ do
10         if $|T| < \varepsilon$ then
11             $T \leftarrow T^*$
12             $\tau_x \leftarrow \tau_x - \mathrm{sign}(\tau_x)\mu$
13             $\tau_y \leftarrow \tau_y - \mathrm{sign}(\tau_y)\mu$
14         $T = T - \mathrm{sign}(T^*)\varepsilon$
15         $\boldsymbol{\Omega} \leftarrow \textbf{allocate}(\boldsymbol{\Omega}(t-1), T, \boldsymbol{\tau})$
```

analogously by including the lower limits $\mathbf{\Omega}_{\min}$ of $\mathbf{\Omega}$ and by iteratively increasing yaw torque or thrust until the constraints are satisfied.

3.5 Adaptive control approach

As shown in Section 3.2.2, the propulsion forces (3.10) of a rotorcraft linearly depend on the air density. This influence is often disregarded in the control of flying robots. Usually, the air density is lumped in the constant thrust and torque coefficients. However, in practice, it changes depending on local weather and altitude, i.e. depending on the atmospheric parameters pressure, temperature, and humidity (cf. the atmosphere model in Section 3.2.4). Position and attitude control and all additional components, like disturbance observers or external wrench estimators (see Section 3.1.2), rely on an estimate of the applied forces. Uncertainty in the propulsion model will inevitably degrade control performance and add to the estimated external disturbances.

The variation of the air density is often implicitly considered via an integral term in the altitude (height) controller [94]. However, the integral term has no physical justification and its convergence and stability is difficult to guarantee. Alternatively, model uncertainties may be estimated by an external wrench estimator [112], but then it is not possible to distinguish between uncertainties in the propulsion model and external disturbances.

Sensors that measure absolute air pressure, air temperature, and relative humidity are readily available, light weight, and easy to integrate. However, measurements acquired with sensors on board the flying robot contain large uncertainties. Barometric sensors are sensitive to wind and subject to large drifts. Temperature sensors are affected by heat dissipation from surrounding electrical components. In addition to that, estimating the propulsion forces based on sensor measurements is prone to sensor failures and provides no guarantee for convergence of position and attitude control.

In order to account for the uncertainty of thrust and torque of a multirotor due to the aforementioned variation in air density, an adaptive control approach for simultaneous position tracking and air density estimation[7] is presented here [209]. It does not require additional sensor measurements and only relies on the desired trajectory and on an accurate pose estimate of the flying robot (namely height and vertical velocity), which is available from a downward facing range sensor, from fusing GPS with IMU measurements, or from a VIO [217]. In addition to the air density, the controller also adapts to changing mass, e.g. if a payload is collected or dropped. The adaptive controller clearly improves position tracking precision and the estimated air density leads to an augmented external wrench estimator with improved accuracy. Moreover, the estimate of the air density may be used for meteorological measurements, for power estimation of wind energy plants, or, if combined with accurate sensor measurements, to identify the thrust and torque coefficients of a flying robot. It utilizes a reference model of the flying robot and is therefore an MRAC method [86].

[7] Patent granted, see [222].

3.5.1 Adaptive controller design

Consider the vertical translational dynamics in the inertial frame without external forces

$$m\ddot{z} = -mg + f_z, \tag{3.60}$$

For a multicopter with N parallel thrust vectors, the vertical force f_z follows as

$$f_z = \boldsymbol{e}_3^T \boldsymbol{R}_{iu} \boldsymbol{e}_3 \sum_{i=1}^{N} T_i = \rho c \boldsymbol{e}_3^T \boldsymbol{R}_{iu} \boldsymbol{e}_3 \sum_{i=1}^{N} \Omega_i^2. \tag{3.61}$$

Note that the linearity in ρ and c follows analogously for the lateral and longitudinal forces as well as for arbitrary rotor configurations. For instance, the vertical force of the coaxial hexacopter is given by

$$f_z = \rho \boldsymbol{e}_3^T \boldsymbol{R}_{iu} \boldsymbol{e}_3 \left(c_u \sum_{i=1}^{3} \Omega_i^2 + c_l \sum_{i=4}^{6} \Omega_i^2 \right). \tag{3.62}$$

Therein, c_u and c_l are the thrust coefficients of upper and lower propeller, respectively. Inserting the relation (3.39) in (3.61) yields an expression for the vertical force

$$f_z = \frac{\rho}{\rho_m} f_{z,d}. \tag{3.63}$$

Let the control input be $u = f_{z,d}$. Then, inserting (3.63) in (3.60) gives[8]

$$\frac{\rho_m}{\rho} m\ddot{z} = -\frac{\rho_m}{\rho} mg + u. \tag{3.64}$$

A controller similar to ([86], pp. 317) that adaptively accounts for the mismatch between ρ_m and ρ is given by

$$u = (1+\varepsilon) m (\ddot{z}_d - 2\lambda\dot{\tilde{z}} - \lambda^2 \tilde{z} + g). \tag{3.65}$$

Therein, $\tilde{z} = z - z_d$ and $\dot{\tilde{z}} = \dot{z} - \dot{z}_d$ are position and velocity error, respectively, $\lambda > 0$ is the controller gain[9], and ε is a parameter introduced for adaptation. It should result in $(1+\varepsilon) \to \frac{\rho_m}{\rho}$. Using $\overline{m} = \frac{\rho_m}{\rho} m$ for brevity, (3.64) is expanded on both sides, such that

$$\overline{m}\ddot{z} - \overline{m}\ddot{z}_d + 2\overline{m}\lambda\dot{\tilde{z}} + \overline{m}\lambda^2\tilde{z} = -\overline{m}(\ddot{z}_d - 2\lambda\dot{\tilde{z}} - \lambda^2\tilde{z} + g) + u. \tag{3.66}$$

Substituting (3.65), the combined tracking error $s = \dot{\tilde{z}} + \lambda\tilde{z}$, and $\nu = \ddot{z}_d - 2\lambda\dot{\tilde{z}} - \lambda^2\tilde{z} + g$ in equation (3.66) leads to

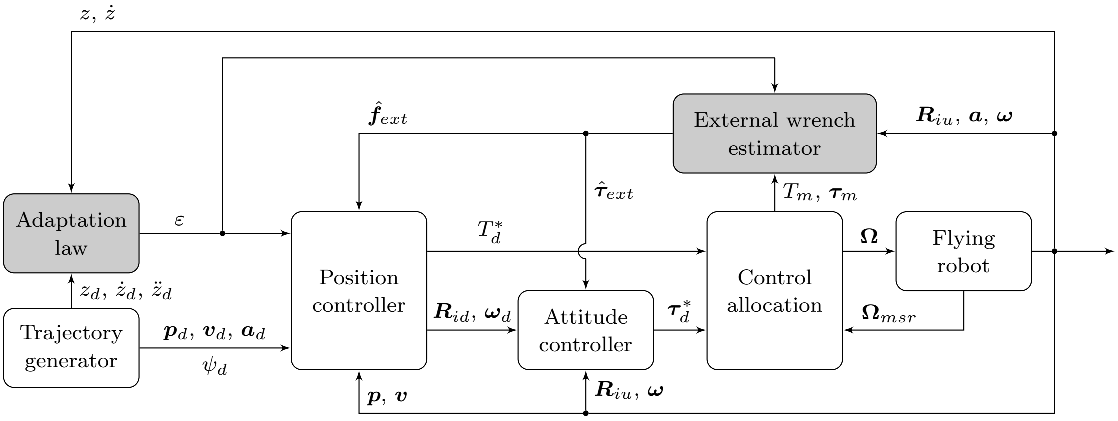
$$\overline{m}(\dot{s} + \lambda s) = \tilde{m}\nu, \tag{3.67}$$

where $\tilde{m} = \hat{m} - \overline{m}$ is the parameter error and $\hat{m} = (1+\varepsilon)m$ will be referred to as the effective mass. For the derivative w.r.t. time of the parameter error it holds, because of $\overline{m} = \text{const.}$, that

$$\dot{\tilde{m}} = \dot{\hat{m}} = \dot{\varepsilon} m. \tag{3.68}$$

[8] Examining equation (3.64) more closely, one realizes that $\ddot{z} + g$ is measured by the accelerometer and, hence, a direct observer on $\frac{\rho_m}{\rho}$ could be created. However, this would not ensure simultaneous convergence of the tracking error.

[9] Note that a damping ratio of 1 is used in order to reduce the number of design parameters.



z, $\dot{z}$
Adaptation law
ε
z_d, $\dot{z}_d$, $\ddot{z}_d$
Trajectory generator
$\boldsymbol{p}_d$, $\boldsymbol{v}_d$, $\boldsymbol{a}_d$
ψ_d
$\hat{\boldsymbol{f}}_{ext}$
Position controller
$\boldsymbol{p}$, $\boldsymbol{v}$
T_d^*
$\boldsymbol{R}_{id}$, $\boldsymbol{\omega}_d$
$\hat{\boldsymbol{\tau}}_{ext}$
Attitude controller
$\boldsymbol{R}_{iu}$, $\boldsymbol{\omega}$
$\boldsymbol{\tau}_d^*$
External wrench estimator
$\boldsymbol{R}_{iu}$, $\boldsymbol{a}$, $\boldsymbol{\omega}$
T_m, $\boldsymbol{\tau}_m$
Control allocation
$\boldsymbol{\Omega}$
$\boldsymbol{\Omega}_{msr}$
Flying robot

Figure 3.16: Schematics of the adaptive control approach combined with an external wrench estimator.

An adaptation law that achieves $\hat{m} \to \overline{m}$, and hence $(1+\varepsilon) \to \frac{\rho_m}{\rho}$, is obtained using the Lyapunov function

$$V = \frac{1}{2}\overline{m}s^2 + \frac{1}{2\gamma}\tilde{m}^2, \tag{3.69}$$

which is lower bounded by $\overline{m} \geq 0$, $s^2 \geq 0$, and $\tilde{m}^2 \geq 0$ and positive definite for a positive design parameter γ. Its first derivative w.r.t. time is

$$\dot{V} = \overline{m}s\dot{s} + \frac{1}{\gamma}\tilde{m}\dot{\hat{m}} = s(\tilde{m}\nu - \overline{m}\lambda s) + \frac{1}{\gamma}\tilde{m}\dot{\hat{m}} \tag{3.70}$$

$$= -\overline{m}\lambda s^2 + \tilde{m}s\nu + \frac{1}{\gamma}\tilde{m}\dot{\hat{m}}. \tag{3.71}$$

Using Barbalat's lemma ([86], p. 123, [87], p. 323), stability and convergence of tracking error $s \to 0$ may be concluded, if $\dot{V} \leq 0$ and $\ddot{V}$ is bounded. However, convergence of the parameter error $\tilde{m}$ can only be guaranteed under persistent excitation, as shown for a similar adaptive control law in ([87], pp. 327). Thus, the adaptation law is found as

$$\dot{\hat{m}} = -\gamma s\nu = -\gamma(\dot{\tilde{z}} + \lambda\tilde{z})(\ddot{z}_d - 2\lambda\dot{\tilde{z}} - \lambda^2\tilde{z} + g), \tag{3.72}$$

for which it holds that $\dot{\varepsilon} = \frac{\dot{\hat{m}}}{m}$, $\dot{V} = -\overline{m}\lambda s^2 \leq 0$, and $\ddot{V} = -2\overline{m}\lambda s\dot{s}$. The latter is bounded, because of the boundedness of $\overline{m}$, s, and ε. The adaptive gain ε is bounded in practice by the maximum and minimum thrust of the flying robot. Typically, it may be constrained artificially in the range $-1.0 < \varepsilon \leq 1.0$.

An overview of the proposed adaptive control approach is depicted in Figure 3.16. For adaptive position control in three degrees of freedom $\boldsymbol{p} = (x \quad y \quad z)^T$ the inner-loop attitude controller (3.18) and the outer-loop position controller

$$\boldsymbol{f}_d = m(1+\varepsilon)\left(\ddot{\boldsymbol{p}}_d - \boldsymbol{K}_d\dot{\tilde{\boldsymbol{p}}} - \boldsymbol{K}_p\tilde{\boldsymbol{p}} + g\boldsymbol{e}_3\right) \tag{3.73}$$

are utilized with $\tilde{\boldsymbol{p}} = \boldsymbol{p} - \boldsymbol{p}_d$. For the positive definite gain matrices it follows from (3.65) that $\boldsymbol{K}_d = \mathrm{diag}(\lambda_x, \lambda_y, 2\lambda_z)$ and $\boldsymbol{K}_p = \mathrm{diag}(a\lambda_x, b\lambda_y, \lambda_z^2)$ with $a > 0$ and $b > 0$ being design parameters. Starting from $\varepsilon(t=0) = 0$, the adaptation law (3.72) is integrated numerically during runtime to obtain ε, which then enters the controller (3.73). Finally, the desired thrust T_d is computed using (3.27), the desired orientation $\boldsymbol{R}_{id}$, i.e. the rotation between $\boldsymbol{e}_3$ and $\boldsymbol{f}_d$, is calculated via (3.28), and the desired angular rate $\boldsymbol{\omega}_d$ is obtained numerically.

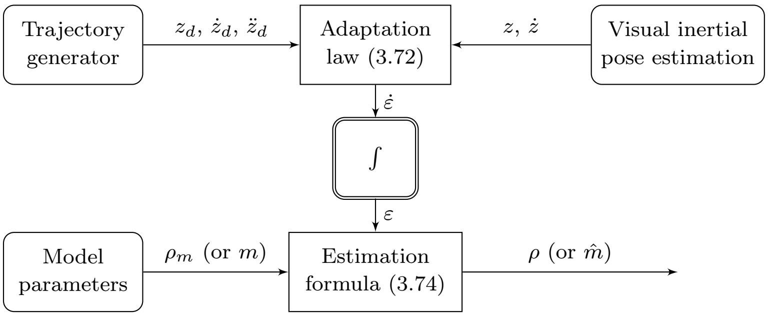


Figure 3.17: Illustration of online air density or payload estimation.

3.5.2 Air density estimation

The air density ρ can be estimated using the adaptive control approach as depicted in Figure 3.17. The estimate of the air density is computed as

$$\hat{\rho} = \frac{1}{1+\varepsilon}\rho_m = \frac{m}{\hat{m}}\rho_m. \tag{3.74}$$

Up to now, the mass m is considered constant and known. Due to the definitions $\overline{m} = \frac{\rho_m}{\rho} m$ and $\hat{m} = (1+\varepsilon)m$, it becomes clear that $\hat{m} \to \overline{m}$ allows to either estimate ρ, if ρ_m and m are given, or m, if ρ_m and ρ are known, e.g. calculated using sensor measurements and the atmosphere model from Section 3.2.4. Assuming that the air density estimate is converged, its current value may be used to estimate a collected or dropped payload Δm via $\Delta m = \varepsilon m - \hat{m}_0$. Therein, m is the known take-off mass and $\hat{m}_0$ is the offset from zero of $\hat{m}$ at the start of the payload estimation.

As stated in ([86], pp. 331), convergence of the estimated parameters can only be guaranteed under persistent excitation. However, it is found in the experiments in Section 3.6.2, that the estimates of ρ and m converge during hover flight. Moreover, it is suspected that the flight trajectories provide sufficient excitation for the parameters to converge. Note that stability and accurate tracking are independent from parameter convergence and ensured by the controller (3.65) even if neither m nor ρ is known exactly.

By design, the adaptation law is applicable to multirotor control without changing the control allocation. Though, to estimate the air density correctly, the coefficients c and k in (3.10) have to be identified experimentally beforehand, as shown above on Figure 3.6.

In practice, the adaptation law may be activated (e.g. through a service call)

- once, to estimate the air density, or
- if a payload is collected or dropped,

and deactivated afterwards in order to avoid adaptation to external disturbances. To realize rejection of external disturbances, an external wrench estimator can be included in Figure 3.16 in addition to the adaptive controller. In Section 3.5.4, it is shown how the estimated air density may be incorporated in order to increase the accuracy of the estimated external wrench.

3.5.3 Generalization and passivity analysis

Assume time-scale separation between rotational and translational dynamics, i.e. that the attitude control is sufficiently fast. Relation (3.64) can be extended to three degrees of freedom such that

$$\overline{m}(\ddot{\boldsymbol{p}} + g\boldsymbol{e}_3) = \boldsymbol{u} + \overline{\boldsymbol{f}}_{ext}, \tag{3.75}$$

where $\overline{m} = \frac{\rho_m}{\rho} m$, $\boldsymbol{u} = \frac{\rho_m}{\rho} \boldsymbol{f}$, and $\overline{\boldsymbol{f}}_{ext} = \frac{\rho_m}{\rho} \boldsymbol{f}_{ext}$. Let m_m be the model mass and $\hat{m} = (1+\varepsilon)m_m$ be the effective mass. Then an adaptive controller for all three degrees of freedom is given by[10]

$$\boldsymbol{u} = \hat{m}(\ddot{\boldsymbol{p}}_d - (\boldsymbol{P} + \boldsymbol{K})\dot{\tilde{\boldsymbol{p}}} - \boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{p}} + g\boldsymbol{e}_3) = \hat{m}\boldsymbol{\nu}, \tag{3.76}$$

which is a general form of the control law (3.65). Adding $-\overline{m}\boldsymbol{\nu}$ on both sides of (3.75) and introducing the new error signal $\boldsymbol{s} = \dot{\tilde{\boldsymbol{p}}} + \boldsymbol{P}\tilde{\boldsymbol{p}}$ yields

$$\overline{m}\dot{\boldsymbol{s}} = -\overline{m}\boldsymbol{K}\boldsymbol{s} + \tilde{m}\boldsymbol{\nu} + \overline{\boldsymbol{f}}_{ext}. \tag{3.77}$$

Consider a Lyapunov function candidate similar to [158]

$$V = \frac{1}{2}(\overline{m}\boldsymbol{s}^T\boldsymbol{s} + \tilde{\boldsymbol{p}}^T\boldsymbol{Q}\tilde{\boldsymbol{p}} + \frac{1}{\gamma}\tilde{m}^2). \tag{3.78}$$

Its first derivative with respect to time is

$$\dot{V} = \boldsymbol{s}^T(-\overline{m}\boldsymbol{K}\boldsymbol{s} + \tilde{m}\boldsymbol{\nu} + \overline{\boldsymbol{f}}_{ext}) + \tilde{\boldsymbol{p}}^T\boldsymbol{Q}(\boldsymbol{s} - \boldsymbol{P}\tilde{\boldsymbol{p}}) + \frac{1}{\gamma}\tilde{m}\dot{\tilde{m}}. \tag{3.79}$$

Note that $\dot{\tilde{m}} = \dot{\hat{m}}$ since $\overline{m} = \text{const.}$ by assumption. Inserting the adaptation law $\dot{\hat{m}} = -\gamma\boldsymbol{s}^T\boldsymbol{\nu}$ and the definition of the error signal $\boldsymbol{s}$ leads to

$$\begin{aligned}
\dot{V} &= -(\dot{\tilde{\boldsymbol{p}}}^T + \tilde{\boldsymbol{p}}^T\boldsymbol{P}^T)\overline{m}\boldsymbol{K}(\dot{\tilde{\boldsymbol{p}}} + \boldsymbol{P}\tilde{\boldsymbol{p}}) + \boldsymbol{s}^T\overline{\boldsymbol{f}}_{ext} + \tilde{\boldsymbol{p}}^T\boldsymbol{Q}(\dot{\tilde{\boldsymbol{p}}} + \boldsymbol{P}\tilde{\boldsymbol{p}}) - \tilde{\boldsymbol{p}}^T\boldsymbol{Q}\boldsymbol{P}\tilde{\boldsymbol{p}} && (3.80)\\
&= -\overline{m}\dot{\tilde{\boldsymbol{p}}}^T\boldsymbol{K}\dot{\tilde{\boldsymbol{p}}} - \overline{m}\tilde{\boldsymbol{p}}^T\boldsymbol{P}^T\boldsymbol{K}\dot{\tilde{\boldsymbol{p}}} - \overline{m}\dot{\tilde{\boldsymbol{p}}}^T\boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{p}} - \overline{m}\tilde{\boldsymbol{p}}^T\boldsymbol{P}^T\boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{p}} + \boldsymbol{s}^T\overline{\boldsymbol{f}}_{ext} + \tilde{\boldsymbol{p}}^T\boldsymbol{Q}\dot{\tilde{\boldsymbol{p}}}. && (3.81)
\end{aligned}$$

Assume $\boldsymbol{P} = \boldsymbol{P}^T$, $\boldsymbol{K} = \boldsymbol{K}^T$, $\boldsymbol{P} > 0$, $\boldsymbol{K} > 0$, and define $\boldsymbol{Q} = 2\overline{m}\boldsymbol{P}^T\boldsymbol{K}$.[11] Then it follows, that

$$\dot{V} = -\overline{m}\dot{\tilde{\boldsymbol{p}}}^T\boldsymbol{K}\dot{\tilde{\boldsymbol{p}}} - \overline{m}\tilde{\boldsymbol{p}}^T\boldsymbol{P}^T\boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{p}} + \boldsymbol{s}^T\overline{\boldsymbol{f}}_{ext} \leq \boldsymbol{s}^T\overline{\boldsymbol{f}}_{ext}, \tag{3.82}$$

wherein the congruent transformation $\boldsymbol{P}^T\boldsymbol{K}\boldsymbol{P}$ preserves the positive semidefiniteness of $\boldsymbol{P}$ and $\boldsymbol{K}$ [159]. Hence, system (3.75) under the controller (3.76) is passive w.r.t. the power port $\boldsymbol{s}^T\overline{\boldsymbol{f}}_{ext}$. In addition, this shows global, asymptotic convergence of the tracking error for the entire translational closed-loop dynamics. The parameter error $\tilde{m}$ will converge for sufficient excitation of the system [86, 160]. Finally, from the result (3.82) Lyapunov stability can be concluded.

[10] Thanks to Manuel Keppler for pointing me to this control law, to the corresponding Lyapunov function candidate, and for providing a sketch of the passivity proof presented here.

[11] Note that the product of two symmetric and positive semidefinite matrices is not necessarily positive semidefinite. However, it can be shown that it is positive semidefinite, if the product of the two matrices is also symmetric [90].

3.5.4 Augmented external wrench estimator

The hybrid external wrench estimator presented in [112] is given by

$$\begin{pmatrix} \hat{\boldsymbol{f}}_{ext} \\ \hat{\boldsymbol{\tau}}_{ext} \end{pmatrix} = \begin{pmatrix} \int_0^t \boldsymbol{K}_f \left(m\ddot{\boldsymbol{p}} - \boldsymbol{f} - \hat{\boldsymbol{f}}_{ext} \right) \mathrm{d}t \\ \boldsymbol{K}_\tau \left(\boldsymbol{J}\boldsymbol{\omega} - \int_0^t \left(\boldsymbol{\tau} + \boldsymbol{S}(\boldsymbol{J}\boldsymbol{\omega})\boldsymbol{\omega} + \hat{\boldsymbol{\tau}}_{ext} \right) \mathrm{d}t \right) \end{pmatrix}, \tag{3.83}$$

where $\boldsymbol{K}_f$ and $\boldsymbol{K}_\tau$ are the lowpass filter gains. The external force $\hat{\boldsymbol{f}}_{ext}$ is estimated using the measured and lowpass filtered linear acceleration $\ddot{\boldsymbol{p}}$ and the known mass m. The external torque $\hat{\boldsymbol{\tau}}_{ext}$ is estimated via the angular momentum using the measured angular velocity $\boldsymbol{\omega}$ and the known inertia $\boldsymbol{J}$. The estimator (3.83) is shown to be input-to-state stable in [112]. It contains the applied wrench $(\boldsymbol{f} \quad \boldsymbol{\tau})^T$, which is unknown on a flying robot. Or more specifically, in (3.83) it is assumed that $\boldsymbol{f}$ and $\boldsymbol{\tau}$ are evaluated using the known air density.

The adaptive control approach allows to deduce the applied wrench via relation (3.39) such that

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{pmatrix} = \frac{\hat{\rho}}{\rho_m} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{\tau}_d \end{pmatrix} = \frac{1}{1+\varepsilon} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{\tau}_d \end{pmatrix} \tag{3.84}$$

where $\boldsymbol{u}$ and $\boldsymbol{\tau_d}$ follow from (3.76) and (3.18), respectively, and $\hat{\rho}$ is computed using (3.74). Alternatively, if rotor speed measurements $\boldsymbol{\Omega}_{msr}$ are available, the applied thrust and torque can be obtained from

$$\begin{pmatrix} T \\ \boldsymbol{\tau} \end{pmatrix} = \hat{\rho} \boldsymbol{B} \boldsymbol{\varpi}(\boldsymbol{\Omega}_{msr}^2). \tag{3.85}$$

Disturbance compensation is achieved, if the flying robot completely counteracts the external wrench. Additionally, the uncertainty in the applied wrench due to varying air density needs be considered. An augmented control input

$$\boldsymbol{u}^* = \boldsymbol{u} - (1+\varepsilon)\hat{\boldsymbol{f}}_{ext}, \tag{3.86}$$

$$\boldsymbol{\tau}_d^* = \boldsymbol{\tau}_d - (1+\varepsilon)\hat{\boldsymbol{\tau}}_{ext}, \tag{3.87}$$

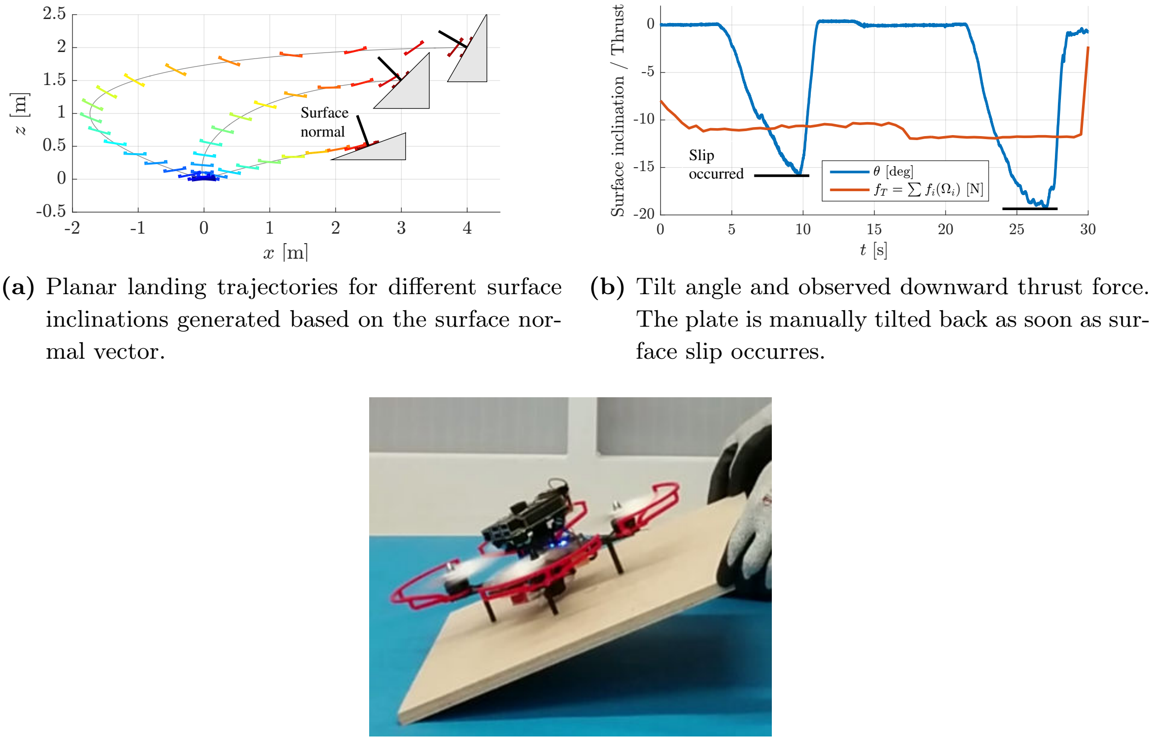
is proposed, with $\boldsymbol{u}$ and $\boldsymbol{\tau}_d$ as defined in (3.76) and (3.18), respectively. Now substitute $\boldsymbol{u}$ and $\boldsymbol{\tau}_d$ in (3.84) with $\boldsymbol{u}^*$ and $\boldsymbol{\tau}_d^*$ from (3.86) and (3.87), respectively. The applied force thus becomes $\boldsymbol{f} = m\boldsymbol{\nu} + \hat{\boldsymbol{f}}_{ext}$, i.e. the uncertainty cancels out. This holds for the applied torque analogously. The desired thrust $T_d^* = \boldsymbol{u}^{*T}\boldsymbol{R}_{ib}\boldsymbol{e}_3$ and the desired torque $\boldsymbol{\tau}_d^*$ are then mapped to desired rotor speeds via (3.38) (see also Figure 3.16).

3.6 Applications

In this section, applications of the control methods for flying robots presented in this chapter, specifically considering bidirectional thrust (Section 3.3.4) and adaptive control (Section 3.5), are discussed. Section 3.6.1 provides an evaluation of bidirectional thrust in simulations and experiments. It is shown that the available downward thrust may be utilized to realize safe landing on inclined surfaces. Furthermore, an efficient method to generate flight trajectories is presented. To the best of the author's knowledge, the conducted experiments include the first autonomously performed transitions from upright to inverted flight of a quadrocopter with fixed-pitch propellers but bidirectional thrust capabilities. In Section 3.6.2, the adaptive control approach and the air density estimation are evaluated in simulations and indoor as well as outdoor experiments with the quadrocopter Sparrow (Figure 3.13a) and the coaxial hexacopter Ardea (Figure 3.13b).

3.6.1 Evaluation of bidirectional thrust

Bidirectional thrust on a multirotor aerial vehicle can be applied to realize for example agile aerobatic maneuevers, rejection of large external disturbances, safe landing on inclined or moving surfaces by increasing the contact pressure in order to stay in the friction cone (see Figure 3.18), or inverted flight as shown in Figure 3.10. The latter is especially useful if different sensors are attached on top and bottom of the flying robot. The two applications slip reduction and flip from upright to inverted flight are treated in the following. For the



(a) Planar landing trajectories for different surface inclinations generated based on the surface normal vector.

(b) Tilt angle and observed downward thrust force. The plate is manually tilted back as soon as surface slip occurres.

(c) Quadrocopter using downward thrust on a tilted wooden plate.

Figure 3.18: Test with downward thrust on an inclined surface: (a) depicts suitable trajectories for landing on surfaces with different inclination angles. The plate shown in (c) is tilted until the quadrocopter starts to slip. Then the thrust is increased and the plate is tilted again. (b) illustrates the observed contact force, which results in a larger maximum inclination angle. The achievable tilt angle also depends on the surface and landing gear material. Copyright © 2018 IEEE [215].

experiments, the fully symmetric Graupner 3D propellers (see Section 3.2.5) are selected due to their nearly equal maximum thrust in both directions.

Landing on inclined surfaces with slip reduction

Landing on inclined surfaces requires to determine the surface normal vector and the generation of a suitable trajectory [219, 161]. Due to its computational efficiency, the trajectory generation method presented in [162] is adapted.

Given the surface normal vector estimated in the body frame u, the target roll angle φ_f and the target pitch angle θ_f can be derived. Due to the differential flatness of the problem and with $\ddot{z}_f = 0$, it follows for the target accelerations $\ddot{x}_f = \tan(\theta_f)g$ and $\ddot{y}_f = \tan(\varphi_f)g$. Some examples for different surface inclinations are shown in Figure 3.18.

Once the flying robot is landed, it can switch to downward thrust and increase the contact pressure to stay within the friction cone. The contact force can be estimated and controlled using the measured rotor speed and the rotor speed to thrust mapping identified in Section 3.2.5. Figure 3.18 depicts the quadrocopter in an experiment with changing surface inclination and estimated contact force. The surface is tilted until the quadrocopter starts to slide and then tilted back immediately. Increasing downward thrust clearly increases the maximum inclination angle θ (cf. Figure 3.18).

Flip trajectory from upright to inverted flight

As mentioned before, switching from upright to inverted flight also requires a feasible trajectory, that satisfies the maximum thrust forces of the flying robot. It is assumed that every multicopter configuration can be transformed to the planar model in the x/z - plane

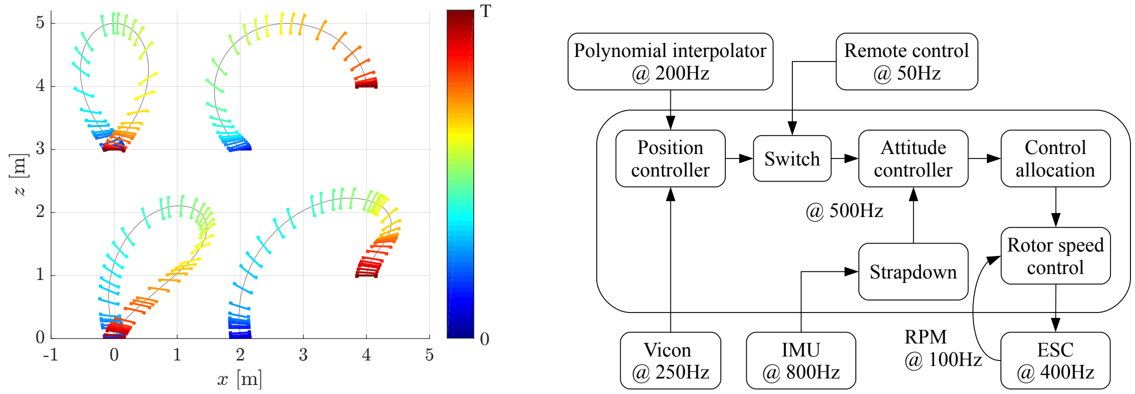
$$\begin{aligned}
\ddot{x} &= \frac{1}{m}(f_1 + f_2)\sin(\theta),\\
\ddot{z} &= \frac{1}{m}(f_1 + f_2)\cos(\theta) - g,\\
\ddot{\theta} &= \frac{1}{J_{yy}}(f_1 - f_2)l,
\end{aligned} \tag{3.88}$$

where θ is the pitch angle and all rotor forces are summarized in f_1 and f_2, respectively. Then, an optimal control problem may be formulated as follows:

$$\min_{t_d, f_1, f_2} \; \Gamma = t_d + f_1^2 + f_2^2, \;\; \text{s.t.} \begin{cases} f_{\min} \leq f_1 \leq f_{\max}, \\ f_{\min} \leq f_2 \leq f_{\max}, \\ \quad |\dot{\theta}| \leq \dot{\theta}_{\max}, \end{cases} \tag{3.89}$$

where t_d is the duration of the trajectory. The yaw angle ψ is neglected and assumed constant throughout the trajectory.

The initial orientation is $\theta_0 = 0$ and the target orientation is $\theta_f = \pi$, or vice versa, while the desired start and target position can be chosen depending on the application. Velocities and accelerations at start and target are assumed to be zero. It can be seen from the



(a) Difference between flip with symmetric (top left and right) and unsymmetric bidirectional thrust (bottom left and right).

(b) Structure of the implemented control architecture.

Figure 3.19: Flip trajectories computed using optimal control solver GPOPS (left) and implemented control architecture (right). Copyright © 2018 IEEE [215].

propeller identification in Section 3.2.5, that one has to distinguish between vehicles with symmetric ($f_{\max} = -f_{\min}$) and unsymmetric ($f_{\max} \neq -f_{\min}$) maximum bidirectional thrust. Obviously, in the planar example, the limit for inverted flight with unsymmetric bidirectional thrust is $f_{\min} = -\frac{1}{2}mg$. Figure 3.19a shows the difference between trajectories for symmetric and unsymmetric bidirectional thrust. All solutions of (3.89) are computed using the optimal control toolbox GPOPS [163]. The upper two trajectories in Figure 3.19a satisfy $f_{\max} = -f_{\min} = mg$, the lower two satisfy $f_{\max} = mg$ and $f_{\min} = -0.6mg$. The parameters of the considered quadrocopter are $m = 0.52\,\mathrm{kg}$, $J_{yy} = 0.003\,\mathrm{kg\,m^2}$, $l = 0.08\,\mathrm{m}$, and $\dot{\theta}_{\max} = 120\,^{\circ}\,\mathrm{s}^{-1}$.

To compute the flip trajectory for flight experiments online, again the polynomial approach by Mueller et al. [162] is used. The method is summarized briefly in Appendix A6. A trajectory consisting of two segments can be iteratively checked for maximum thrust and angular rate. For a 180 deg ($\phi_f = \pm\pi$) flip, the constraint $\phi_h = \pm\frac{\pi}{2}$ at height $z_h = h$ is introduced. It follows from

$$\cos(\phi_h) = \frac{\boldsymbol{f}^T \boldsymbol{e}_3}{||\boldsymbol{f}^T \boldsymbol{e}_3||} = \frac{\ddot{z}_h + g}{||\ddot{z}_h + g||} = 0, \tag{3.90}$$

that the acceleration at z_h must be $\ddot{z}_h = -g$. For a smooth trajectory, jerk continuity at the intersection of two segments is required. Therefore, a constraint on the durations of the two segments is obtained as

$$t_{d,1} = t_{d,2} = \sqrt{\frac{20h}{3g}}. \tag{3.91}$$

Note that actually the whole (position) trajectory has $\mathcal{C}^3$ continuity (up to jerk). The flip axis follows from the desired flight direction $\Delta\boldsymbol{p}$ as (cf. Appendix A7)

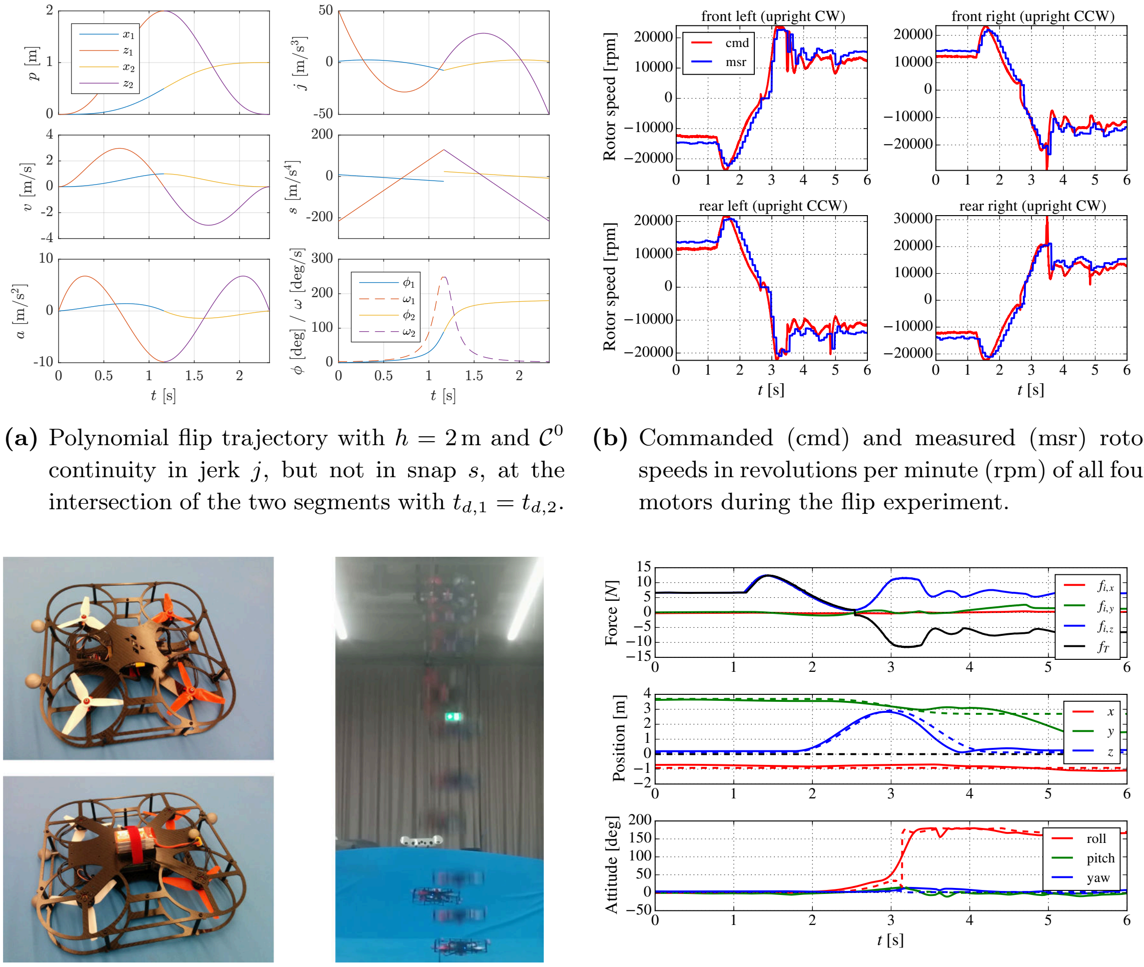
$$\boldsymbol{\epsilon} = \frac{\Delta\boldsymbol{p} \times \boldsymbol{e}_3}{||\Delta\boldsymbol{p} \times \boldsymbol{e}_3||}. \tag{3.92}$$

Figure 3.20(a) shows a polynomial flip trajectory generated using the described method.

The computed optimal trajectories could also be generalized, e.g. by using Dynamic Movement Primitives (DMPs) as done in [219, 220], or by solving the above optimization problem directly on the flight control computer, which is deemed realistic with modern optimal control solvers [164].

Flip experiment

Finally, the performance of the control approach is assessed in a flip experiment. It is carried out with the quadrocopter Sparrow shown in Figure 3.20c. The structure of the implemented control architecture is depicted in Figure 3.19b. A Vicon motion tracking system is used to estimate the pose of the flying robot in real-time. The results of the experiment and a picture



(a) Polynomial flip trajectory with $h = 2\,\mathrm{m}$ and $\mathcal{C}^0$ continuity in jerk j, but not in snap s, at the intersection of the two segments with $t_{d,1} = t_{d,2}$.

(b) Commanded (cmd) and measured (msr) rotor speeds in revolutions per minute (rpm) of all four motors during the flip experiment.

(c) Left top and bottom images: Second design iteration of the quadrocopter used for the experiments with 0.68 kg take-off mass. Right image: Picture sequence of a successful flip experiment with $h = 2.8\,\mathrm{m}$, $\dot{y}(h) = 1\,\frac{\mathrm{m}}{\mathrm{s}}$, and $\Delta y = 1\,\mathrm{m}$.

(d) Controller forces in the inertial frame $\boldsymbol{f}_i$ and thrust T with switch from positive to negative direction in the body frame during the flip experiment. Desired position and orientation are plotted dashed, measured position and orientation are depicted solid.

Figure 3.20: Results of a flip experiment in order to evaluate the performance of the developed controller. Copyright © 2018 IEEE [215].

of the quadrocopter performing a flip are shown in Figure 3.20. Desired height, orientation (c.f. Figure 3.20d), and rotor speeds (see Figure 3.20b) are tracked well. However, below 50% battery capacity the motors saturate and the quadrocopter is not able to complete the flip. This is because roll and pitch control authority is lost at full thrust. A video of the performed flight experiments can be found here: `https://youtu.be/yIAFX61MZMw`.

3.6.2 Evaluation of adaptive control

This section summarizes results of simulations using a quadrocopter model and experiments performed with the quadrocopter Sparrow ($m = 0.77\,\mathrm{kg}$) and with the hexacopter Ardea ($m = 2.74\,\mathrm{kg}$). The aim is to evaluate the performance of the presented adaptive control approach, the air density estimation, and the augmented external wrench observer. In summary, these points are examined:

- comparison of different controllers in simulation,
- trajectory tracking with air density estimation at two different altitudes with Ardea,
- set-point control with air density or payload estimation with Sparrow, and
- height tracking and air density estimation with transition between indoors and outdoors with Sparrow.

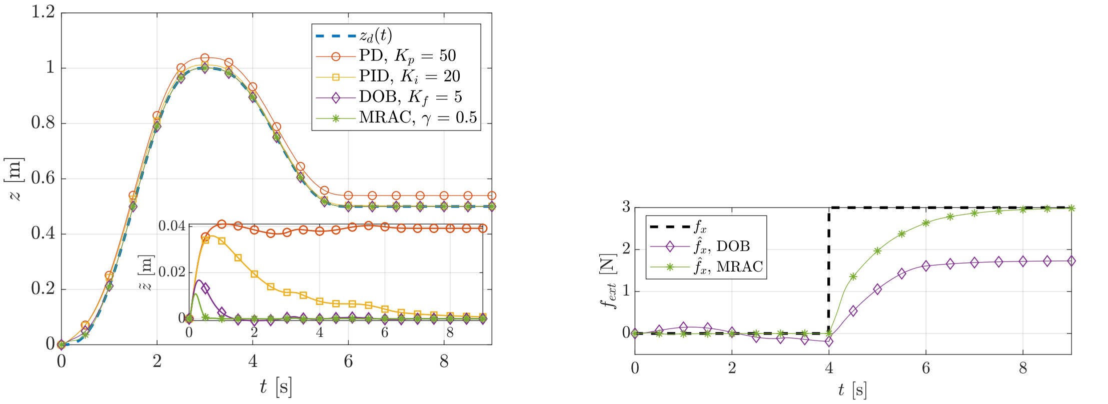
A video of the experiments conducted with Ardea on Mount Etna and with Sparrow at DLR is provided online (`https://youtu.be/HUCw56TkOEE`).

Comparison of different controllers in simulation

Proportional Derivative (PD) and Proportional Integral Derivative (PID) controllers are often used in practice and are therefore compared with the presented adaptive controller. Figure 3.21a shows a comparison of simulation results obtained with a PD controller, a PID controller using the integral of the position error, a PD controller which uses the estimated

Table 3.7: Quadrocopter simulation parameters.

m [kg]	$\boldsymbol{I}$ [kg m^2]	ρ_m $[\frac{\mathrm{kg}}{\mathrm{m}^3}]$	ρ $[\frac{\mathrm{kg}}{\mathrm{m}^3}]$
0.52	$\mathrm{diag}(2e^{-3}, 2e^{-3}, 2e^{-3})$	1.204	1.3
c_m [N s^2/rad^2]	c [N s^2/rad^2]	κ [m]	r [m]
0.0005	0.0008	0.1	0.11
$\boldsymbol{K}_d$	$\boldsymbol{K}_\omega$	k_R	$\boldsymbol{K}_f$
$2\sqrt{K_p}\boldsymbol{I}_{3\times3}$	$30\boldsymbol{I}_{3\times3}$	150	$5\boldsymbol{I}_{3\times3}$



(a) Reference trajectory vs. actual trajectory for the different controllers.

(b) External force estimated in the simulation.

Figure 3.21: Comparison in simulation of PD, PID, DOB, and MRAC using a quadrocopter model. Copyright © 2020 IEEE [209].

external wrench for disturbance observation and rejection (DOB), and the adaptive controller (MRAC) presented in this work. The PD, PID, and DOB controllers use $\varepsilon = 0$. For the simulation, a quadrocopter model and the parameters and controller gains summarized in Table 3.7 and in Figure 3.21a are considered. All gains are tuned manually and, where applicable, are equal for the considered controllers. The estimator for the external torque with gain K_τ is not treated here for conciseness. For the PD controller, Figure 3.21a reveals a steady-state error due to the difference in the product of air density and thrust coefficient (cf. Table 3.7). With the PID controller, the actual height z only ever converges to the desired height z_d once the desired position is constant (dashed blue line). Both DOB and MRAC converge much faster with the same gains as used for the PD controller. MRAC, i.e. the adaptive approach, provides perfect trajectory tracking, i.e. $\tilde{\boldsymbol{p}}, \dot{\tilde{\boldsymbol{p}}} \to \mathbf{0}$ for $t \to \infty$. The main difference of DOB compared to MRAC is that with MRAC external forces are estimated correctly, which is due to the multiplicative nature of the air density. An example is shown in Figure 3.21b for a step disturbance in the inertial x-direction of 3 N (dashed black line). The DOB alone estimates the external force incorrectly due to wrong air density and thrust coefficient, whereas with MRAC the estimate is the exact lowpass filtered reconstruction of the external force.

Air density estimation experiment

To evaluate the air density estimation at different altitudes, experiments at DLR in Oberpfaffenhofen, Germany, and on Mount Etna in Sicily, Italy, were conducted (see Figure 3.23(a) and (b)). Figure 3.22 shows experimental results of the air density estimation approach acquired with the hexacopter Ardea (Figure 3.13b). The atmospheric parameters of that day (Dec. 7, 2018) in Oberpfaffenhofen, Germany, are $p = 960\,\text{hPa}$, $T_a = 20\,°\text{C}$, and $\delta = 46\,\%$. They are available through local weather stations and were double-checked in the lab using an external BME280 sensor [143]. It can be seen that the air density converges to $\rho = 1.137\,\text{kg/m}^3$, which is calculated for comparison using the atmosphere model from Sec-

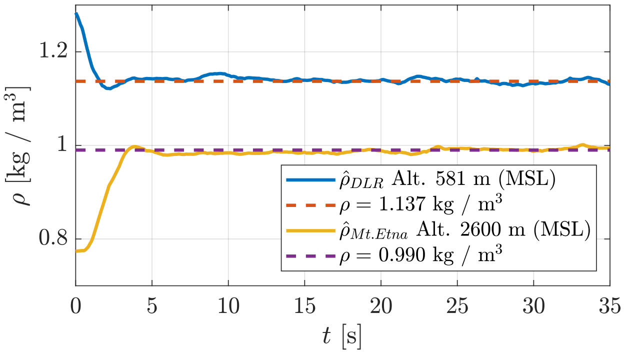


Figure 3.22: Estimated air density ρ (solid blue line). ρ_m is the modeled air density based on the sea level standard atmosphere (1013 hPa, 20 °C, black dashed line) used in the control allocation. The sensor measurement for comparison is $\rho = 1.137\,\mathrm{kg/m^3}$ (red dashed line). Copyright © 2020 IEEE [209].

tion 3.2.4. The convergence to the measured air density also proves the correctness of the identified thrust coefficients. Next, the position tracking accuracy of the adaptive controller is compared to the non-adaptive case ($\varepsilon = 0$) in experiments with the hexacopter Ardea (Figure 3.13b) and with the quadrocopter Sparrow (Figure 3.13a).

Adaptive control experiments with hexacopter Ardea

The results for Ardea are shown in Figure 3.24. The adaptive controller is activated prior to take-off and remains active until Ardea is landed. It can be seen that the trajectory is tracked more precisely with the adaptive controller (cf. Figure 3.24b) than with the non-adaptive controller (cf. Figure 3.24a). The controller gains used in both cases are $\boldsymbol{K}_d = \mathrm{diag}(1.6, 1.6, 4.0)$ and $\boldsymbol{K}_p = \mathrm{diag}(2.0, 2.0, 4.0)$ and the gain of the adaptation law is $\gamma = 0.1$. For comparison, the Root Mean Square Error (RMSE) in (x, y, z) is (0.10, 0.05, 0.12) m without and (0.09, 0.03, 0.02) m with the adaptive controller, i.e. the latter reduces the RMSE in all three axes by (10, 40, 83.3) %. Hence, without external disturbances, it is shown that the air density converges to the real value and that the tracking error is reduced.

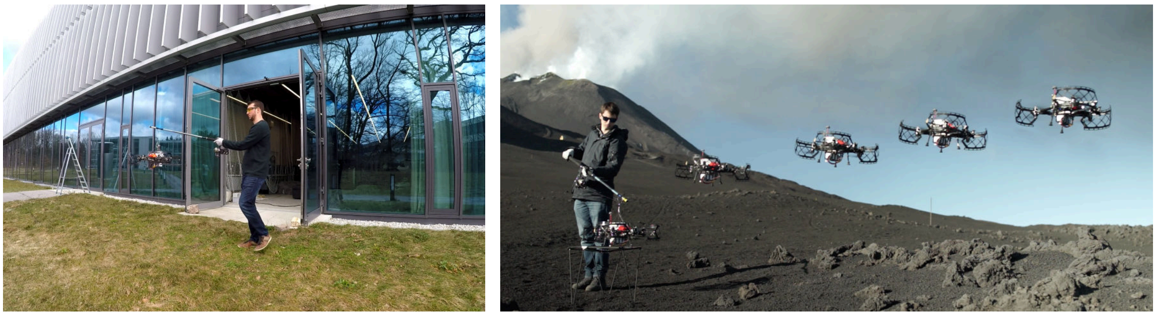
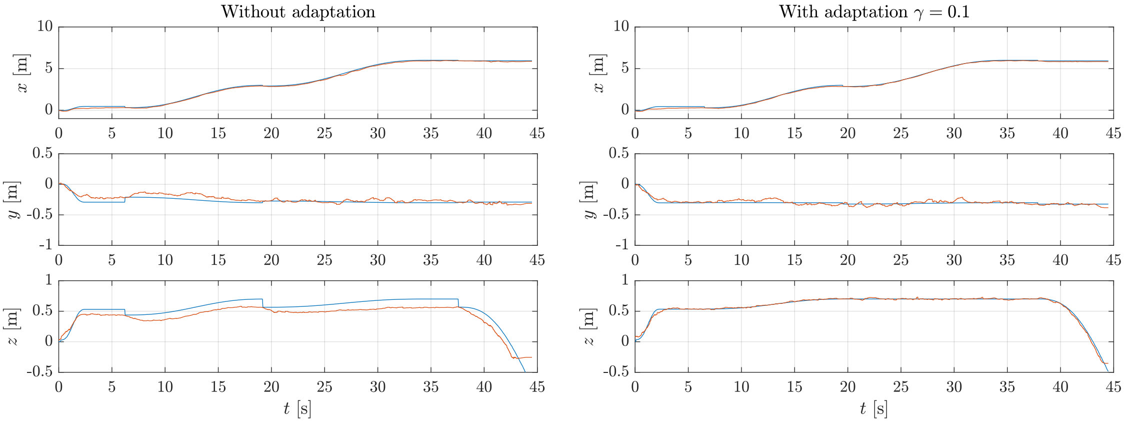

(a) **(b)** Copyright © 2020 IEEE [209].

Figure 3.23: Experiments performed with the coaxial hexacopter Ardea at DLR (a), including transition from indoors to outdoors, and on Mount Etna in Sicily (b) in order to test the adaptation to different ambient conditions. The fishing rod is attached to Ardea for safety reasons.



(a) Non-adaptive position controller ($\varepsilon = 0$).

(b) Adaptive position controller.

Figure 3.24: Experimental evaluation of position tracking with the hexacopter Ardea. The position $\boldsymbol{p} \in \mathbb{R}^3$ estimated by the VIO system is depicted in red and the desired polynomial trajectory $\boldsymbol{p}_d$ is depicted in blue. The interpolator resets to the estimated position after every trajectory segment.

An additional experiment conducted with Ardea including the transition from indoors to outdoors is shown in Figure 3.23. Note that Ardea is not equipped with a barometric pressure sensor. Take-off on Mount Etna (see Figure 3.23(b)) at an altitude of 2600 m above Mean Sea Level (MSL) without adaptive control or manual tuning of the model air density is not possible, due to the required increase in thrust [209].

Adaptive control experiments with quadrocopter Sparrow

The results of the experiments with Sparrow are shown in Figure 3.26. In Figure 3.26(a) and (b), a payload of $\Delta m = 0.1$ kg is added after approximately 14 s to assess the disturbance rejection and payload estimation of the adaptive controller compared to the nominal position controller without adaptation, i.e. $\varepsilon = 0$. With the latter, the altitude is not tracked well even without external disturbances, because of model uncertainties and decreasing battery voltage. This is usually compensated using a battery model [132]. With the adaptive controller, the altitude is tracked accurately and the additional payload is compensated, i.e. no additional battery model is needed for accurate trajectory tracking. Again, the adaptive gain is $\gamma = 0.1$. An image sequence of the experiments with Sparrow is depicted in Figure 3.25. It can be seen in Figure 3.26(b) that the payload mass Δm is estimated correctly and that the disturbance is rejected. However, without closed-loop rotor speed control, the estimates of both air density and payload are affected by changes in battery voltage and therefore require a battery model.
The results of a transition flight from indoors to outdoors are depicted in Figure 3.26(c). For this, Sparrow is piloted manually, but with adaptive height control using a downward facing range sensor. The air density estimate is close to the ground truth indoors (20 °C, 1.122 kg/m^3, dash-dotted) and outdoors (5 °C, 1.179 kg/m^3, dashed). The RMSE along the z-direction is 0.29 m with PD and 0.08 m with adaptive control.

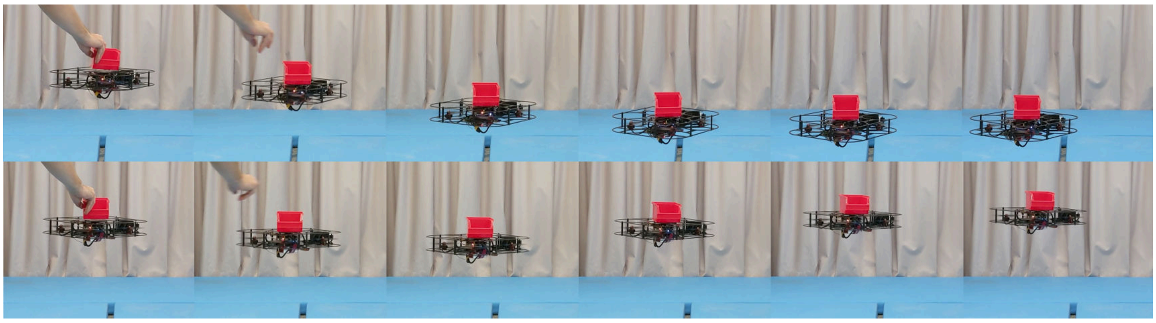
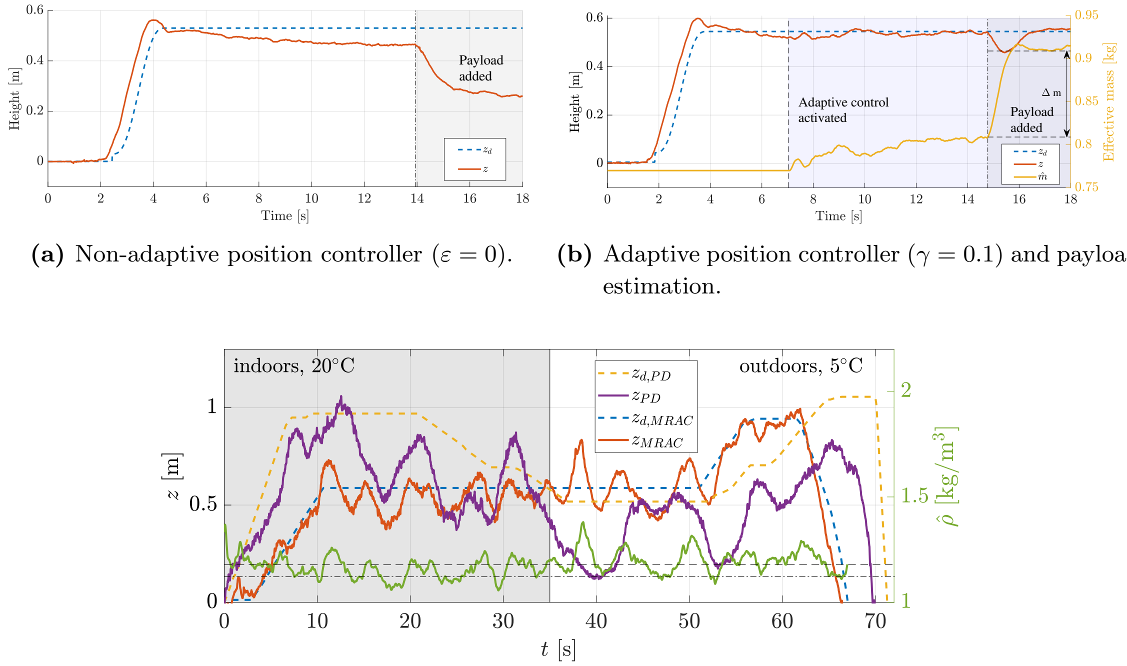


Figure 3.25: Experiments with 0.1 kg added payload without (top) and with adaptive controller (bottom). The time between each of the video frames is 0.5 s, i.e. the total duration of the motion is 2.5 s. Copyright © 2020 IEEE [209].



(a) Non-adaptive position controller ($\varepsilon = 0$).

(b) Adaptive position controller ($\gamma = 0.1$) and payload estimation.

(c) Indoor/outdoor experiment with Sparrow. Copyright © 2020 IEEE [209].

Figure 3.26: Experimental evaluation of position tracking and payload estimation (top) and indoor/outdoor transition (bottom) with the quadrocopter Sparrow. A payload of 0.1 kg is manually added in (a) and (b) (cf. image sequence in Figure 3.25) after approximately 14.8 s.

3.7 Summary and conclusion

This chapter is devoted to the development of a robust pose tracking controller for flying robots under model uncertainties, external disturbances, and actuator saturation. First, related work in trajectory tracking control, external wrench estimation, and prioritized control allocation for flying robots is summarized. Then, the considered dynamical model of a flying robot is introduced in detail. Essentially, the rigid body dynamics, quadratic and polynomial propulsion models, and an established atmosphere model are considered for controller design. The developed control approach for flying robots is composed of a generic attitude tracking controller, a position tracking controller considering uni-, bi-, and omnidirectional thrust vectors, and a prioritized control allocation procedure. For the latter, iterative and direct saturation handling methods are discussed. The robustness of the controller against variations in air density, thrust coefficients, and payload is increased using an adaptive control approach. If persistent excitation is provided, it allows for simultaneous position tracking and air density (or payload) estimation and only requires a rigid body model and position and velocity measurements. Moreover, the closed-loop system is shown to be passive. The estimated air density may be used for flight control, to enhance the accuracy of a hybrid external wrench estimator, or for meteorological measurements. In a simulative comparison with state-of-the-art approaches, the proposed adaptive controller combined with an external wrench observer clearly decreases the tracking error and increases the accuracy of the estimated external wrench. The performance of the adaptive controller is verified in experiments with the custom-built quadrocopter Sparrow and the coaxial hexacopter Ardea. Furthermore, different applications of bidirectional thrust are discussed and evaluated experimentally using the quadrocopter Sparrow. The experiments with Sparrow include the first autonomously performed transitions from upright to inverted flight of a quadrocopter with fully symmetric fixed-pitch propellers [215].

"If one is working from the point of view of getting beauty into one's equation, ... one is on a sure line of progress."
— Paul A.M. Dirac

4 Controller designs using separate models

In this chapter, coordinated control approaches are presented for robot-assisted take-off and landing of a flying robot by means of a ground-based robot manipulator. The controller designs assume a separation of the combined system dynamics of a flying robot attached to the end-effector of the robot manipulator (see Section 2.4.3). In Section 4.1, the separate dynamics are recalled and the underlying assumptions are summarized. The model is decomposed into an independent part and a part which contains the coupling terms. This allows to design distributed controllers in a straightforward way. The obtained controllers are independent in the sense that one system is only aware of its own dynamics and the other system handles the coupling terms. This is especially applicable for a robot manipulator with high payload capabilities and a light flying robot equipped with an off-the-shelf autopilot. All approaches presented in this chapter are intended for the partially clamped case (c.f. Section 2.3). Note that the content of Section 4.2 is taken from the author's article [208], whereas Section 4.3 and Section 4.4 are based on the author's papers [213] and [212], respectively.

The collection of coordinated control approaches in Section 4.2 utilizes linearization[1] of the flying robot dynamics. Using a linearization about hover (see Section 4.2.1), a stability criterion for the low-level attitude controller of the flying robot is found in Section 4.2.2. In Section 4.2.3, a computed torque control law for the manipulator is derived allowing to command a desired end-effector acceleration. Alternatively, to increase robustness against external disturbances such as a flying robot attached at the end-effector of the robot manipulator, an impedance controller with acceleration input is presented. Then, considering the linearized dynamics and using established linear control techniques, a pole placement design (Section 4.2.4) and modal decoupling design (Section 4.2.5) are presented. The latter generates a desired end-effector acceleration for the manipulator and an orientation command for the flying robot. In Section 4.2.6, all linear approaches are evaluated and compared in experiments using a DLR/KUKA Light Weight Robot (LWR) and an off-the-shelf quadrocopter.

The approach in Section 4.3 uses an established nonlinear control method, called backstepping ([87], p. 589), which is applicable to the system at hand due to its strict feedback

[1] Here, linearization, in contrast to feedback linearization, means linear approximation of the dynamics about a given state in order to obtain a linear and, hence, less complex representation of the dynamics in the vicinity of the considered state (also referred to as linearization point).

form. The backstepping method is explained briefly in Section 4.3.1. To avoid singularities, quaternions are used to represent the attitude of the flying robot. Thus, the backstepping approach is extended to quaternions in Section 4.3.2 and the quaternion error and an additional control input are introduced in Section 4.3.3. Then, a backstepping controller is designed in Section 4.3.4 and used to compute a feed-forward force for the augmented impedance controller presented in Section 4.3.5. In Section 4.3.6, the proposed backstepping control approach is evaluated experimentally using a DLR/KUKA LWR and an off-the-shelf quadrocopter.

The above approaches assume a hovering flying robot and do not explicitly consider motion of the base of the robot manipulator. Therefore, in Section 4.4 base motion compensation and active thrust control are added. For illustration and simulation purposes, a planar model of a robot manipulator on a moving base is introduced in Section 4.4.1. The extended controller in Section 4.4.2 relies on known (measured) base motion. In Section 4.4.3, the attitude controller of the flying robot and the stability criterion (4.18) is applied to the planar case. The active thrust vector control approach in Section 4.4.4 uses the feed-forward acceleration provided by a trajectory generator. Finally, both extensions are evaluated in an extensive simulation case study in Section 4.4.5 using a planar moving base model, simulated ship motion, and the inertial parameters of five flying robots with different thrust-to-weight ratios.

4.1 Separate dynamics of flying robot attached to manipulator

Recall the dynamics (2.43) of the combined system

$$\begin{bmatrix} \boldsymbol{M}_{bb} & \boldsymbol{M}_{br} & \boldsymbol{M}_{bu} \\ \boldsymbol{M}_{br}^T & \boldsymbol{M}_{rr} & \boldsymbol{M}_{ru} \\ \boldsymbol{M}_{bu}^T & \boldsymbol{M}_{ru}^T & \boldsymbol{M}_{uu} \end{bmatrix} \begin{pmatrix} \ddot{\boldsymbol{\phi}}_b \\ \ddot{\boldsymbol{\phi}}_r \\ \ddot{\boldsymbol{\phi}}_u \end{pmatrix} + \begin{bmatrix} \boldsymbol{C}_{bb} & \boldsymbol{C}_{br} & \boldsymbol{C}_{bu} \\ \boldsymbol{C}_{rb} & \boldsymbol{C}_{rr} & \boldsymbol{C}_{ru} \\ \boldsymbol{C}_{ub} & \boldsymbol{C}_{ur} & \boldsymbol{C}_{uu} \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{\phi}}_b \\ \dot{\boldsymbol{\phi}}_r \\ \dot{\boldsymbol{\phi}}_u \end{pmatrix} - \begin{pmatrix} \boldsymbol{Q}_b \\ \boldsymbol{Q}_r \\ \boldsymbol{Q}_u \end{pmatrix} = \boldsymbol{0}, \tag{4.1}$$

composed of base b, robot manipulator r, and flying robot (or UAV) u. From (4.1) the separate dynamics of the flying robot are extracted as

$$\boldsymbol{M}_{uu}\ddot{\boldsymbol{\phi}}_u + \boldsymbol{C}_{uu}\dot{\boldsymbol{\phi}}_u + \boldsymbol{Q}_{ext} = \boldsymbol{Q}_u, \tag{4.2}$$

where $\boldsymbol{Q}_{ext} = \boldsymbol{M}_{ru}^T\ddot{\boldsymbol{\phi}}_r + \boldsymbol{C}_{ur}\dot{\boldsymbol{\phi}}_r + \boldsymbol{M}_{bu}^T\ddot{\boldsymbol{\phi}}_b + \boldsymbol{C}_{ub}\dot{\boldsymbol{\phi}}_b$ comprises all coupling forces and $\boldsymbol{\phi}_u$ is the configuration of the flying robot. The configuration space velocities are related to task space velocities via

$$\begin{pmatrix} \boldsymbol{\omega}_u \\ \boldsymbol{v}_u \end{pmatrix} = \boldsymbol{J}_u(\boldsymbol{\phi}_b, \boldsymbol{\phi}_r, \boldsymbol{\phi}_u) \begin{pmatrix} \dot{\boldsymbol{\phi}}_b \\ \dot{\boldsymbol{\phi}}_r \\ \dot{\boldsymbol{\phi}}_u \end{pmatrix}, \tag{4.3}$$

where $\boldsymbol{J}_u$ is a Jacobian matrix. Equation (4.2) are the dynamics of a single rigid body. This follows directly from observation 3 in Section 2.4.5. Hence, the model (4.2) is equivalent to the rigid body dynamics of a flying robot shown in Section 3.2.1. The rotational dynamics

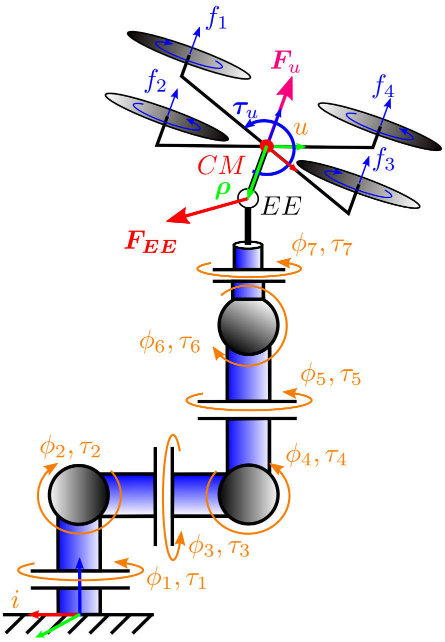


Figure 4.1: Model of a quadrocopter connected to a serial robotic manipulator with seven DoF.

may be written in the body-fixed frame u and the translational dynamics in the inertial frame i, such that[2]

$$\begin{bmatrix} \boldsymbol{I}_u & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & m_u \boldsymbol{E}_{3\times3} \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{\omega}}_u \\ \dot{\boldsymbol{v}}_u \end{pmatrix} = \begin{pmatrix} \boldsymbol{S}(\boldsymbol{I}_u \boldsymbol{\omega}_u)\boldsymbol{\omega}_u + \boldsymbol{\tau}_{ur} + \boldsymbol{\tau}_{ext} \\ -m_u g \boldsymbol{e}_3 + \boldsymbol{F}_{ur} + \boldsymbol{F}_{ext} \end{pmatrix}. \tag{4.4}$$

Therein, m_u is the mass of the flying robot, $\boldsymbol{I}_u = \text{diag}(I_{xx}, I_{yy}, I_{zz})$ is its diagonal inertia tensor with respect to its center of mass CM, g is the gravitational acceleration, $\boldsymbol{\omega}_u$ is the rotational, $\boldsymbol{v}_u$ is the translational velocity, and $\boldsymbol{e}_3 = (0 \quad 0 \quad 1)^T$ is a unit vector. The matrix $\boldsymbol{S}(\cdot)$ again denotes the skew-symmetric cross product operator (2.5) defined in Section 2.4.1. Note, that $\boldsymbol{\tau}_{ur}$ and $\boldsymbol{F}_{ur}$ both contain forces and moments applied by flying robot and robot manipulator and $\boldsymbol{\tau}_{ext}$ as well as $\boldsymbol{F}_{ext}$ contain external disturbances and the associated parts of the coupling forces $\boldsymbol{Q}_{ext}$.

From a torque balance about CM for the flying robot connected to the manipulator via a universal hinge as depicted in Figure 4.1, one obtains

$$\boldsymbol{\tau}_{ur} = \boldsymbol{\tau}_{att} + \boldsymbol{S}(\boldsymbol{\rho})\boldsymbol{R}_{ui}(\varphi, \theta, \psi)\, \boldsymbol{F}_{EE}, \tag{4.5}$$

where $\boldsymbol{R}_{ui}(\varphi, \theta, \psi)$ is a rotation matrix constructed using Euler angles $\boldsymbol{\Phi} = (\varphi \quad \theta \quad \psi)^T$, that transfers a vector given in the inertial frame i to the body-fixed frame u of the UAV. The force applied by the robot at its end-effector is denoted as $\boldsymbol{F}_{EE}$. The displacement vector written in u from CM to EE is defined as $\boldsymbol{\rho} = (0 \quad 0 \quad -l)^T$ with the length l of the universal hinge. The torque required to stabilize a desired attitude is denoted as $\boldsymbol{\tau}_{att}$ in (4.5). As shown in Section 3.4, it holds for underactuated flying robots, like helicopters

[2] The equivalence of (4.2) and (4.4) may be verified by inserting the derivative w.r.t. time of (4.3) in (4.4), multiplying $\boldsymbol{J}_u^T$ from the left, and rearranging. It also follows from the general partial decomposition of redundant systems presented in Section 2.4.5.

and multicopters, that the thrust vector is given by $\boldsymbol{T} = (0 \quad 0 \quad T)^T$. It is assumed to be always perpendicular to the (x_u,y_u)-plane of the UAV [165]. For a quadrocopter, as shown in Figure 4.1, the collective thrust T is the sum of all rotor thrust forces $\sum_{i=1}^{4} T_i$. If the flying robot is connected to the robot manipulator, a balance of forces yields

$$\boldsymbol{F}_{ur} = T\boldsymbol{R}_{ui}^T(\varphi, \theta, \psi)\boldsymbol{e}_3 + \ \boldsymbol{F}_{EE}. \tag{4.6}$$

Combining equations (4.4), (4.5), and (4.6) with the kinematics

$$\dot{\boldsymbol{r}}_{CM} = \ \boldsymbol{v}_u, \tag{4.7}$$
$$\dot{\boldsymbol{R}}_{ui}^T = -\boldsymbol{R}_{ui}^T\boldsymbol{S}^T(\boldsymbol{\omega}_u) = \boldsymbol{R}_{ui}^T\boldsymbol{S}(\boldsymbol{\omega}_u), \tag{4.8}$$

gives the complete equations of motion of the flying robot fixed to the robot manipulator via a universal hinge. Therein, $\boldsymbol{r}_{CM}$ is the position of the aerial vehicle's center of mass in the inertial frame.

In the following, it is assumed that

- the flying robot is equipped with an arbitrary attitude controller which generates $\boldsymbol{\tau}_{att}$,
- T is either only used for gravity compensation or alternatively for height control,
- the forces applied by the flying robot onto the manipulator are negligible.

4.2 Linear state space control

In this section, the equations of motion (4.4) of the unmanned aerial vehicle fixed to a robot manipulator by means of a universal hinge are linearized and analyzed for stability (Section 4.2.2). The nonlinear terms in the translational dynamics, introduced by the transformation from body-fixed to inertial frame, as well as the cross product term in the rotational dynamics vanish through linearization. For a light flying robot, their influence can be considered small, as will be shown in the experiments in Section 4.2.6 of this chapter. Linear control is a well-studied field, so many tools are readily available and the results can be interpreted in-depth. In the following, these established linear control methods are applied:

- PID control ([166], p. 263) in Section 4.2.3,
- pole placement using Ackermann formula ([167], p. 331) in Section 4.2.4,
- decoupling via Roppenecker formula [168], ([167], p. 334) in Section 4.2.5.

The overall controller scheme with its three different realizations is shown in Figure 4.2. For robot manipulator control, a nonlinear model and a feedback linearization in task space (cf. Section 2.6) is used, which tracks a desired translational acceleration computed by the linear controllers. The coupling terms in the task space dynamics can be incorporated if they are known. Uncertainties in flying robot model and control allocation (which influence thrust and, thus, the gravity compensation of the flying robot) are accounted for via an integral

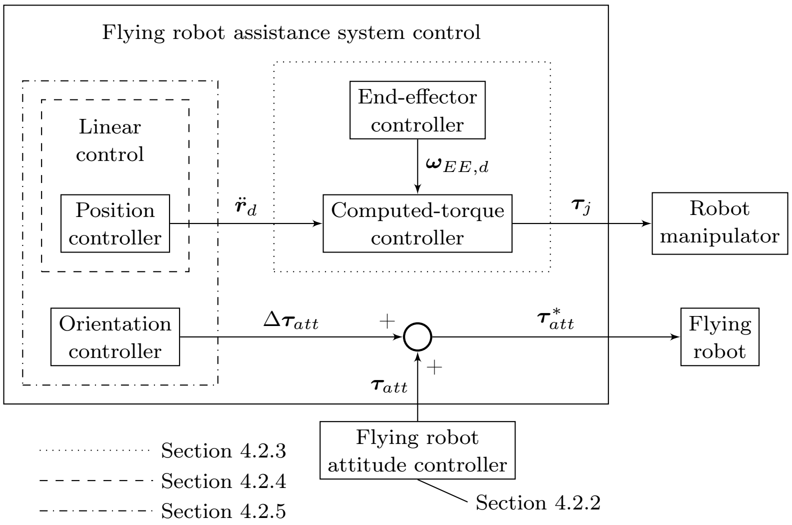


Figure 4.2: Scheme of the state space control approach with the three realizations of the controller depicted using different line styles. The controller (4.23) for the orientation of the end-effector as well as the feed-forward torque controller (4.24) from Section 4.2.3 are used for all three realizations. The coupled controller (4.29) - (4.31) in Section 4.2.4 produces a desired translational acceleration of the robot's end-effector to control both position and orientation of the flying robot (or UAV) simultaneously. In order to fulfill the same task, the decoupling controller (4.39)-(4.40) in Section 4.2.5 generates an appropriate end-effector acceleration and a torque which is added to the attitude control torque of the aerial vehicle.

part in the linear control law. The computed-torque control law for the manipulator is derived in Section 4.2.3 allowing to command a desired end-effector acceleration. Then, a nominal linear state space controller is designed, neglecting the dynamics of the flying robot. This controller is then extended in Section 4.2.4 in order to control both the flying robot's center of mass position CM and its orientation about CM in a coupled manner using the robot manipulator. The attitude controller of the flying robot is assumed to be linear with known gains, as presented in Section 4.2.2. Finally, a modal control approach [168] is used to decouple the acceleration of the end-effector from the orientation of the flying robot in Section 4.2.5.

4.2.1 Linear approximation about hover

Here, the linearization $\boldsymbol{h}(\boldsymbol{x}) \approx \boldsymbol{h}(\boldsymbol{x_0}) + \nabla\boldsymbol{h}|_{\boldsymbol{x}_0}\Delta\boldsymbol{x}$ with $\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_0$ ([166], p. 55) of the flying robot dynamics about hover $\boldsymbol{R}_{ui} = \boldsymbol{E}_{3\times3}$ is presented briefly. Given the orientation of the aerial vehicle is described by the rotation matrix $\boldsymbol{R}_{ui}$ introduced in Section 2.4.1. The equations of motion (4.4) are linearized about $\varphi_0 = \theta_0 = \psi_0 = 0$ and $\boldsymbol{\omega}_0 = \boldsymbol{0}$, where the index 0 indicates the steady hover state. The linearized rotation matrix is given by

$$\boldsymbol{R}_{ui}(\Delta\varphi, \Delta\theta, \Delta\psi) \approx \begin{bmatrix} 1 & \Delta\psi & -\Delta\theta \\ -\Delta\psi & 1 & \Delta\varphi \\ \Delta\theta & -\Delta\varphi & 1 \end{bmatrix}, \tag{4.9}$$

for which the small angle assumption $\sin(x) = x$, $\cos(x) = 1$ is used and products of deviations are neglected, e.g. $\Delta\psi\Delta\theta = 0$. Under equal assumptions, it follows for the kinematics (4.8) that

$$\dot{\boldsymbol{R}}_{ui}^T \approx \begin{bmatrix} 0 & -\Delta\omega_z & \Delta\omega_y \\ \Delta\omega_z & 0 & -\Delta\omega_x \\ -\Delta\omega_y & \Delta\omega_x & 0 \end{bmatrix}. \tag{4.10}$$

Finally, with $\dot{\boldsymbol{v}}_0 = \boldsymbol{0}$, $\boldsymbol{\tau}_{ur,0} = \boldsymbol{0}$, $T_0 = m_u g$, and $\boldsymbol{F}_{EE,0} = \boldsymbol{0}$, the rotational dynamics are obtained as

$$\dot{\boldsymbol{\omega}}_u \approx \begin{bmatrix} 0 & -\frac{m_u l}{I_{xx}} & 0 \\ \frac{m_u l}{I_{yy}} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \left(\Delta\ddot{\boldsymbol{r}}_{EE} + g \begin{pmatrix} \Delta\theta \\ -\Delta\varphi \\ 1 \end{pmatrix} \right) + \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \Delta\boldsymbol{\tau}_{att} \tag{4.11}$$

and the translational dynamics as

$$\dot{\boldsymbol{v}}_u \approx \begin{bmatrix} 0 & g & 0 \\ g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta\varphi \\ -\Delta\theta \\ \Delta\psi \end{pmatrix} + \frac{1}{m_u} \Delta\boldsymbol{F}_{\mathrm{u}} + \Delta\ddot{\boldsymbol{r}}_{EE}. \tag{4.12}$$

Therein, $\ddot{\boldsymbol{r}}_{EE}$ is the acceleration of the manipulator's end-effector and $\boldsymbol{\tau}_{att}$ and $\ddot{\boldsymbol{r}}_{EE}$ are the control inputs. This is similar to the equations of motion of a linearized inverted pendulum, e.g. used in [75], but with the difference that the attitude may be controlled directly via $\boldsymbol{\tau}_{att}$, i.e. by means of the rotors of the flying robot. Since $\boldsymbol{h}(\boldsymbol{x}_0) = \boldsymbol{0}$ and $\boldsymbol{x}_0 = \boldsymbol{0}$ and therefore $\Delta\boldsymbol{x} = \boldsymbol{x}$, the Δ notation is omitted in the following for brevity.

4.2.2 Stability criterion for PD attitude control of flying robot

The question arises whether it is possible for a given attitude controller to stabilize the orientation while the flying robot is rigidly connected to the end-effector of the manipulator. Therefore, the stability of the hover state $(\boldsymbol{\omega}^T \quad \boldsymbol{\Phi}^T)^T = \boldsymbol{0}_{6\times1}$ with constant thrust force $T = m_u g$ is analyzed. It is reasonable to assume that the flying robot is equipped with a linear Proportional Derivative (PD) attitude controller [165] and that the rotors produce the torque

$$\boldsymbol{\tau}_{att} = -\mathrm{diag}(k_{\omega_x}, k_{\omega_y}, k_{\omega_z})\, \boldsymbol{\omega}_u - \mathrm{diag}(k_\varphi, k_\theta, k_\psi) \begin{pmatrix} \varphi \\ \theta \\ \psi \end{pmatrix}. \tag{4.13}$$

The torque $\boldsymbol{\tau}_{att}$ is generated depending on the geometry of the flying robot and on the rotor parameters. Further details are presented in Section 3.2 and Section 3.4. In the following, x- and y-direction are combined for conciseness. This is indicated using the notation x/y which means that the equations equally hold for x or y, respectively. Inserting (4.5), (4.6), and (4.13) in (4.4) and linearizing about the hover state yields for the rotational dynamics about the x-, y-, and z-axis, respectively (cf. body coordinate axes depicted in Figure 4.1)

$$\begin{pmatrix} \dot{\omega}_{x/y} \\ \dot{\varphi}/\dot{\theta} \end{pmatrix} = \underbrace{\begin{bmatrix} \frac{-k_{\omega_{x/y}}}{I_{xx/yy}} & \frac{-k_{\varphi/\theta}+m_u gl}{I_{xx/yy}} \\ 1 & 0 \end{bmatrix}}_{\boldsymbol{A}_{x/y}} \begin{pmatrix} \omega_{x/y} \\ \varphi/\theta \end{pmatrix}, \tag{4.14}$$

$$\begin{pmatrix} \dot{\omega}_z \\ \dot{\psi} \end{pmatrix} = \underbrace{\begin{bmatrix} \frac{-k_{\omega_z}}{I_{zz}} & \frac{-k_{\psi}}{I_{zz}} \\ 1 & 0 \end{bmatrix}}_{\boldsymbol{A}_z} \begin{pmatrix} \omega_z \\ \psi \end{pmatrix}. \tag{4.15}$$

The characteristic polynomials $P_i = \det(\boldsymbol{A}_i - \lambda_i \boldsymbol{E}_{2\times 2})$ for $i \in \{x, y, z\}$ are obtained as

$$P_{x/y} = \lambda_{x/y}^2 + \frac{k_{\omega_{x/y}}}{I_{xx/yy}}\lambda_{x/y} + \frac{k_{\varphi/\theta} - m_u gl}{I_{xx/yy}}, \tag{4.16}$$

$$P_z = \lambda_z^2 + \frac{k_{\omega_z}}{I_{zz}}\lambda_z + \frac{k_\psi}{I_{zz}}. \tag{4.17}$$

According to the Routh-Hurwitz criterion ([166], p. 391), the hover state is stable if all coefficients within (4.16) and (4.17) are positive. Since all other parameters are positive, it follows that the gains $k_{\varphi/\theta}$, $k_{\omega_{x/y}}$, k_ψ, and k_{ω_z} have to satisfy the (necessary, but not sufficient) conditions [208]

$$k_{\varphi/\theta} > m_u gl, \quad k_{\omega_{x/y}} > 0, \quad k_\psi > 0, \quad k_{\omega_z} > 0. \tag{4.18}$$

Moreover, from the eigenfrequency ω_0 and the damping ζ of (4.16)

$$\omega_0 = \sqrt{\frac{k_{\varphi/\theta} - m_u gl}{I_{xx/yy}}}, \tag{4.19}$$

$$\zeta = \frac{k_{\omega_{x/y}}}{2\sqrt{I_{xx/yy}(k_{\varphi/\theta} - m_u gl)}}, \tag{4.20}$$

it can be seen that by increasing the length l of the universal hinge or the mass m_u of the flying robot, the eigenfrequency ω_0 can be decreased while the damping ζ is increased and vice versa.

4.2.3 Manipulator control with acceleration input

In this section, the dynamics (2.42) of a serial robotic manipulator with n joints, as presented in Section 2.4.4, are considered. In order to mitigate the influence of the manipulator's motor inertias, of friction in the joints, and of model uncertainties, the integral of the position error $\tilde{\boldsymbol{r}} = \boldsymbol{r}_{EE,d} - \boldsymbol{r}_{EE}$ is added to the system states. The desired translational acceleration of the end-effector $\ddot{\boldsymbol{r}}_{EE,d}$ is defined as intermediate control input. Assuming direct control of the acceleration, a linear state space model for the translational motion of the end-effector is obtained as

$$\begin{pmatrix} \ddot{\boldsymbol{r}}_{EE} \\ \dot{\boldsymbol{r}}_{EE} \\ \tilde{\boldsymbol{r}} \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \boldsymbol{E} & 0 & 0 \\ 0 & -\boldsymbol{E} & 0 \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{r}}_{EE} \\ \boldsymbol{r}_{EE} \\ \int \tilde{\boldsymbol{r}} \mathrm{d}t \end{pmatrix} + \begin{pmatrix} \ddot{\boldsymbol{r}}_{EE,d} \\ 0 \\ \boldsymbol{r}_{EE,d} \end{pmatrix}. \tag{4.21}$$

A linear controller

$$\ddot{\boldsymbol{r}}_{EE,d} = -\boldsymbol{K}_1 \begin{pmatrix} \dot{\boldsymbol{r}}_{EE} & \boldsymbol{r}_{EE} & \int \tilde{\boldsymbol{r}}\mathrm{d}t \end{pmatrix}^T \tag{4.22}$$

can be designed such that the poles of the closed-loop are placed at a desired location in the left complex half-plane using e.g. Ackermann's formula [169]. Since flying robot and manipulator are connected via the rotational hinge, the orientation of the aerial vehicle and the robotic end-effector can be controlled independently. The control law from [170] is adopted in order to generate a desired rotational acceleration for the robot

$$\dot{\boldsymbol{\omega}}_{EE,d} = -\frac{1}{2}k_{EE} \begin{pmatrix} (\boldsymbol{R}_{EE}\boldsymbol{R}_d)^{\{3,2\}} - (\boldsymbol{R}_{EE}\boldsymbol{R}_d)^{\{2,3\}} \\ (\boldsymbol{R}_{EE}\boldsymbol{R}_d)^{\{1,3\}} - (\boldsymbol{R}_{EE}\boldsymbol{R}_d)^{\{3,1\}} \\ (\boldsymbol{R}_{EE}\boldsymbol{R}_d)^{\{2,1\}} - (\boldsymbol{R}_{EE}\boldsymbol{R}_d)^{\{1,2\}} \end{pmatrix} - k_{\omega_{EE}}\boldsymbol{\omega}_{EE} \tag{4.23}$$

using the actual and the desired orientation $\boldsymbol{R}_{EE}$ and $\boldsymbol{R}_d$ as well as the rotational part of the velocity of the end-effector $(\boldsymbol{v}^T \quad \boldsymbol{\omega}^T)^T_{EE} = \boldsymbol{J}(\boldsymbol{\phi})\dot{\boldsymbol{\phi}}$. In (4.23), the notation $(\boldsymbol{R}_{EE}\boldsymbol{R}_d)^{\{i,j\}}$ denotes the element of $\boldsymbol{R}_{EE}\boldsymbol{R}_d$ in row i and column j. Note the similarity with (3.19) in Section 3.3.1 and the vee map (2.7) defined in Section 2.4.1. The feedback linearization (2.81) - (2.83) presented in Section 2.6 allows to directly include the desired acceleration, for which it should hold that $(\ddot{\boldsymbol{r}}^T \quad \dot{\boldsymbol{\omega}}^T)^T_{EE} = (\ddot{\boldsymbol{r}}^T \quad \dot{\boldsymbol{\omega}}^T)^T_{EE,d}$. Hence, (2.83) becomes

$$\boldsymbol{\tau}_j = \boldsymbol{J}^T(\boldsymbol{\phi})\boldsymbol{\Lambda}(\boldsymbol{\phi}) \begin{pmatrix} \ddot{\boldsymbol{r}} \\ \dot{\boldsymbol{\omega}} \end{pmatrix}_{EE,d} + \boldsymbol{J}^T(\boldsymbol{\phi})\boldsymbol{\mu}_d(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}}) + \boldsymbol{g}(\boldsymbol{\phi}) + \boldsymbol{\tau}_{nsp}, \tag{4.24}$$

where $\boldsymbol{\tau}_{nsp}$ is a general nullspace torque (2.51) introduced in Section 2.4.4. The latter is a design parameter and can be used to include additional objectives, e.g. to realize a desired posture of the manipulator without interfering with the end-effector task. An experimental evaluation of the controller (4.24) with (4.22) and (4.23) can be found in Section 4.2.6. In the next section, the control law (4.21) is extended in order to stabilize the position and the orientation of the flying robot simultaneously.

4.2.4 Flying robot control design using pole placement

As soon as the aerial vehicle is connected to the manipulator via the universal hinge, the position of its center of mass in the inertial frame is given by

$${}^{i}\boldsymbol{r}_{CM} = {}^{i}\boldsymbol{r}_{EE} + \boldsymbol{R}_{ib}\,{}^{b}\boldsymbol{\rho}. \tag{4.25}$$

It is assumed that the flying robot is equipped with the linear attitude controller (4.13) and the thrust force is $T = mg$. This yields the following independent linear state space models for the x-, y-, and z-motion of the flying robot

$$\begin{pmatrix} \ddot{r}_x \\ \dot{r}_x \\ \dot{\omega}_y \\ \dot{\theta} \end{pmatrix}_{UAV} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & g \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{$\boldsymbol{A}_y$}} \\ 0 & 0 & & \end{bmatrix}}_{\overline{\boldsymbol{A}}_x} \begin{pmatrix} \dot{r}_x \\ r_x \\ \omega_y \\ \theta \end{pmatrix}_{UAV} + \begin{pmatrix} 1 \\ 0 \\ -\frac{ml}{I_{yy}} \\ 0 \end{pmatrix} \ddot{r}_{EE,x}, \tag{4.26}$$

$$\begin{pmatrix} \ddot{r}_y \\ \dot{r}_y \\ \dot{\omega}_x \\ \dot{\varphi} \end{pmatrix}_{UAV} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & -g \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{$\boldsymbol{A}_x$}} \\ 0 & 0 & & \end{bmatrix}}_{\bar{\boldsymbol{A}}_y} \begin{pmatrix} \dot{r}_y \\ r_y \\ \omega_x \\ \varphi \end{pmatrix}_{UAV} + \begin{pmatrix} 1 \\ 0 \\ \frac{ml}{I_{xx}} \\ 0 \end{pmatrix} \ddot{r}_{EE,y}, \tag{4.27}$$

$$\begin{pmatrix} \ddot{r}_z \\ \dot{r}_z \\ \dot{\omega}_z \\ \dot{\psi} \end{pmatrix}_{UAV} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{$\boldsymbol{A}_z$}} \\ 0 & 0 & & \end{bmatrix} \begin{pmatrix} \dot{r}_z \\ r_z \\ \omega_z \\ \psi \end{pmatrix}_{UAV} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \ddot{r}_{EE,z}. \tag{4.28}$$

It can be seen from (4.28), that the yaw angle ψ of the UAV is not controllable by $\ddot{\boldsymbol{r}}_{EE}$. Instead, it can be stabilized by the UAV attitude controller at an arbitrary orientation ψ_0, which is free but has to be known. Furthermore, to reduce the influence of motor inertia and friction, the integral of the position error $\tilde{\boldsymbol{r}} = \boldsymbol{r}_{CM,d} - \boldsymbol{r}_{CM}$ is again added to the system's state vector. Then, instead of the pure translational controller (4.22), three independent linear controllers are designed using pole placement [169]

$$\ddot{r}_{EE,x,d} = -\boldsymbol{K}_2 \begin{pmatrix} \dot{r}_x & r_x - r_{x,0} & \omega_y & \theta & \int \tilde{r}_x \mathrm{d}t \end{pmatrix}^T, \tag{4.29}$$

$$\ddot{r}_{EE,y,d} = -\boldsymbol{K}_3 \begin{pmatrix} \dot{r}_y & r_y - r_{y,0} & \omega_x & \varphi & \int \tilde{r}_y \mathrm{d}t \end{pmatrix}^T, \tag{4.30}$$

$$\ddot{r}_{EE,z,d} = -\boldsymbol{K}_4 \begin{pmatrix} \dot{r}_z & r_z - r_{z,0} & \int \tilde{r}_z \mathrm{d}t \end{pmatrix}^T. \tag{4.31}$$

Therein, $(r_{x,0} \quad r_{y,0} \quad r_{z,0})^T$ is the UAV's initial center of mass position. Inserting (4.29), (4.30), and (4.31) in (4.24) and using (4.23) for stabilizing the end-effector orientation at $\boldsymbol{R}_d = \boldsymbol{E}$ yields the final computed-torque control law for the robot manipulator.

The stability criterion (4.18) can be interpreted as the lower bound on the controller gains, since the acceleration $\ddot{\boldsymbol{r}}_{EE}$ is assumed to be zero. An additional bound on the maximum end-effector acceleration can be found using the Lyapunov function

$$V_x = \frac{1}{2}\omega_x^2 + \frac{1}{2}(k_\varphi - m_u g l)\varphi^2 \geq 0, \tag{4.32}$$

with $k_\varphi > m_u g l$. A Lyapunov function V_y for ω_y and θ can be found analogously (see Appendix A8). Using the derivative of (4.32) and V_y the following additional criteria is found

$$\ddot{r}_{EE,x} \geq -\frac{k_{\omega_y} I_{yy}}{m_u l}\omega_y, \qquad \ddot{r}_{EE,y} \leq \frac{k_{\omega_x} I_{xx}}{m_u l}\omega_x. \tag{4.33}$$

The complete derivation of (4.33) can be found in Appendix A8. In conclusion, the flying robot can be connected to the manipulator via an universal hinge without considering its attitude in the manipulator controller, as long as both criteria (4.18) and (4.33) are fulfilled.

4.2.5 Flying robot control design via modal decoupling

In Section 4.2.4, actuation of the flying robot is neglected, i.e. it is treated like an inverted pendulum. However, the flying robot is able to control its orientation independently using its rotors. The communication channel between manipulator and flying robot allows to send commands and enables the UAV to contribute to the take-off and landing task. This can be considered conveniently by adding a torque input $\Delta\boldsymbol{\tau}_{att}$ to the linear system dynamics, as depicted in Figure 4.2. The torque $\Delta\boldsymbol{\tau}_{att}$ is the deviation from the already present attitude control torque $\boldsymbol{\tau}_{att}$ of the flying robot and leads to

$$\begin{pmatrix}\ddot{r}_x\\ \dot{r}_x\\ \dot{\omega}_y\\ \dot{\theta}\end{pmatrix}_{UAV} = \overline{\boldsymbol{A}}_x \begin{pmatrix}\dot{r}_x\\ r_x\\ \omega_y\\ \theta\end{pmatrix}_{UAV} + \underbrace{\begin{bmatrix}1 & 0\\ 0 & 0\\ -\frac{ml}{I_{yy}} & \frac{1}{I_{yy}}\\ 0 & 0\end{bmatrix}}_{\overline{\boldsymbol{B}}_x} \underbrace{\begin{pmatrix}\ddot{r}_{EE,x}\\ \Delta\tau_{att,y}\end{pmatrix}}_{\boldsymbol{u}_x}, \tag{4.34}$$

$$\begin{pmatrix}\ddot{r}_y\\ \dot{r}_y\\ \dot{\omega}_x\\ \dot{\varphi}\end{pmatrix}_{UAV} = \overline{\boldsymbol{A}}_y \begin{pmatrix}\dot{r}_y\\ r_y\\ \omega_x\\ \varphi\end{pmatrix}_{UAV} + \underbrace{\begin{bmatrix}1 & 0\\ 0 & 0\\ \frac{ml}{I_{xx}} & \frac{1}{I_{xx}}\\ 0 & 0\end{bmatrix}}_{\overline{\boldsymbol{B}}_y} \underbrace{\begin{pmatrix}\ddot{r}_{EE,y}\\ \Delta\tau_{att,x}\end{pmatrix}}_{\boldsymbol{u}_y}, \tag{4.35}$$

through which ψ is now controllable as well:

$$\begin{pmatrix}\ddot{r}_z\\ \dot{r}_z\end{pmatrix}_{UAV} = \begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix} \begin{pmatrix}\dot{r}_z\\ r_z\end{pmatrix}_{UAV} + \begin{pmatrix}1\\ 0\end{pmatrix} \ddot{r}_z, \tag{4.36}$$

$$\begin{pmatrix}\dot{\omega}_z\\ \dot{\psi}\end{pmatrix}_{UAV} = \boldsymbol{A}_z \begin{pmatrix}\omega_z\\ \psi\end{pmatrix}_{UAV} + \begin{pmatrix}\frac{1}{I_{zz}}\\ 0\end{pmatrix} \Delta\tau_{att,z}. \tag{4.37}$$

Again, the integral of the position error $\tilde{\boldsymbol{r}}$ and additionally the integral of the orientation error $\tilde{\boldsymbol{\Phi}} = (\tilde{\varphi} \quad \tilde{\theta} \quad \tilde{\psi})^T = (\varphi_d - \varphi \quad \theta_d - \theta \quad \psi_d - \psi)^T$ is added to the system states. The matrices $\overline{\boldsymbol{A}}_x$, $\overline{\boldsymbol{A}}_y$, $\overline{\boldsymbol{B}}_x$, and $\overline{\boldsymbol{B}}_y$ are adapted accordingly and become

$$\overline{\boldsymbol{A}}^*_{x/y} = \left[\begin{array}{cccc|c} \multicolumn{4}{c}{\overline{\boldsymbol{A}}_{x/y}} & \boldsymbol{0}\\ 0 & -1 & 0 & 0 & \multirow{2}{*}{$\boldsymbol{0}$}\\ 0 & 0 & 0 & -1 & \end{array}\right], \ \overline{\boldsymbol{B}}^*_{x/y} = \begin{bmatrix}\overline{\boldsymbol{B}}_{x/y}\\ \boldsymbol{0}\end{bmatrix}. \tag{4.38}$$

The aim is to find decoupling control laws

$$\boldsymbol{u}_x = \boldsymbol{K}_x \begin{pmatrix}\dot{r}_x & r_x - r_{x,0} & \omega_y & \theta & \int \tilde{r}_x \mathrm{d}t & \int \tilde{\theta} \mathrm{d}t\end{pmatrix}^T_{UAV}, \tag{4.39}$$

$$\boldsymbol{u}_y = \boldsymbol{K}_y \begin{pmatrix}\dot{r}_y & r_y - r_{y,0} & \omega_x & \varphi & \int \tilde{r}_y \mathrm{d}t & \int \tilde{\varphi} \mathrm{d}t\end{pmatrix}^T_{UAV}, \tag{4.40}$$

for (4.34) and (4.35) and use a formula by Roppenecker [168] to compute the gains $\boldsymbol{K}_x$ and $\boldsymbol{K}_y$

$$\boldsymbol{K}_{x/y} = \Big[\boldsymbol{p}_1 \ldots \boldsymbol{p}_6\Big]_{x/y} \Big[\boldsymbol{w}_1 \ldots \boldsymbol{w}_6\Big]^{-1}_{x/y}. \tag{4.41}$$

Therein, $\boldsymbol{p}_{i,x/y}$ with $i \in \{1,2,3,4,5,6\}$ are the so-called parameter vectors defined $\boldsymbol{p}_{i,x/y} = \boldsymbol{K}_{x/y}\boldsymbol{w}_{i,x/y}$, with $\boldsymbol{w}_{i,x/y}$ being the eigenvectors associated with the eigenvalues $\lambda_{i,x/y}$ of the closed-loop system (4.34) with (4.39), or (4.35) with (4.40), respectively. From the definition of the eigenvectors, the following linear system of equations is obtained

$$\underbrace{\begin{bmatrix} \lambda_i \boldsymbol{E} - \overline{\boldsymbol{A}}^* & \overline{\boldsymbol{B}}^* \\ & \\ \boldsymbol{e}_{j(i)}^T & \end{bmatrix}_{x/y}}_{\boldsymbol{D}_{i,x/y}} \begin{pmatrix} \boldsymbol{w}_i \\ \boldsymbol{p}_i \end{pmatrix}_{x/y} = \boldsymbol{0}. \tag{4.42}$$

The unit row vectors $\boldsymbol{e}_{j(i)}^T$, whose elements are all zero except for the jth element which is one, are added in (4.42) in order to make the system of equations solvable and allow to define additional requirements for the solution of (4.42). The goal is to suppress certain states in the eigenmodes $\exp(\lambda_i(t-t_0))\boldsymbol{w}_{i,x/y}$ of the closed-loop systems. A suitable requirement can be formulated as follows: The eigenvalue $\lambda_{i,x/y}$ should not influence the jth state with $i \in \{1,2,3,4,5,6\}$ and $j \in \{1,2,3,4,4,2\}$. The solution of (4.42) is obtained from $\begin{pmatrix} \boldsymbol{w}_i & \boldsymbol{p}_i \end{pmatrix}_{x/y}^T = \text{kernel}\left(\boldsymbol{D}_{i,x/y}\right)$ analytically using the six 7×8 matrices $\boldsymbol{D}_{i,x/y}$ yielding

$$\boldsymbol{K}_x = \begin{bmatrix} -a & -m_u \cdot l \cdot a \\ b & m_u \cdot l \cdot b \\ 0 & -k_{\omega_y} - I_{yy} \cdot d \\ g & -k_\theta + 2 \cdot m_u \cdot g \cdot l + I_{yy} \cdot e \\ c & m_u \cdot l \cdot c \\ 0 & I_{yy} \cdot f \end{bmatrix}^T, \tag{4.43}$$

with

$$\begin{aligned} a &= \lambda_3 + \lambda_4 + \lambda_5, & b &= \lambda_3\lambda_4 + \lambda_3\lambda_5 + \lambda_4\lambda_5, \\ c &= \lambda_3\lambda_4\lambda_5, & d &= \lambda_1 + \lambda_2 + \lambda_6, \\ e &= \lambda_1\lambda_2 + \lambda_1\lambda_6 + \lambda_2\lambda_6, & f &= \lambda_1\lambda_2\lambda_6. \end{aligned}$$

From the closed-loop equation for the x-direction

$$\overline{\boldsymbol{A}}_x^* - \overline{\boldsymbol{B}}_x^* \boldsymbol{K}_x = \begin{bmatrix} a & -b & 0 & 0 & -c & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & d & -e & 0 & -f \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}, \tag{4.44}$$

it can be seen that the translational and the rotational states are indeed decoupled if the controller (4.43) is used. In addition, the integrals in the controllers (4.29) - (4.31) result in zero steady-state error in position and orientation. A stability analysis of (4.34) - (4.37) considering time delay in the communication channel between the flying robot and the manipulator can be found in Appendix A9.

The controller gain $\boldsymbol{K}_y$ follows analogously, but is omitted here for the sake of brevity. The controllers for the subsystems (4.36) and (4.37) with gains $\boldsymbol{K}_z$ and $\boldsymbol{K}_\psi$ are again obtained using pole placement and with the integral of the position and the orientation error $\tilde{r}_z$ and $\tilde{\psi}$, respectively, as additional states.

4.2.6 Experimental evaluation

The landing system demonstrator is shown in Figure 4.3. It is composed of a torque-controlled DLR/KUKA Light Weight Robot (LWR) with seven DoF (see Section 2.2 for details). At the end-effector, a camera system for visual localization of the UAV, an electromagnet for attaching and releasing the aerial vehicle, and a force-torque sensor for measuring the interaction forces are mounted. The off-the-shelf quadrocopter AR.Drone 2.0 [171], which is used in this lab experiment, is equipped with a custom-built universal hinge that decouples UAV and end-effector orientation such that only translational forces are applied by the robot (see Figure 4.3). The model parameters of the quadrocopter can be found in Table 4.1. The attitude controller gains are those of the manufacturer and are identified using step responses about the three principal axes as $(k_\varphi, k_\theta, k_\psi, k_{\omega_x}, k_{\omega_y}, k_{\omega_z}) = (0.59, 0.66, 0.03, 0.04, 0.05, 0.03)$. The gravitational acceleration is assumed to be $g = 9.81\,\mathrm{m/s^2}$. All controllers are implemented using MATLAB/Simulink® and the communication with the quadrocopter is established using standard 2.4 GHz wireless LAN.

The gain $\boldsymbol{K}_1$ of the controller (4.22) in Section 4.2.3 is obtained by placing the poles at $\lambda_{i,1} \in \{-3, -4, -5\}$, which results in a subjectively fast motion of the LWR. The poles for the controllers (4.29) - (4.31) in Section 4.2.4 are chosen as $\lambda_{i,2/3} \in \{-5, -5, -5, -5, -5\}$ and $\lambda_{i,4} \in \{-3, -4, -5\}$ yielding $\boldsymbol{K}_{2/3}$ and $\boldsymbol{K}_4$, respectively. The pole $\lambda_{i,2/3}$ is chosen faster

Table 4.1: Model parameters of quadrocopter AR.Drone 2.0.

Mass m_u [kg]	Inertia $\boldsymbol{I}$ [kg m^2]	Hinge length l [m]
0.480	diag([0.006 0.007 0.012])	0.06

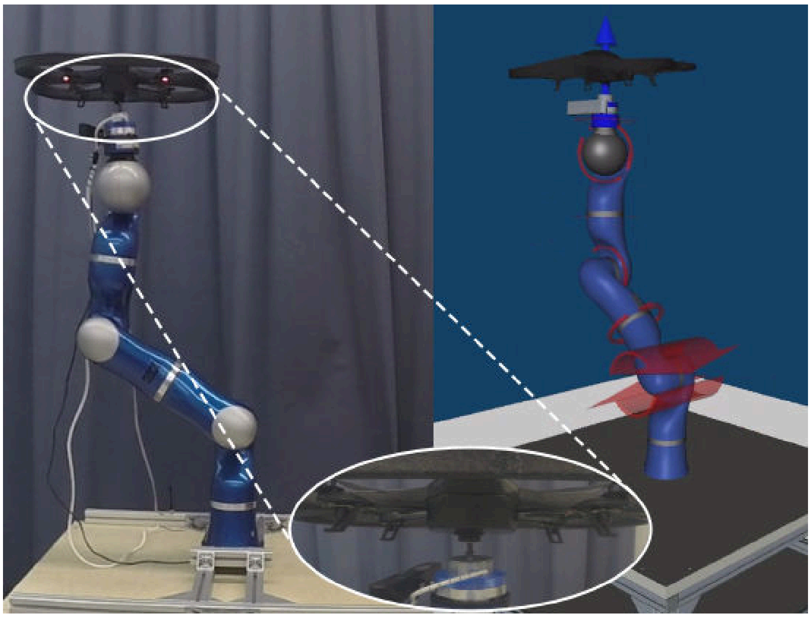


Figure 4.3: Robotic VTOL UAV landing system in the DLR Flying Robots lab (left) and a screenshot from the simulation environment (right). The enlarged detail shows the universal hinge and the electromagnet which connect the quadrocopter to the end-effector of the serial robotic manipulator. Copyright © 2016 IEEE [208].

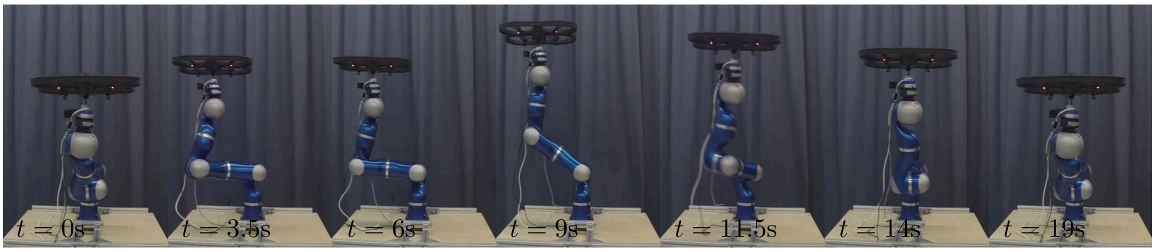


Figure 4.4: Picture sequence from the performed experiment with the controller from Section 4.2.5. The indicated time instants correspond to the plot in Figure 4.7. A short video is provided online [172] showing the robotic assistance system for flying robots and some of the conducted experiments. Copyright © 2016 IEEE [208].

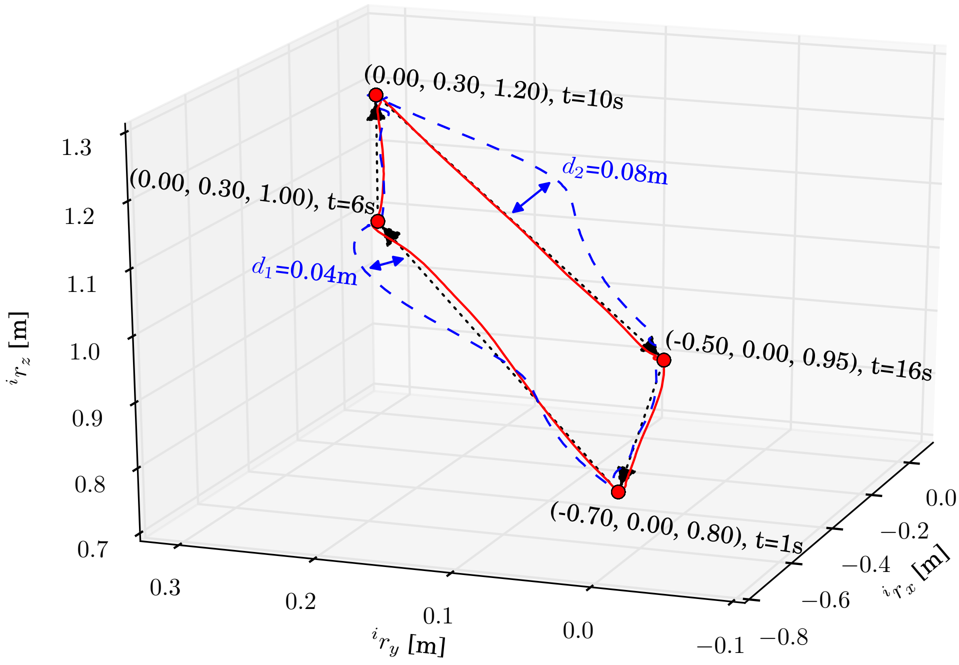
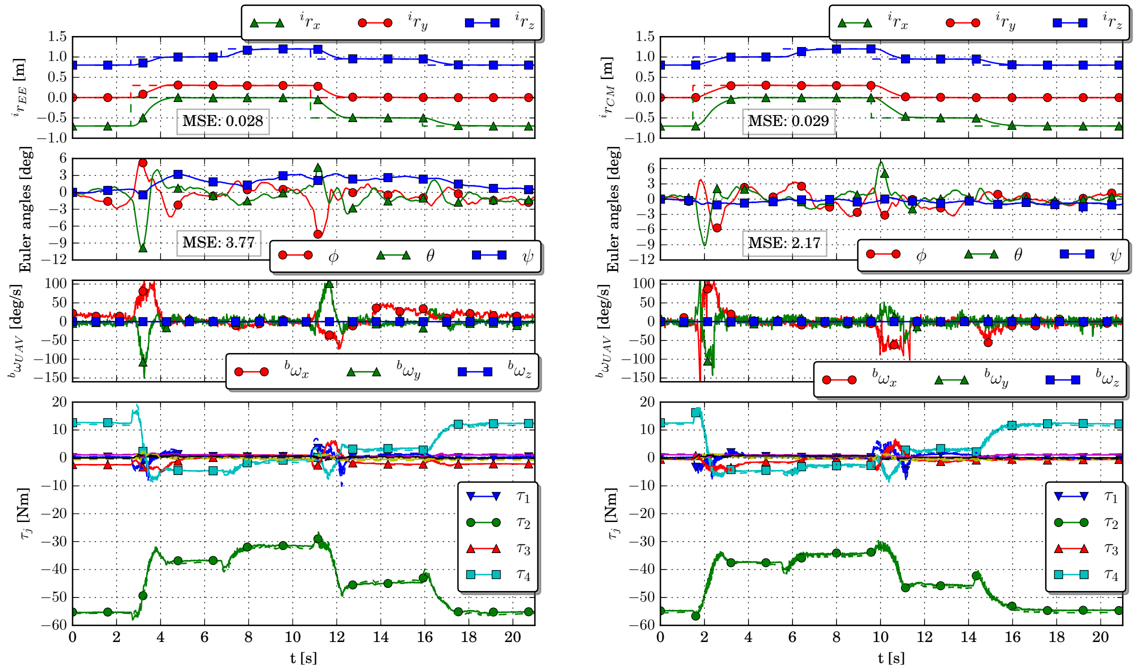


Figure 4.5: Path of ${}^i\boldsymbol{r}_{CM}$ in the experiments with the control laws (4.29) - (4.31) (—) and with the control laws (4.39), (4.40) (- -). The dotted line (···) is the direct connection between the waypoints and the arrows indicate the direction of the motion. For the controllers (4.39), (4.40) two maximum deviations d_1 and d_2 from the direct path between two waypoints are depicted. Copyright © 2016 IEEE [208].

than the open-loop poles to test if the orientation control is nevertheless better compared to the nominal manipulator controller (4.22). The poles for the controllers (4.39) and (4.40) from Section 4.2.5 are chosen with slightly different orientation eigenvalues $\lambda_{1,x/y}$, $\lambda_{2,x/y}$, and $\lambda_{6,x/y}$ compared to the open-loop, as $\lambda_{i,x/y} \in \{-3.5+9\mathrm{i}, -3.5-9\mathrm{i}, -3, -4, -5, -5\}$ and $\lambda_{i,z/\psi} \in \{-3, -3, -3\}$ yielding $\boldsymbol{K}_{x/y}$ and $\boldsymbol{K}_{z/\psi}$ respectively, to examine the influence of the decoupling in the closed-loop behaviour. It depends on the ratio between position and attitude gains, if either position or orientation is prioritized. If the gains for the translational motion are higher, the controller tries to keep the end-effector close to the desired position.



(a) Experimental results using the linear controller (4.22) from Section 4.2.3. The top row shows the trajectory of the robot's end-effector.

(b) Experimental results using the control laws (4.29) - (4.31) from Section 4.2.4. The top row shows the trajectory of the UAV's center of mass.

Figure 4.6: Experimental results using the three different linear state space control approaches. The last row shows the joint torques of the Light Weight Robot. Only the first four torques are labelled since the other three are close to zero. Copyright © 2016 IEEE [208].

This deteriorates the accuracy of the attitude control. On the other hand, if the gains for the rotational motion are higher, orientation control is more accurate (similar to an inverted pendulum), while the deviation from the desired position is increased. The end-effector of the robot is controlled such that it always points upwards, as shown in Figure 4.3, using the orientation controller (4.23) with gains $(k_{EE}, k_{\omega_{EE}}) = (12.0, 7.0)$. Alternatively, this tasks can be realized by the nullspace controller, which would result in task decoupling and increased authority of the end-effector task. For now, the joint redundancy of the LWR is resolved using the elbow field [71]. It constrains the elbow such that it points downwards and produces $\boldsymbol{\tau}_{nsp}$. The term $\boldsymbol{J}^T(\boldsymbol{\phi})\boldsymbol{\mu}_d(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}})$ in (4.24) is neglected in the implementation, because of its minor numerical value compared to the other terms.

All three controllers are tested with the same sequence of four position set-points as shown in the top row of Figure 4.6 and Figure 4.7 for the controllers from Section 4.2.3, Section 4.2.4, and Section 4.2.5. A picture sequence from the experiments with the controller from Section 4.2.5 is shown in Figure 4.4. No additional interpolator is used between the desired set-points. Note that for the controller (4.22) from Section 4.2.3 the set-point for the end-effector ${}^i\boldsymbol{r}_{EE,d}$ is commanded, while for controller from Section 4.2.4 and Section 4.2.5 the desired position of the UAV's center of mass ${}^i\boldsymbol{r}_{CM,d}$ is commanded. Every set-point is sent as soon as the previous waypoint is reached within 1 cm accuracy. Hence, the duration of the trajectory segments for the three different controllers varies, as can be seen in the top row of Figure 4.6 and 4.7. With the nominal controller (4.22), it takes about 19 s to execu-

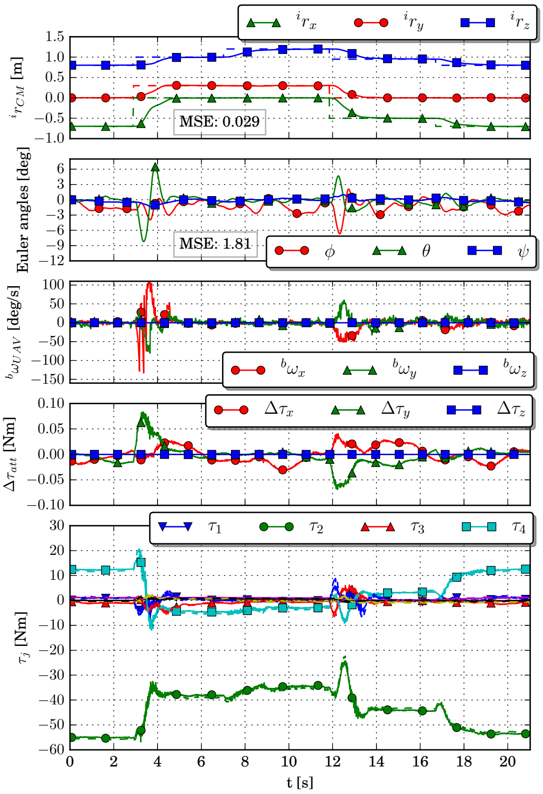


Figure 4.7: Experimental results using the control laws (4.39) and (4.40) from Section 4.2.5. The fourth row shows the attitude torques which are sent to the aerial vehicle. The last row shows the joint torques of the Light Weight Robot. Only the first four torques are labelled since the other three are close to zero. Copyright © 2016 IEEE [208].

tion the complete trajectory sequence, with the control laws (4.29) - (4.31) the duration is approximately 17 s and the control laws (4.39) and (4.40) result in the slowest motion with a duration of about 20 s. The mean squared error (MSE) of the deviation from the four position set-points is very similar for controller (4.22) (MSE: 0.028), the controllers (4.29), (4.30), (4.31) (MSE: 0.029), and the controllers (4.39), (4.40) (MSE: 0.029). However, the path of the UAV's center of mass in Figure 4.5 reveals a deviation from the direct connection between two waypoints of up to 8 cm with the control laws (4.39) and (4.40).

The desired orientation of the quadrocopter, written in Euler angles, is $(\varphi \quad \theta \quad \psi)^T = \mathbf{0}$ and its trajectory is shown in the second row of Figure 4.6 and 4.7. The MSE of the deviation from the desired orientation of the Euler angles φ, θ, ψ, depicted in Figure 4.6 and 4.7, shows that the precision of the orientation control using decoupled control (4.39), (4.40) is superior compared to the other two controllers (MSE: 1.81). The nominal controller (4.22) shows the worst orientation accuracy (MSE: 3.77). The third row of Figure 4.6 and 4.7 depicts the corresponding angular velocities of the quadrocopter measured by its on-board IMU. The desired and measured joint torques of the robot manipulator are depicted in the last row of Figure 4.6 and 4.7 and show equal tracking performance in all three experiments of the low-level torque controller of the LWR. The additional torque commands for the quadrocopter's attitude controller $\Delta\boldsymbol{\tau}_{att}$ are only shown in the fourth row of Figure 4.7 since $\Delta\boldsymbol{\tau}_{att}$ is only used with this controller (see also Figure 4.2).

The experimental results indicate that the linear controllers (4.29) - (4.31) designed using pole placement in Section 4.2.4 as well as the model decoupling controllers (4.39) and (4.40) from Section 4.2.5 both increase the performance in position and attitude control compared to the nominal manipulator controller (4.22) presented in Section 4.2.3. The controllers (4.29) - (4.31) can fulfill position and orientation control simultaneously while the weighting between both is defined by the eigenvalues of the closed-loop system. With the selected eigenvalues, the execution time of the test sequence is reduced by 2 s while the accuracy in orientation control is increased. The controllers (4.39) and (4.40) mitigate the effect of a change in the UAV's orientation on its center of mass position in the inertial frame and in addition generates correction torques for the flying robot's attitude. This leads to deviations from the direct path between two waypoints but to best performance in attitude control compared to the other two controllers. The reason for the deviation in position is, that in this experiment set-points are commanded instead of a continuous trajectory between the four points in Cartesian space. The attitude of the flying robot is regulated using the off-the-shelf controller implemented by the manufacturer. Closed-loop torque control is not available, which may be the reason why the disturbance is not compensated completely. In practice, the difference in performance of the presented linear control approaches highly depends on the acceleration control of the robot as well as on the torque control of the UAV. The controllers (4.29), (4.30), and (4.31) only use state feedback from the flying robot, while for the controllers (4.39) and (4.40) it is important that the computed torque $\Delta\boldsymbol{\tau}_{att}$ is really produced by the flying robot. Therefore, the controllers (4.29) - (4.31) from Section 4.2.4 are beneficial for applications where no accurate UAV torque controller is available.

4.3 Nonlinear backstepping control

Here, the same decomposition of the system dynamics as in the previous section is used. However, a nonlinear backstepping controller is designed to control the pose of the flying robot attached at the end-effector of the robot arm. First, the backstepping method and quaternion backstepping are introduced, then the novel control approach is presented and evaluated in experiments with the robotic assistance system demonstrator.

4.3.1 Backstepping method

Backstepping or integrator backstepping is a nonlinear control method ([87], p. 589) suitable for dynamical systems given in strict feedback form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{w}, \tag{4.45}$$

$$\dot{\boldsymbol{w}} = \boldsymbol{f}_1(\boldsymbol{x},\boldsymbol{w}) + \boldsymbol{B}_1(\boldsymbol{x},\boldsymbol{w})\boldsymbol{u}. \tag{4.46}$$

Proposition 4.1. *Assume that asymptotic stability of (4.45) under the virtual control law* $\boldsymbol{\alpha}$ *with* $\boldsymbol{w} = \boldsymbol{\alpha}(\boldsymbol{x})$ *is confirmed using the Lyapunov function* $V_0(\boldsymbol{x})$*. If* $\boldsymbol{B}_1(\boldsymbol{x},\boldsymbol{w})$ *in (4.46) is invertible, then it is possible to use the backstepping control law*

$$\boldsymbol{u}(\boldsymbol{x},\boldsymbol{w}) = \boldsymbol{B}_1^{-1}(\boldsymbol{x},\boldsymbol{w})\left(\dot{\boldsymbol{\alpha}} - \boldsymbol{B}^T(\boldsymbol{x})\left(\frac{\partial V_0(\boldsymbol{x})}{\partial \boldsymbol{x}}\right) - \boldsymbol{K}(\boldsymbol{w}-\boldsymbol{\alpha}) - \boldsymbol{f}_1(\boldsymbol{x},\boldsymbol{w})\right), \tag{4.47}$$

with $\dot{\boldsymbol{\alpha}} = \frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{x}}(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{w})$*, to render the equilibrium* $\boldsymbol{x}_e$ *of the dynamical system (4.45, 4.46) asymptotically stable.*

Proof. Consider the Lyapunov function

$$V(\boldsymbol{x}, \boldsymbol{w}) = V_0(\boldsymbol{x}) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{\alpha})^T(\boldsymbol{w} - \boldsymbol{\alpha}) \geq 0 \tag{4.48}$$

with time derivative

$$\dot{V}(\boldsymbol{x}, \boldsymbol{w}) = \left(\frac{\partial V_0(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)^T \underbrace{(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{w})}_{\dot{\boldsymbol{x}}} + (\boldsymbol{w} - \boldsymbol{\alpha})^T(\dot{\boldsymbol{w}} - \dot{\boldsymbol{\alpha}}). \tag{4.49}$$

Inserting the controller (4.47) in (4.46) yields

$$\dot{\boldsymbol{w}} = \dot{\boldsymbol{\alpha}} - \boldsymbol{B}^T(\boldsymbol{x})\left(\frac{\partial V_0(\boldsymbol{x})}{\partial \boldsymbol{x}}\right) - \boldsymbol{K}(\boldsymbol{w} - \boldsymbol{\alpha}). \tag{4.50}$$

Using the above result in (4.49) and rearranging leads to

$$\dot{V}(\boldsymbol{x}, \boldsymbol{w}) = \left(\frac{\partial V_0(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)^T (\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{\alpha}) - (\boldsymbol{w} - \boldsymbol{\alpha})^T \boldsymbol{K}(\boldsymbol{w} - \boldsymbol{\alpha}), \tag{4.51}$$

for which it follows because of $\left(\frac{\partial V_0(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)^T (\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{\alpha}) \leq 0$ with a suitably chosen $\boldsymbol{\alpha}$ and a positive definite $\boldsymbol{K}$ that

$$\dot{V}(\boldsymbol{x}, \boldsymbol{w}) \leq 0. \tag{4.52}$$

□

4.3.2 Quaternion backstepping

In order to avoid representation singularities, quaternions shall be used as attitude representation. Since the kinematics of a quaternion $\boldsymbol{q} = (q_w \quad q_x \quad q_y \quad q_z)^T = (\eta \quad \boldsymbol{\epsilon}^T)^T$ are given in strict feedback form (4.45, 4.46), the backstepping approach (4.47) can be applied directly. The kinematics (cf. (2.25)) with input $\boldsymbol{u}$ read

$$\dot{\boldsymbol{q}} = \frac{1}{2}\begin{bmatrix} -\boldsymbol{\epsilon}^T \\ \eta \boldsymbol{E}_{3\times3} + \boldsymbol{S}(\boldsymbol{\epsilon}) \end{bmatrix} \boldsymbol{\omega}_u = \frac{1}{2}\boldsymbol{Q}_q(\boldsymbol{q})\,\boldsymbol{\omega}_u, \tag{4.53}$$

$$\dot{\boldsymbol{\omega}}_u = \boldsymbol{u}. \tag{4.54}$$

The attitude error $\boldsymbol{q}_e$ is defined using quaternion multiplication (see (2.23) and (2.24)), hence

$$\boldsymbol{q}_e = \begin{pmatrix} \eta_e & \boldsymbol{\epsilon}_e^T \end{pmatrix}^T = \boldsymbol{q}_d^{-1} \otimes \boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_d^{-1} & \boldsymbol{Q}_q(\boldsymbol{q}_d^{-1}) \end{bmatrix} \boldsymbol{q}, \tag{4.55}$$

which is equal to the rotation from the desired attitude $\boldsymbol{q}_d$ to the current attitude $\boldsymbol{q}$ [173]. Due to the ambiguity of quaternions mentioned in Section 2.4.1, the backstepping variables have to be chosen carefully to avoid unwinding [174], i.e. it needs to be assured that from the two possible rotations always the shorter one is performed. Here, only the regulation

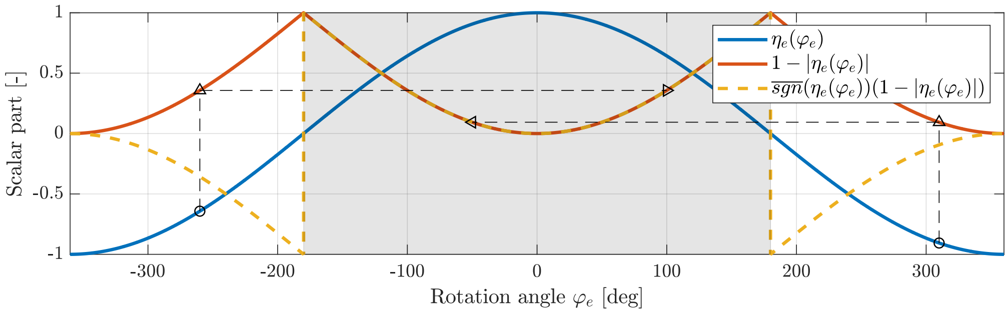


Figure 4.8: Illustration of the terms $1-|\eta_e|$ and $\overline{\mathrm{sgn}}(\eta_e)(1-|\eta_e|)$ in (4.56) and (4.62), respectively. Consider a quaternion error $\boldsymbol{q}_e$ with arbitrary rotation axis $\boldsymbol{\epsilon}_e$ and rotation angle $\varphi_e \in [-2\pi, 2\pi]$. Then, applying the above terms restricts the angle φ_e to $\hat{\varphi}_e = 2\overline{\mathrm{sgn}}(\eta_e)\arccos(1-|\eta_e|)$ (gray area). For example, the dashed lines show the evolution of $\varphi_e = -260\,\mathrm{deg}$, which ends up at $\hat{\varphi}_e = 100\,\mathrm{deg}$, and $\varphi_e = 300\,\mathrm{deg}$, which ends up at $\hat{\varphi}_e = -60\,\mathrm{deg}$.

case (or set-point control), i.e. a constant attitude reference, is considered. A suitable choice of the backstepping variables is

$$\begin{aligned} \boldsymbol{z}_1 &= \begin{pmatrix} 1-|\eta_e| & \boldsymbol{\epsilon}_e^T \end{pmatrix}^T, & \dot{\boldsymbol{z}}_1 &= \underbrace{\frac{1}{2}\begin{bmatrix} \overline{\mathrm{sgn}}(\eta_e)\boldsymbol{\epsilon}_e^T \\ \eta_e \boldsymbol{E}_{3\times 3} + \boldsymbol{S}(\boldsymbol{\epsilon}_e) \end{bmatrix}}_{=:\boldsymbol{Z}} \boldsymbol{\omega}_u, \\ \boldsymbol{z}_2 &= \boldsymbol{\omega}_u - \boldsymbol{\alpha}, & \dot{\boldsymbol{z}}_2 &= \dot{\boldsymbol{\omega}}_u - \dot{\boldsymbol{\alpha}}. \end{aligned} \tag{4.56}$$

The augmented quaternion error is illustrated in Figure 4.8 and the augmented signum function is given by

$$\overline{\mathrm{sgn}}(\eta_e) = \begin{cases} -1, & \eta_e < 0, \\ 1, & \eta_e \geq 0. \end{cases} \tag{4.57}$$

Consider the Lyapunov functions

$$V_0 = \frac{k}{2}\boldsymbol{z}_1^T\boldsymbol{z}_1 > 0, \tag{4.58}$$

$$V = V_0 + \frac{1}{2}\boldsymbol{z}_2^T\boldsymbol{z}_2 > 0. \tag{4.59}$$

A virtual control input $\boldsymbol{\alpha}$, which results in asymptotic stability of the equilibrium $\boldsymbol{\omega}_u = \boldsymbol{0}$ is

$$\boldsymbol{\alpha} = -\boldsymbol{K}_1\boldsymbol{Z}^T\boldsymbol{z}_1. \tag{4.60}$$

For the latter and with $k > 0$ and $\boldsymbol{K}_1 > \boldsymbol{0}$, it is straightforward to show that

$$\dot{V}_0 = k\boldsymbol{z}_1^T\dot{\boldsymbol{z}}_1 = -k\boldsymbol{z}_1^T\boldsymbol{Z}\boldsymbol{K}_1\boldsymbol{Z}^T\boldsymbol{z}_1 + k\boldsymbol{z}_1^T\boldsymbol{Z}\boldsymbol{z}_2. \tag{4.61}$$

Then, according to (4.47) from Section 4.3.1, the backstepping controller is derived as

$$\boldsymbol{u} = \dot{\boldsymbol{\alpha}} - k\boldsymbol{Z}^T\boldsymbol{z}_1 - \boldsymbol{K}_2\boldsymbol{z}_2, \tag{4.62}$$

for which it holds that

$$\dot{V} = \dot{V}_0 + \boldsymbol{z}_2^T\dot{\boldsymbol{z}}_2 = -k\boldsymbol{z}_1^T\boldsymbol{Z}\boldsymbol{K}_1\boldsymbol{Z}^T\boldsymbol{z}_1 \underbrace{+k\boldsymbol{z}_1^T\boldsymbol{Z}\boldsymbol{z}_2 - k\boldsymbol{z}_2^T\boldsymbol{Z}^T\boldsymbol{z}_1}_{=0} - \boldsymbol{z}_2^T\boldsymbol{K}_2\boldsymbol{z}_2 \leq 0. \tag{4.63}$$

Thus, stability and convergence of the orientation error may be concluded applying LaSalle's invariance principal ([87], p. 128). Note that $\boldsymbol{Z}$ is a state-dependent gain matrix and that the control $\boldsymbol{\alpha}$ has a discontinuity at $\eta_e = 0$, due to $\boldsymbol{Z}^T \boldsymbol{z}_1 = \overline{\mathrm{sgn}}(\eta_e)\boldsymbol{\epsilon}_e$. This is necessary in order to disambiguate the quaternion and avoid unwinding [174]. Asymptotic stability can therefore only be shown locally for $\eta_e \neq 0$.

4.3.3 Additional control input

Recall the separate dynamics model derived in Section 4.1. Inserting (4.5) and (4.6) in (4.4) yields the concise formulation

$$\underbrace{\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & m_u \boldsymbol{E} \end{bmatrix}}_{\boldsymbol{M}_u} \begin{pmatrix} \dot{\boldsymbol{\omega}}_u \\ \dot{\boldsymbol{v}}_u \end{pmatrix} = \underbrace{\begin{pmatrix} \boldsymbol{S}(\boldsymbol{I}\boldsymbol{\omega}_u)\boldsymbol{\omega}_u + \boldsymbol{\tau}_{att} \\ -m_u g \boldsymbol{e}_3 + T \boldsymbol{R}_{ui}^T \boldsymbol{e}_3 \end{pmatrix}}_{\boldsymbol{a}_u} + \underbrace{\begin{bmatrix} \boldsymbol{S}(\boldsymbol{\rho})\boldsymbol{R}_{ui} \\ \boldsymbol{E} \end{bmatrix}}_{\boldsymbol{B}_u} \boldsymbol{F}_{EE}, \tag{4.64}$$

where the inertia matrix of the flying robot is denoted as $\boldsymbol{M}_u$, the input matrix as $\boldsymbol{B}_u$, and the rest of the right-hand side of (4.64) as $\boldsymbol{a}_u$.

The input matrix $\boldsymbol{B}_u$ in (4.64) is not invertible due to the underactuation of the system, which prevents this model from being directly utilized for nonlinear backstepping design.[3] Therefore, an additional torque input $\Delta\boldsymbol{\tau}_{att}$ is appended, which is sent to the flying robot and added to $\boldsymbol{\tau}_{att}$, such that the control input becomes

$$\boldsymbol{u} = \begin{pmatrix} \boldsymbol{u}_1^T & \boldsymbol{u}_2^T \end{pmatrix}^T = \begin{pmatrix} \Delta\boldsymbol{\tau}_{att}^T & \boldsymbol{F}_{EE}^T \end{pmatrix}^T. \tag{4.67}$$

Hence, the system becomes fully actuated and the augmented input matrix $\overline{\boldsymbol{B}}_u$ is invertible[4] with

$$\overline{\boldsymbol{B}}_u = \begin{bmatrix} \boldsymbol{E} & \boldsymbol{S}(\boldsymbol{\rho})\boldsymbol{R}_{ui} \\ \boldsymbol{0} & \boldsymbol{E} \end{bmatrix}. \tag{4.68}$$

4.3.4 Backstepping controller design

The controller for the system depicted in Fig. 4.9 is derived following the procedure presented in Section 4.3.2 and [175]. It is extended to include both the rotational and the translational dynamics of the flying robot. As depicted in Figure 4.10, the final control approach combines orientation and position control and the latter is used in Section 4.3.5 to extend the classical impedance controller (2.87) from Section 2.6.

[3] A similar conclusion may be drawn if the controllability of the analogous linear time-invariant (LTI) system

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{4.65}$$

with $\boldsymbol{A} = \boldsymbol{M}_u^{-1}$ and $\boldsymbol{B} = \boldsymbol{M}_u^{-1}\boldsymbol{B}_u$ is examined. For the controllability matrix ([167], p. 291)

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B} & \dots & \boldsymbol{A}^{n-1}\boldsymbol{B} \end{bmatrix}, \tag{4.66}$$

with $n = 6$, it follows that $\mathrm{rank}(\boldsymbol{C}) = 5 < n$, i.e. the system is not controllable.

[4] For $\boldsymbol{B} = \boldsymbol{M}_u^{-1}\overline{\boldsymbol{B}}_u$, it follows that $\mathrm{rank}(\boldsymbol{C}) = 6 = n$, i.e. the system is controllable.

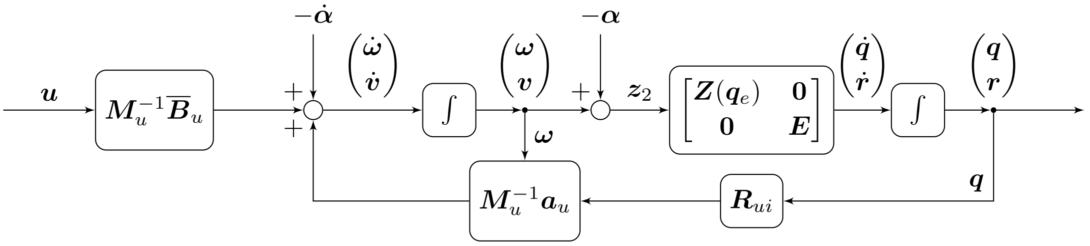

$-\dot{\boldsymbol{\alpha}}$
$-\boldsymbol{\alpha}$
$\boldsymbol{u}$
$\boldsymbol{M}_u^{-1}\overline{\boldsymbol{B}}_u$
$+$
$+$
$\begin{pmatrix}\dot{\boldsymbol{\omega}}\\ \dot{\boldsymbol{v}}\end{pmatrix}$
$\int$
$\begin{pmatrix}\boldsymbol{\omega}\\ \boldsymbol{v}\end{pmatrix}$
$\boldsymbol{\omega}$
$+$
$\boldsymbol{z}_2$
$\begin{bmatrix}\boldsymbol{Z}(\boldsymbol{q}_e) & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{E}\end{bmatrix}$
$\begin{pmatrix}\dot{\boldsymbol{q}}\\ \dot{\boldsymbol{r}}\end{pmatrix}$
$\int$
$\begin{pmatrix}\boldsymbol{q}\\ \boldsymbol{r}\end{pmatrix}$
$\boldsymbol{M}_u^{-1}\boldsymbol{a}_u$
$\boldsymbol{R}_{ui}$
$\boldsymbol{q}$

Figure 4.9: Flow diagram of the flying robot dynamics for backstepping controller design.

The first backstepping variable $\boldsymbol{z}_1$ is chosen to be

$$\boldsymbol{z}_1 = \begin{pmatrix} 1-|\eta_e| & \boldsymbol{\epsilon}_e^T & \boldsymbol{r}_e^T \end{pmatrix}^T . \tag{4.69}$$

The Cartesian position error $\boldsymbol{r}_e = \boldsymbol{r} - \boldsymbol{r}_d$ is defined as the difference between current position $\boldsymbol{r}$ and desired position $\boldsymbol{r}_d$ of the end-effector EE and the term $1-|\eta_e|$ is again used to avoid unwinding. A graphical explanation of the latter is given in Figure 4.8. Then, the first derivative of (4.69) with respect to time is

$$\dot{\boldsymbol{z}}_1 = \underbrace{\begin{bmatrix}\boldsymbol{Z}(\boldsymbol{q}_e)_{4\times 3} & \boldsymbol{0}_{4\times 3}\\ \boldsymbol{0}_{3\times 3} & \boldsymbol{E}_{3\times 3}\end{bmatrix}}_{=:\boldsymbol{W}} \begin{pmatrix}\boldsymbol{\omega}_u\\ \boldsymbol{v}_u\end{pmatrix} . \tag{4.70}$$

Next, the Lyapunov function candidate

$$V_1 = \frac{k_3}{2}\boldsymbol{z}_1^T\boldsymbol{z}_1 > 0 \tag{4.71}$$

is introduced, which is positive definite for $k_3 > 0$. Its derivative with respect to time is

$$\dot{V}_1 = k_3\boldsymbol{z}_1^T\dot{\boldsymbol{z}}_1 = k_3\boldsymbol{z}_1^T\boldsymbol{W}\begin{pmatrix}\boldsymbol{\omega}_u\\ \boldsymbol{v}_u\end{pmatrix} . \tag{4.72}$$

Using a virtual control input $\boldsymbol{\alpha}$, the second backstepping variable $\boldsymbol{z}_2$ is defined as

$$\boldsymbol{z}_2 = \begin{pmatrix}\boldsymbol{\omega}_u\\ \boldsymbol{v}_u\end{pmatrix} - \boldsymbol{\alpha}. \tag{4.73}$$

An intermediate control law, that preserves asymptotic stability for the equilibrium $\boldsymbol{\omega}_u = \boldsymbol{v}_u = \boldsymbol{0}$ of (4.73) is

$$\boldsymbol{\alpha} = -\boldsymbol{K}_1\boldsymbol{W}^T\boldsymbol{z}_1. \tag{4.74}$$

Therein, $\boldsymbol{K}_1 > 0$ is a positive definite diagonal gain matrix. Hence, (4.72) becomes

$$\dot{V}_1 = \underbrace{-k_3\boldsymbol{z}_1^T\boldsymbol{W}\boldsymbol{K}_1\boldsymbol{W}^T\boldsymbol{z}_1}_{\leq 0} + k_3\boldsymbol{z}_1^T\boldsymbol{W}\boldsymbol{z}_2. \tag{4.75}$$

Next, the outer-loop controller as depicted in Figure 4.9 is designed. The derivative of (4.73) is found as

$$\dot{\boldsymbol{z}}_2 = \begin{pmatrix}\dot{\boldsymbol{\omega}}_u\\ \dot{\boldsymbol{v}}_u\end{pmatrix} - \dot{\boldsymbol{\alpha}}, \tag{4.76}$$

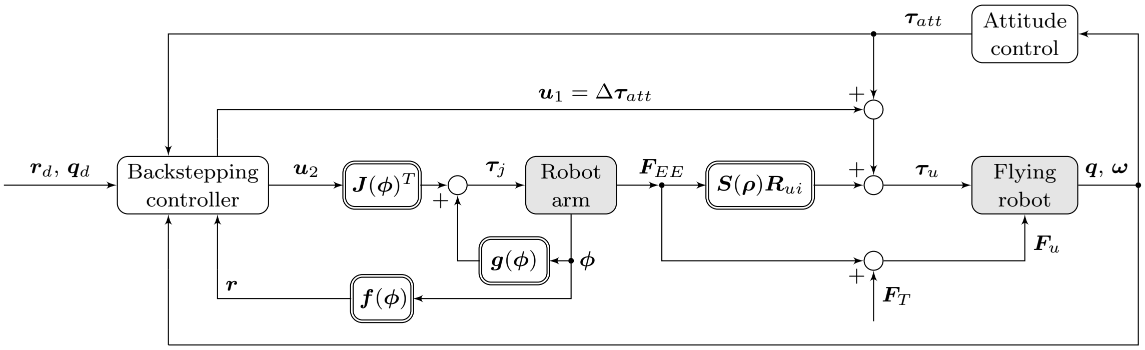

$\boldsymbol{\tau}_{att}$
Attitude control
$\boldsymbol{u}_1 = \Delta\boldsymbol{\tau}_{att}$
+
+
$\boldsymbol{r}_d$, $\boldsymbol{q}_d$
Backstepping controller
$\boldsymbol{u}_2$
$\boldsymbol{J}(\boldsymbol{\phi})^T$
+
$\boldsymbol{\tau}_j$
Robot arm
$\boldsymbol{F}_{EE}$
$\boldsymbol{S}(\boldsymbol{\rho})\boldsymbol{R}_{ui}$
$\boldsymbol{\tau}_u$
Flying robot
$\boldsymbol{q}$, $\boldsymbol{\omega}$
$\boldsymbol{g}(\boldsymbol{\phi})$
$\boldsymbol{\phi}$
$\boldsymbol{F}_u$
$\boldsymbol{r}$
$\boldsymbol{f}(\boldsymbol{\phi})$
+
$\boldsymbol{F}_T$

Figure 4.10: Integration of the backstepping controller in the proposed framework for coordinated control of robot manipulator and flying robot.

which can be rewritten using (4.64) to

$$\boldsymbol{M}_u \dot{\boldsymbol{z}}_2 = \boldsymbol{a}_u + \overline{\boldsymbol{B}}_u \boldsymbol{u} - \boldsymbol{M}_u \dot{\boldsymbol{\alpha}}. \tag{4.77}$$

A possible Lyapunov function candidate for the closed-loop system is then

$$V_2 = V_1 + \frac{1}{2}\boldsymbol{z}_2^T \boldsymbol{M}_u \boldsymbol{z}_2 > 0, \tag{4.78}$$

which is positive definite and whose first derivative with respect to time is

$$\dot{V}_2 = \dot{V}_1 + \boldsymbol{z}_2^T \boldsymbol{M}_u \dot{\boldsymbol{z}}_2. \tag{4.79}$$

Inserting (4.77) in (4.79) yields

$$\begin{aligned} \dot{V}_2 &= \dot{V}_1 + \boldsymbol{z}_2^T \boldsymbol{a}_u + \boldsymbol{z}_2^T \overline{\boldsymbol{B}}_u \boldsymbol{u} - \boldsymbol{z}_2^T \boldsymbol{M}_u \dot{\boldsymbol{\alpha}} \\ &= -k_3 \boldsymbol{z}_1^T \boldsymbol{W} \boldsymbol{K}_1 \boldsymbol{W}^T \boldsymbol{z}_1 + k_3 \boldsymbol{z}_1^T \boldsymbol{W} \boldsymbol{z}_2 + \boldsymbol{z}_2^T \boldsymbol{a}_u + \boldsymbol{z}_2^T \overline{\boldsymbol{B}}_u \boldsymbol{u} - \boldsymbol{z}_2^T \boldsymbol{M}_u \dot{\boldsymbol{\alpha}}. \end{aligned} \tag{4.80}$$

The aim is to render (4.80) negative definite, hence, the backstepping control law

$$\boldsymbol{u} = \overline{\boldsymbol{B}}_u^{-1}(-\boldsymbol{K}_2 \boldsymbol{z}_2 - \boldsymbol{a}_u + \boldsymbol{M}_u \dot{\boldsymbol{\alpha}} - k_3 \boldsymbol{W}^T \boldsymbol{z}_1) \tag{4.81}$$

is derived, wherein $\boldsymbol{K}_2 > 0$ is another positive definite diagonal gain matrix. Finally, inserting (4.81) into (4.80) yields the Lyapunov function for the complete system (4.64) under the controller (4.81)

$$\dot{V}_2 = -k_3 \boldsymbol{z}_1^T \boldsymbol{W} \boldsymbol{K}_1 \boldsymbol{W}^T \boldsymbol{z}_1 - \boldsymbol{z}_2^T \boldsymbol{K}_2 \boldsymbol{z}_2 \leq 0, \tag{4.82}$$

with $\boldsymbol{K}_1 > 0$, $\boldsymbol{K}_2 > 0$ and $k_3 > 0$. Note, that the quaternion part of $\boldsymbol{W}$ is an additional state-dependent gain matrix. Asymptotic stability of the equilibrium $\boldsymbol{r}_e = \boldsymbol{0}$, $\boldsymbol{q}_e = (1 \quad 0 \quad 0 \quad 0)^T$, $\boldsymbol{\omega}_u = \boldsymbol{v}_u = \boldsymbol{0}$ can be shown by applying the LaSalle invariance theorem ([87], p. 128).

Remark 1: Due to

$$\boldsymbol{W}^T \boldsymbol{z}_1 = \begin{bmatrix} \overline{\operatorname{sgn}}(\eta_e)\boldsymbol{\epsilon}_e \\ \boldsymbol{E}_{3\times 3}\boldsymbol{r}_e \end{bmatrix}, \tag{4.83}$$

the control $\boldsymbol{u}$ can jump at the transition from $\eta_e > 0$ to $\eta_e < 0$ and vice versa. Asymptotic stability can therefore only be guaranteed locally for $\eta_e \neq 0$.

Remark 2: For implementing the controller (4.81), the derivative of the intermediate control law (4.74) with respect to time can be computed from attitude, position, and velocity measurements as

$$\dot{\boldsymbol{\alpha}} = -\boldsymbol{K}_1\left(\dot{\boldsymbol{W}}^T\boldsymbol{z}_1 + \boldsymbol{W}^T\dot{\boldsymbol{z}}_1\right) = -\boldsymbol{K}_1\left(\left(\frac{\partial\boldsymbol{W}^T}{\partial\eta}\dot{\eta} + \frac{\partial\boldsymbol{W}^T}{\partial\boldsymbol{\epsilon}}\dot{\boldsymbol{\epsilon}}\right)\boldsymbol{z}_1 + \boldsymbol{W}^T\boldsymbol{W}\begin{pmatrix}\boldsymbol{\omega}_u\\ \boldsymbol{v}_u\end{pmatrix}\right). \tag{4.84}$$

4.3.5 Extended manipulator controller

The backstepping controller (4.81) presented above generates a torque $\Delta\boldsymbol{\tau}_{att} = \boldsymbol{u}_1$ that is sent to the flying robot and a force $\boldsymbol{F}_{EE} = \boldsymbol{u}_2$ that should be applied by the manipulator's end-effector to the UAV. The latter follows from (4.81) as

$$\boldsymbol{F}_{EE} = -(\boldsymbol{K}_{1,r}\boldsymbol{K}_{2,r} + k_3\boldsymbol{E})\boldsymbol{r}_e - (m_u\boldsymbol{K}_{1,r} + \boldsymbol{K}_{2,r})\,\boldsymbol{v}_u + \underbrace{m_u g\boldsymbol{e}_3 - T\boldsymbol{R}_{iu}\boldsymbol{e}_3}_{\boldsymbol{F}_{FTS,F_z}}. \tag{4.85}$$

Note that since the thrust force T of a flying robot is not directly measured in practice, it is convenient to use the force $\boldsymbol{F}_{FTS,F_z}$ measured by the force-torque sensor (FTS) at the end-effector for feedback control. The positive diagonal matrices $\boldsymbol{K}_{1,r}$ and $\boldsymbol{K}_{1,r}$ as well as the scalar k_3 are the design parameters of the backstepping controller for the translational direction.

The force (4.85) can be mapped to robot joint torques, as presented in Section 2.6 for the classical Cartesian impedance controller (2.87) via

$$\boldsymbol{\tau}_j = \boldsymbol{J}^T(\boldsymbol{\phi})\boldsymbol{\Lambda}(\boldsymbol{\phi})\ddot{\boldsymbol{x}}_d + \boldsymbol{J}^T(\boldsymbol{\phi})\boldsymbol{\mu}_d(\boldsymbol{\phi},\dot{\boldsymbol{\phi}}) + \boldsymbol{J}^T(\boldsymbol{\phi})\begin{pmatrix}\boldsymbol{F}_{EE}\\ \boldsymbol{\tau}_{ori}\end{pmatrix} + \boldsymbol{g}(\boldsymbol{\phi}) + \boldsymbol{\tau}_{nsp}, \tag{4.86}$$

where $\boldsymbol{\tau}_{nsp}$ is a nullspace torque, $\boldsymbol{g}(\boldsymbol{\phi})$ is the gravity compensation of the manipulator, and $\boldsymbol{\tau}_{ori}$ allows to control the orientation of the end-effector independently from the desired force (4.85), and thus independently from the orientation of the flying robot. End-effector orientation control is implemented using the control law (4.23) from Section 4.2.3. The desired acceleration $\ddot{\boldsymbol{x}}_d$ in the additional tracking term $\boldsymbol{\Lambda}(\boldsymbol{\phi})\ddot{\boldsymbol{x}}_d$ can be generated using a Cartesian interpolator. The control laws (4.85) and (4.23) can be expressed in general as

$$\boldsymbol{F}_\tau = \begin{pmatrix}\boldsymbol{F}_{EE}\\ \boldsymbol{\tau}_{ori}\end{pmatrix} = -\boldsymbol{K}\boldsymbol{x}_e - \boldsymbol{D}\dot{\boldsymbol{x}}. \tag{4.87}$$

Inserting (4.87) with gravity compensation and the remaining tracking term $\boldsymbol{\Lambda}(\boldsymbol{\phi})\ddot{\boldsymbol{x}}_d$ in the task space dynamics (2.53) yields the dynamical relationship for the forces at the end-effector

$$\boldsymbol{\Lambda}(\boldsymbol{\phi})\ddot{\boldsymbol{x}}_e + (\boldsymbol{\mu}(\boldsymbol{\phi},\dot{\boldsymbol{\phi}}) + \boldsymbol{D})\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x}_e = \boldsymbol{F}_{ext}. \tag{4.88}$$

Compared to the manipulator controller (4.24) from Section 4.2.3, now forces are mapped to joint torques instead of accelerations. The control law (4.86) with (4.85) includes the nonlinear dynamics of the flying robot, instead of the linearized dynamics used throughout Section 4.2. Furthermore, the controller (4.86) results in a compliant behaviour (e.g. like a mass-spring-damper system) in Cartesian space, which is advantageous during physical interaction between manipulator and flying robot.

4.3.6 Experimental results

In order to validate the proposed control strategy, experiments using the demonstrator depcited in Figure 4.11 and the off-the-shelf quadrocopter AR.Drone 2.0 [171] are preformed. The setup is identical to Section 4.2.6. The quadrocopter is connected to the end-effector of the robot via the universal hinge shown in Figure 4.11. Again, all controllers are implemented using MATLAB/Simulink® and communication with the AR.Drone is established via standard 2.4 GHz wireless LAN.

Both the performance of the orientation control of the flying robot and the Cartesian position control of the robot manipulator arm are evaluated. The conditions used in the three

Table 4.2: Conditions considered in the three experiments E1, E2, and E3.

Experiment	Conditions
E1	$\Delta\boldsymbol{\tau}_{att} = \mathbf{0}$, $\ddot{\boldsymbol{x}}_d = \mathbf{0}$, $\boldsymbol{F}_{FTS} = \mathbf{0}$
E2	Using $\Delta\boldsymbol{\tau}_{att}$ and $\ddot{\boldsymbol{x}}_d$; $\boldsymbol{F}_{FTS} = \mathbf{0}$
E3	Using $\Delta\boldsymbol{\tau}_{att}$, $\ddot{\boldsymbol{x}}_d$, and $\boldsymbol{F}_{FTS}$

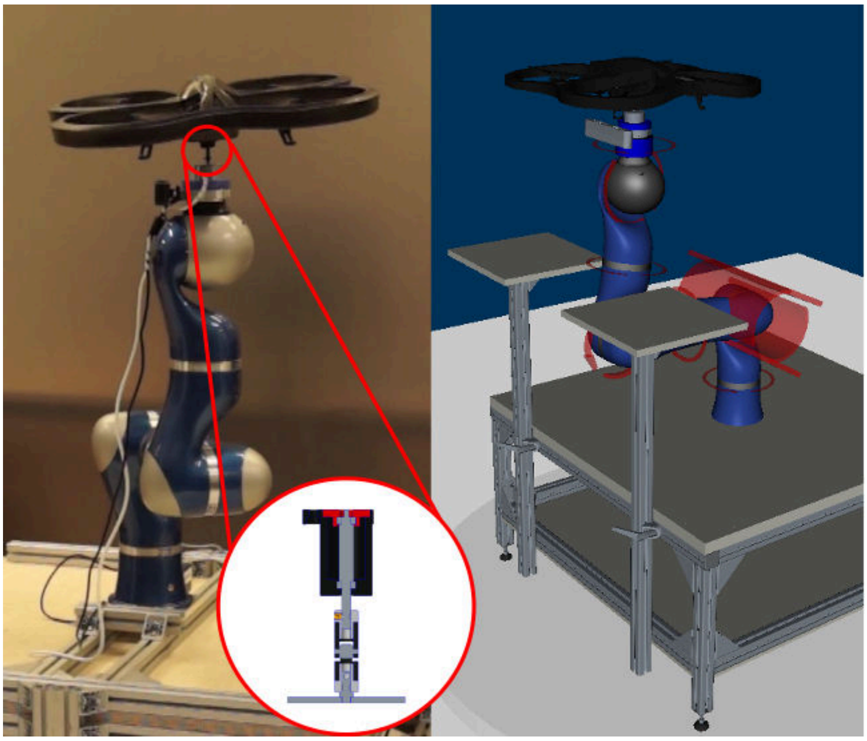
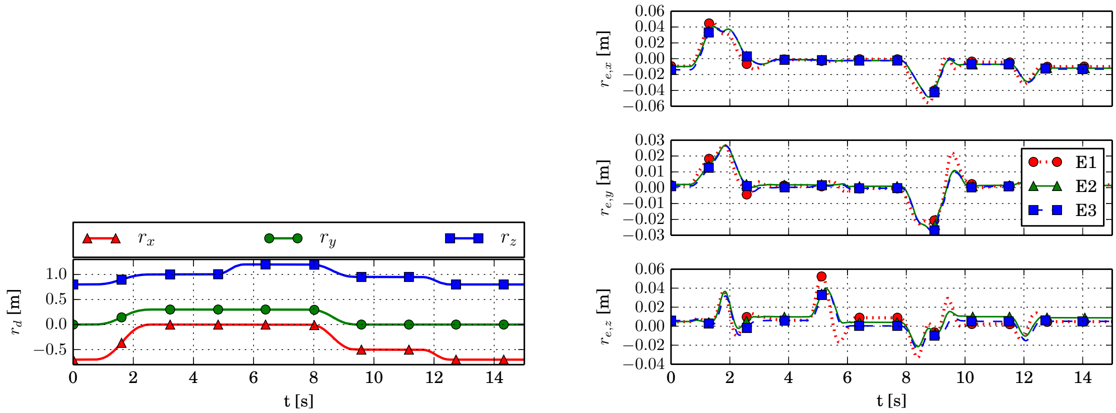


Figure 4.11: Robotic VTOL UAV assistance system demonstrator in the DLR Flying Robots lab with custom-built universal hinge (left) and screenshot from the simulation and visualization environment (right). Copyright © 2015 IEEE [213].

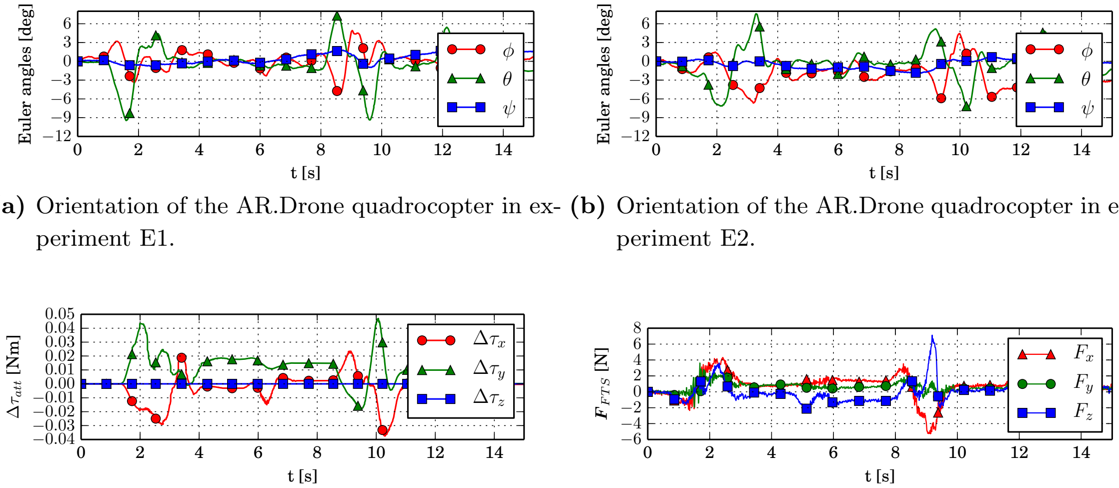
experiments are summarized in Table 4.2. For all experiments, the same desired position trajectory with respect to time t, shown in Figure 4.12a, is used and the desired orientation of the quadrocopter is set to $\boldsymbol{q}_d = (1 \quad 0 \quad 0 \quad 0)^T$. The gains of the backstepping controller found by trial-and-error are $\boldsymbol{K}_1 = \boldsymbol{K}_2 = \begin{bmatrix} \boldsymbol{E}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & 30\boldsymbol{E}_{3\times3} \end{bmatrix}$ and $k_3 = 35$. The implementation of the controller (4.86) includes the gravity compensation $\boldsymbol{g}(\boldsymbol{\phi})$. To compute $\boldsymbol{\tau}_{nsp}$, the elbow-field from [71] is used such that the elbow of the robot points downwards. The end-effector should always point upwards. The PD controller (4.23) with gains $k_p = 25$ and $k_d = 7$ generates the torque $\boldsymbol{\tau}_{ori}$ for the end-effector orientation.



(a) Desired trajectory $\boldsymbol{r}_d(t)$ for experiments E1-E3.

(b) Position errors $\boldsymbol{r}_e(t)$ in experiments E1-E3.

Figure 4.12: Desired position $\boldsymbol{r}_d(t)$ (left) and position errors $\boldsymbol{r}_e(t)$ in experiments E1, E2, and E3 (right). Copyright © 2015 IEEE [213].



(a) Orientation of the AR.Drone quadrocopter in experiment E1.

(b) Orientation of the AR.Drone quadrocopter in experiment E2.

(c) Torque $\Delta\boldsymbol{\tau}_{att}$ sent to the AR.Drone quadrocopter in experiment E2.

(d) FTS measurement used in E3 to increase the position tracking accuracy.

Figure 4.13: Orientation of the quadrocopter in E1 (top, left) and E2 (top, right), torque $\Delta\boldsymbol{\tau}_{att}$ sent to the AR.Drone in E2 (bottom, left), and force measurement $\boldsymbol{F}_{FTS}$ in E3 (bottom, right). Copyright © 2015 IEEE [213].

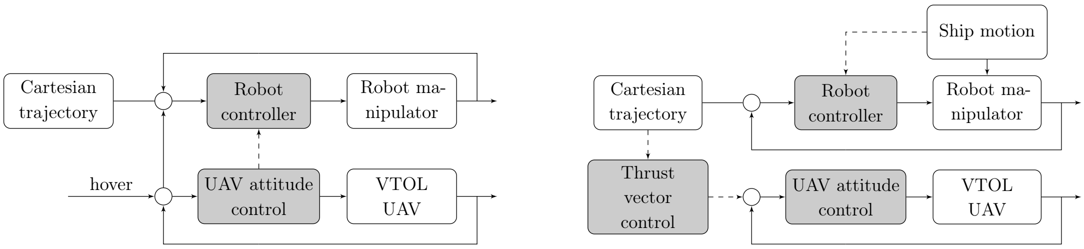
The initial experiment E1 sets the basis for comparison. Therefore, neither the additional torque $\Delta\boldsymbol{\tau}_{att}$ nor the measurement from the force-torque sensor $\boldsymbol{F}_{FTS}$ is used. The desired acceleration of the end-effector $\ddot{\boldsymbol{x}}_d$ is also set to zero. The results of E1 reveal overshoots in the end-effector position (see Figure 4.12b) and a deviation of up to 9.2 deg from the desired orientation, illustrated in Figure 4.13a using Euler angles $(\varphi \quad \theta \quad \psi)^T = \boldsymbol{0}$.

In the second experiment E2, the desired acceleration $\ddot{\boldsymbol{x}}_d$ is used in the controller (4.86) and the torque $\Delta\boldsymbol{\tau}_{att}$, depicted in Figure 4.13c, is sent to the UAV. Figure 4.12b shows that using the acceleration $\ddot{\boldsymbol{x}}_d$ leads to an increased accuracy in position tracking with less overshoot, while there is still a deviation from the desired trajectory. Figure 4.13b shows a decreased deviation from the desired orientation compared to E1, but the amplitude is still up to 6.8 deg. This is due to the fact that the torque of the AR.Drone is feed-forward controlled using the controller implemented by the manufacturer which does not yield accurate results. In the final experiment E3, the force $\boldsymbol{F}_{FTS,F_z}$ measured by the force-torque sensor (Figure 4.13d) is additionally fed back into (4.85). The results of E3, depicted in Figure 4.12b, indicate that the forces acting at the robotic end-effector in vertical direction are compensated and that therefore the controller's performance in position tracking in this direction is increased. The torque $\Delta\boldsymbol{\tau}_{att}$ in E3 is similar to E2 (see Figure 4.13c) and is therefore omitted here. Hence, it can be concluded that the backstepping controller (4.81) combined with the extended manipulator controller (4.86) enables compliant and coordinated physical interaction between the robot manipulator arm and the aerial vehicle with decent accuracy for the landing task.

4.4 Base motion compensation and active thrust control

So far the actual thrust vector direction of the flying robot is not considered and the motion of the robot manipulator's base is neglected (see Figure 4.14 (a)). Both are addressed in this section, as shown in Figure 4.14 (b), extending the linear and nonlinear control approaches already presented in this chapter.

Compensation of the base motion is required to realize accurate trajectory tracking. Active thrust vector control allows the flying robot to contribute to the assistance task and reduces the workload of the robot manipulator. Both extensions are evaluated in a simulation case



(a) Overview of control framework used in Section 4.2 [208] and Section 4.3 [213].

(b) Schematic overview of the active thrust vector approach presented in Section 4.4 [212].

Figure 4.14: Comparison of two different approaches for coordinated control of robot manipulator and flying robot. Copyright © 2017 IEEE [212].

study. In order to test the robustness of the control approach, the model parameters of five flying robots with different mass and inertia are used in the simulation.

4.4.1 Planar moving base system model

For the simulation case study, the planar model shown in Figure 4.15 of a three link robot manipulator on a moving base is used. A flying robot attached at the end-effector via a universal hinge (or ball joint) adds one more degree-of-freedom to the complete system, i.e. $n = 7$. The state coordinates of the planar model are defined as

$$\boldsymbol{s} = \begin{pmatrix} x_b & z_b & \beta & \phi_1 & \phi_2 & \phi_3 & \phi_u \end{pmatrix}^T = \begin{pmatrix} \boldsymbol{s}_b^T & \boldsymbol{s}_r^T & s_u \end{pmatrix}^T, \tag{4.89}$$

where $\boldsymbol{s}_b = (x_b \quad z_b \quad \beta)^T$, $\boldsymbol{s}_r = (\phi_1 \quad \phi_2 \quad \phi_3)^T$, and $s_u = \phi_u$. Therein, x_b, z_b are the translational and β is the rotational DoF of the base, ϕ_i, $i \in \{1, 2, 3\}$ are the joint angles of the robot, and ϕ_u is the relative orientation of the flying robot.

The planar system dynamics can be derived using the Projection Equation approach presented in Section 2.4.2. Model parameters of the considered robot manipulator are summarized in Table 4.3. Therein, m_i and I_i^c are the mass and the inertia w.r.t. the center of mass C of each link with length l_i, r_{pi} is the distance from the origin of frame p to the origin of frame i along the z-axis of p, d_i is the distance from the origin of p to c, and $\hat{\tau}_i$ are the maximum joint torques. These parameters are reasonable for the considered class of manipulators, e.g. for the KUKA/DLR Light Weight Robot.

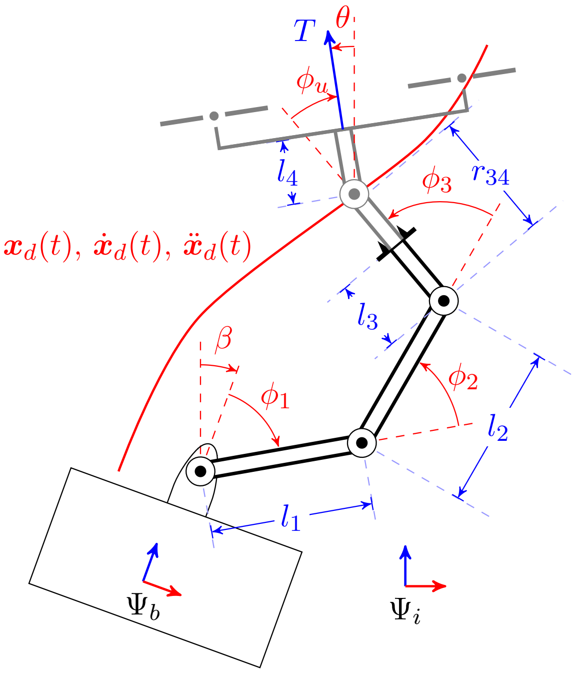


Figure 4.15: Planar three link manipulator with flying robot attached at the end-effector via an universal hinge (grayed out), which adds one more DoF to the mutibody system. Copyright © 2017 IEEE [212].

Table 4.3: Parameters of three link robot manipulator with attached flying robot.

(p,i)	l_i [m]	r_{pi} [m]	d_i [m]	m_i [kg]	I_i^c [kg·m^2]	$\hat{\tau}_i$ [Nm]
$(0,1)$	0.52	0	0.2	5.25	$1/12\, m_i l_i^2$	180.0
$(1,2)$	0.45	0.4	0.195	3.81	$1/12\, m_i l_i^2$	80.0
$(2,3)$	0.07	0.39	0.03	1.76	$1/2\, m_i l_i^2$	30.0
$(3,4)$	0.15	0.22	0.15	Table 4.4	Table 4.4	-

From the combined system dynamics (2.43) written in block form in Section 2.4.3, the couplings between base (index b), robot arm (index r), and flying robot (UAV, index u) become apparent. For a robot manipulator operating on a large platform, e.g. a ship, it is reasonable to neglect the influence of the manipulator on the ship (but not vice versa) [55]. Hence, the coupling terms in the base dynamics vanish, such that

$$\begin{bmatrix} \boldsymbol{M}_{bb} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{M}_{br}^T & \boldsymbol{M}_{rr} & \boldsymbol{M}_{ru} \\ \boldsymbol{M}_{bu}^T & \boldsymbol{M}_{ru}^T & \boldsymbol{M}_{uu} \end{bmatrix} \begin{pmatrix} \ddot{\boldsymbol{s}}_b \\ \ddot{\boldsymbol{s}}_r \\ \ddot{s}_u \end{pmatrix} + \begin{bmatrix} \boldsymbol{C}_{bb} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{C}_{rb} & \boldsymbol{C}_{rr} & \boldsymbol{C}_{ru} \\ \boldsymbol{C}_{ub} & \boldsymbol{C}_{ur} & \boldsymbol{C}_{uu} \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{s}}_b \\ \dot{\boldsymbol{s}}_r \\ \dot{s}_u \end{pmatrix} + \begin{pmatrix} \boldsymbol{g}_b \\ \boldsymbol{g}_r \\ \boldsymbol{g}_u \end{pmatrix} = \begin{pmatrix} \boldsymbol{\tau}_b \\ \boldsymbol{\tau}_r \\ \tau_u \end{pmatrix} + \boldsymbol{J}^T \begin{pmatrix} \boldsymbol{R}_{ui}^T \begin{pmatrix} 0 \\ T \end{pmatrix} \\ 0 \end{pmatrix}, \tag{4.90}$$

where $\boldsymbol{\tau}_r$ and τ_u are the torque inputs of robot arm and flying robot, respectively, T is the magnitude of the rotor thrust (see Figure 4.15), and $\boldsymbol{g} = \begin{pmatrix} \boldsymbol{g}_b^T & \boldsymbol{g}_r^T & \boldsymbol{g}_u^T \end{pmatrix}^T$ is the gravity vector. Note that the inputs of the base and external forces are not considered here. Instead, the motion of the base is assumed to be known. It enters the dynamics of the planar floating base system (4.90) via $\ddot{\boldsymbol{s}}_b$ and $\dot{\boldsymbol{s}}_b$.

4.4.2 Manipulator control with base motion compensation

The goal is to enable the robot manipulator to tow the aerial vehicle to the landing spot on the moving surface, as depicted in Figure 4.15. The end-effector pose of the robot should follow a given trajectory $\boldsymbol{x}_d(t) \in \mathbb{R}^3$ in the world frame with desired velocity $\dot{\boldsymbol{x}}_d(t)$ and acceleration $\ddot{\boldsymbol{x}}_d(t)$ while compensating for measured base motion. For consistency with Section 4.2 and Section 4.3, the influence of the UAV on the robot manipulator is neglected, hence only the manipulator dynamics are taken into account.

Inspired by [55], the remaining coupling terms in (4.90) can be deleted using the compensation torque

$$\boldsymbol{\tau}_{comp} = \boldsymbol{M}_{br}^T \ddot{\boldsymbol{s}}_b + \boldsymbol{C}_{rb} \dot{\boldsymbol{s}}_b. \tag{4.91}$$

Assuming that (4.91) is applied, the configuration dynamics of the manipulator become

$$\boldsymbol{M}_{rr} \ddot{\boldsymbol{s}}_r + \boldsymbol{C}_{rr} \dot{\boldsymbol{s}}_r + \boldsymbol{g}_r = \boldsymbol{\tau}_r. \tag{4.92}$$

With $\boldsymbol{J} = \begin{bmatrix}\boldsymbol{J}_b & \boldsymbol{J}_r\end{bmatrix}$, the velocities and accelerations in task space can be decomposed as

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{s}} = \underbrace{\boldsymbol{J}_b\dot{\boldsymbol{s}}_b}_{\dot{\boldsymbol{x}}_b} + \boldsymbol{J}_r\dot{\boldsymbol{s}}_r, \tag{4.93}$$

$$\ddot{\boldsymbol{x}} = \underbrace{\dot{\boldsymbol{J}}_b\dot{\boldsymbol{s}}_b + \boldsymbol{J}_b\ddot{\boldsymbol{s}}_b}_{\ddot{\boldsymbol{x}}_b} + \dot{\boldsymbol{J}}_r\dot{\boldsymbol{s}}_r + \boldsymbol{J}_r\ddot{\boldsymbol{s}}_r. \tag{4.94}$$

Rearranging (4.92) and inserting in the above yields

$$\ddot{\boldsymbol{x}} = \ddot{\boldsymbol{x}}_b + \dot{\boldsymbol{J}}_r\dot{\boldsymbol{s}}_r + \boldsymbol{J}_r\boldsymbol{M}_{rr}^{-1}\left(\boldsymbol{\tau}_r - \boldsymbol{C}_{rr}\dot{\boldsymbol{s}}_r - \boldsymbol{g}_r\right). \tag{4.95}$$

Similar to (2.54) - (2.57) in Section 2.4.4, the following substitutions

$$\boldsymbol{\tau}_r = \boldsymbol{J}_r^T\boldsymbol{F}_\tau, \tag{4.96}$$

$$\boldsymbol{g}_r = \boldsymbol{J}_r^T\boldsymbol{F}_{g,r}, \tag{4.97}$$

$$\boldsymbol{C}_{rr}\dot{\boldsymbol{s}}_r = \boldsymbol{J}_r^T\boldsymbol{\mu}_r + \boldsymbol{J}_r^T\boldsymbol{\Lambda}_r\dot{\boldsymbol{J}}_r\dot{\boldsymbol{s}}_r, \tag{4.98}$$

lead to

$$\boldsymbol{\Lambda}_r\ddot{\boldsymbol{x}} + \boldsymbol{\mu}_r + \boldsymbol{F}_{g,r} = \boldsymbol{\Lambda}_r\ddot{\boldsymbol{x}}_b + \boldsymbol{F}_\tau, \tag{4.99}$$

where $\boldsymbol{\Lambda}_r = (\boldsymbol{J}_r\boldsymbol{M}_{rr}^{-1}\boldsymbol{J}_r^T)^{-1}$ and $\ddot{\boldsymbol{x}}_b$ is the acceleration of the base. A possible choice for a task space controller with base motion compensation is

$$\boldsymbol{F}_\tau = \boldsymbol{\Lambda}_r\ddot{\boldsymbol{x}}_d + \boldsymbol{\mu}_{r,d} + \boldsymbol{K}\left(\boldsymbol{x}_d - \boldsymbol{\chi}(\boldsymbol{s})\right) + \boldsymbol{D}\left(\dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}\right) + \boldsymbol{F}_{g,r} - \boldsymbol{\Lambda}_r\ddot{\boldsymbol{x}}_b, \tag{4.100}$$

where $\boldsymbol{\chi}(\boldsymbol{s})$ is the forward kinematics of the complete system and $\boldsymbol{K} > \boldsymbol{0}$, $\boldsymbol{D} > \boldsymbol{0}$ are the desired stiffness and damping matrices, respectively. The stiffness and damping terms may be directly substituted with the control laws derived in Section 4.2 and Section 4.3. The computed-torque controller is finally implemented using (4.100), (4.91), and a nullspace torque $\boldsymbol{\tau}_{nsp}$ (2.51) as

$$\boldsymbol{\tau}_j = \boldsymbol{J}_r^T\boldsymbol{F}_\tau + \boldsymbol{\tau}_{comp} + \boldsymbol{\tau}_{nsp}. \tag{4.101}$$

It is directly applicable to redundant systems. However, since the planar 3 DoF manipulator depicted in Figure 4.15 is non-redundant w.r.t. the planar task, the control law reduces to

$$\begin{aligned}\boldsymbol{\tau}_j = {} & \underbrace{\boldsymbol{M}_{rr}\boldsymbol{J}_r^{-1}\ddot{\boldsymbol{x}}_d + (\boldsymbol{C}_{rr} - \boldsymbol{M}_{rr}\boldsymbol{J}_r^{-1}\dot{\boldsymbol{J}}_r)\boldsymbol{J}_r^{-1}\dot{\boldsymbol{x}}_d}_{\text{tracking terms}} + \underbrace{\boldsymbol{J}_r^T\left(\boldsymbol{K}\left(\boldsymbol{x}_d - \boldsymbol{\chi}(\boldsymbol{s})\right) + \boldsymbol{D}\left(\dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}\right)\right)}_{\text{PD terms}} \\ & + \underbrace{\boldsymbol{g}_r}_{\text{gravity compensation}} + \underbrace{\boldsymbol{M}_{br}^T\ddot{\boldsymbol{s}}_b + \boldsymbol{C}_{rb}\dot{\boldsymbol{s}}_b - \boldsymbol{M}_{rr}\boldsymbol{J}_r^{-1}\ddot{\boldsymbol{x}}_b - (\boldsymbol{C}_{rr} - \boldsymbol{M}_{rr}\boldsymbol{J}_r^{-1}\dot{\boldsymbol{J}}_r)\boldsymbol{J}_r^{-1}\dot{\boldsymbol{x}}_b}_{\text{base motion compensation}}\end{aligned} \tag{4.102}$$

This always applies if the number of DoF of manipulator, base, and task is equal. Note that neither inertia nor Coriolis/centrifugal matrices, but only the gravity compensation in (4.102) depend on the base orientation β. Most importantly, the control law does not require the inertia of the base, which is not available in general. Note that the coupling terms used in the base motion compensation also do not contain the base inertia. This can be directly seen from the inertia matrix (2.61) and the Coriolis/centrifugal matrix (2.62) of the representative two body model presented in Section 2.4.5.

The controller (4.102) does not take the coupling terms in (4.90) between flying robot and manipulator dynamics into account. The combined system model of robot manipulator and flying robot including the coupling terms is considered in Chapter 5. The advantage of the controller (4.102) is that it does not require UAV model or state information. However, for heavy flying robots with large thrust T and high desired accelerations, the robot can reach its maximum joint torque limits. This fact is addressed in Section 4.4.4.

4.4.3 Planar flying robot attitude control

As before, it is assumed that the vertical take-off and landing unmanned aerial vehicle is equipped with a PD attitude controller

$$\tau_u = k_p(\theta_d - \theta) + k_d(\dot{\theta}_d - \dot{\theta}) = -k_p\theta_e - k_d\dot{\theta}_e, \tag{4.103}$$

with $\theta = \beta + \sum_{i=1}^{4}\phi_i$, desired orientation θ_d and angular velocity $\dot{\theta}_d$. As stated above in Section 4.2.2, the PD controller has to satisfy condition (4.18)

$$k_p > m_u \cdot g \cdot l_4, \quad k_d > 0, \tag{4.104}$$

in order to ensure stability while the flying robot is attached to the robot via a universal hinge as shown in Figure 4.15. This conservative condition is derived from the linearized system dynamics. Robustness is increased by increasing k_p. However, an alternative formulation can be found using a Lyapunov analysis of the multibody system components that influence the flying robot. Extracting and evaluating those terms from (4.90)

$$\begin{bmatrix} \boldsymbol{M}_{bu}^T & \boldsymbol{M}_{ru}^T & \boldsymbol{M}_{uu} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{s}}_b \\ \ddot{\boldsymbol{s}}_r \\ \ddot{s}_u \end{bmatrix} - \boldsymbol{Q}_{uu} = 0 \tag{4.105}$$

for the planar model yields

$$I_u(\ddot{\beta} + \sum_{i=1}^{4}\ddot{\phi}_i) - m_u \begin{pmatrix} l_4 \\ 0 \end{pmatrix} \boldsymbol{R}_{ui} \begin{pmatrix} 0 \\ g \end{pmatrix} - \tau_u = 0 \tag{4.106}$$

and, hence,

$$\ddot{\theta} = \frac{-m_u \cdot g \cdot l_4 \cdot \sin(\theta) + \tau_u}{I_u}. \tag{4.107}$$

Note that $m_u = m_4$ and $I_u = I_4^c$, since $n = 4$ and the flying robot is the last element in the chain. The Coriolis and centrifugal terms in $\begin{bmatrix} \boldsymbol{C}_{ub} & \boldsymbol{C}_{ur} & \boldsymbol{C}_{uu} \end{bmatrix}$ are neglected, which is reasonable for a straight-line trajectory. Hence, the controller (4.103) is augmented using the above results as

$$\tau_u = -k_p\theta_e - k_d\dot{\theta}_e + m_u \cdot g \cdot l_4 \cdot \sin(\theta). \tag{4.108}$$

In order to prove stability, the Lyapunov function candidate

$$V = \frac{1}{2}\frac{k_p}{I_u}\theta_e^2 + \frac{1}{2}\dot{\theta}_e^2$$

Table 4.4: Flying robot model parameters.

Name	Type	m_u [kg]	I_u [kg m^2]	$T_{\max}$ [N]	$\frac{T_{\max}}{\mathrm{W}}$ [-]
Parrot AR.Drone 2.0	Quad	0.480	0.006	10.0	2.12
AscTec Hummingbird	Quad	0.710	0.003	20.0	2.87
AscTec Firefly	Hexa	1.00	0.010	36.0	3.67
MIT X-Cell	Heli	4.54	0.18	162.0	3.64
Yamaha R-50	Heli	43.94	5.98	754.0	1.75

is used, for which it now follows intuitively with (4.107), (4.108), and $\ddot{\theta}_d = 0$, that

$$\dot{V} = \frac{k_p}{I_u}\theta_e\dot{\theta}_e + \dot{\theta}_e\ddot{\theta}_e = \frac{k_p}{I_u}\theta_e\dot{\theta}_e + \dot{\theta}_e\left(-\frac{k_p}{I_u}\theta_e - \frac{k_d}{I_u}\dot{\theta}_e\right) = -\frac{k_p}{I_u}\dot{\theta}_e^2 < 0, \tag{4.109}$$

i.e. the extended controller (4.108) renders the equilibrium of (4.107) asymptotically stable. Note that the linear stability criterion (4.104) also applies to (4.107) directly, since $|\sin(\theta)| \leq 1$.

4.4.4 Flying robot thrust vector control

In general, the thrust force of a VTOL UAV points along the body-fixed vertical axis. For multicopters and helicopters thrust and torque are not independent, i.e. zero thrust would imply zero admissible torque and therefore zero attitude control authority. As can be seen from Figure 4.15 and Eq. (4.90), the thrust does not have a moment about the pivot of the rotational hinge but can counteract the gravity force. For the robot manipulator, the thrust is an external force acting at the end-effector.

Hence, the objective is to find a reasonable value for the thrust, which is nonzero and does not disturb the robot manipulator in the assistance task. The solution is to demand the thrust vector to counteract gravity and to contain the desired direction of motion of robot manipulator and flying robot, i.e. $\boldsymbol{f}_i = \boldsymbol{g}_u + m_u\ddot{\boldsymbol{x}}_d$. Therein, the known acceleration $\ddot{\boldsymbol{x}}_d$ of the desired trajectory is used for feed-forward control. As presented in Section 3.3.3, the thrust magnitude can be obtained by projecting the force in the inertial frame on the vertical axis of the flying robot via

$$T = \boldsymbol{z}_u^T\boldsymbol{f}_i. \tag{4.110}$$

Finally, the desired thrust vector orientation is derived from (4.110) for the planar case as

$$\theta_d = \operatorname{asin}\left(\frac{\boldsymbol{f}_i^T\boldsymbol{e}_1}{||\boldsymbol{f}_i||}\right). \tag{4.111}$$

The general solution in 3D can be found in Appendix A4. As a result, the flying robot will counteract its weight and decrease the forces applied by the robot (and therefore the required joint torques) in order to accelerate along the trajectory. Note that for the practical application, a well identified rotor model, as well as rotorspeed and airspeed measurements are mandatory, because the thrust of a rotor depends not only on its rotation speed but also on the airspeed. A simulative validation of the presented control approach is provided in the next section.

4.4.5 Simulation case study with ship motion

In order to evaluate the performance of the control approach (see Figure 4.14(b)), a simulation case study using MATLAB/Simulink® is conducted. To consider thrust vector control and base motion compensation independently, the actual case study is carried out in two steps: under no base motion and with ship motion at three different sea states. In each part, the following scenarios are treated:

i.) a worst case scenario, i.e. maximum thrust is applied,

ii.) a flying robot under pure gravity compensation, and

iii.) active thrust vector control.

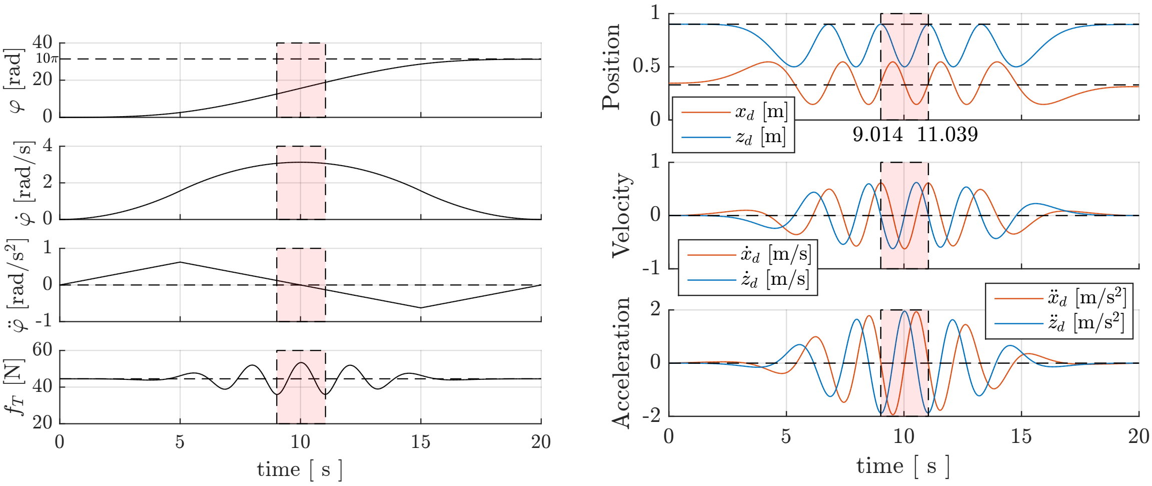
For the study, the three link robot model parameters shown in Table 4.3 are used. To assess the robustness of the presented control approach, the five different VTOL UAVs shown in Figure 4.16 are considered. In Table 4.4, m_u, I_u, $T_{\max}$, and $\frac{T_{\max}}{W}$ are mass, inertia, maximum thrust, and thrust-to-weight ratio of the flying robots, respectively. The multicopter parameters are those provided by the manufacturers and the helicopter parameters are taken from [176]. In order to generate results for an admissible range of desired velocities and accelerations, the circular reference trajectory depicted in Figure 4.17 and Figure 4.18a is used.

Note that in i.) and ii.), it is assumed that the robot controller knows about the mass m_u, inertia I_u, and applied thrust T and that the flying robot maintains its hover attitude, which are similar assumptions as made in the previous control approach shown in Figure 4.14(a). In iii.), the flying robot tracks a desired orientation computed by (4.111) and controls its thrust using (4.110). In i.) - iii.), the robot controller gains are set to $\boldsymbol{K} = 1000\boldsymbol{E}$ and $\boldsymbol{D} = 30\boldsymbol{E}$. The attitude controller gains $k_p = 1800 \cdot g \cdot l_4$ and $k_d = 0.6$ are found to hold well for the five considered VTOL UAVs.

In Figure 4.17, active thrust vector control for a X-Cell helicopter with mass $m_u = 4.54\,\mathrm{kg}$ (see Table 4.4 for further details) is visualized using the exemplary circular trajectory defined in polar coordinates with $R = 0.2\,\mathrm{m}$ and a bang-bang jerk profile. It results in five revolutions within 20 s with varying velocity and acceleration in Cartesian space. The resulting orientation θ of the flying robot during the area of maximum acceleration (highlighted red in Figure 4.17(a) and (b)) is shown using a colored contour of a multicopter in Figure 4.17(c). The thrust vector magnitude is depicted in the bottom plot of Figure 4.17(a). Note that

(a) AR.Drone **(b)** Hummingbird **(c)** Firefly **(d)** X-Cell **(e)** R-50

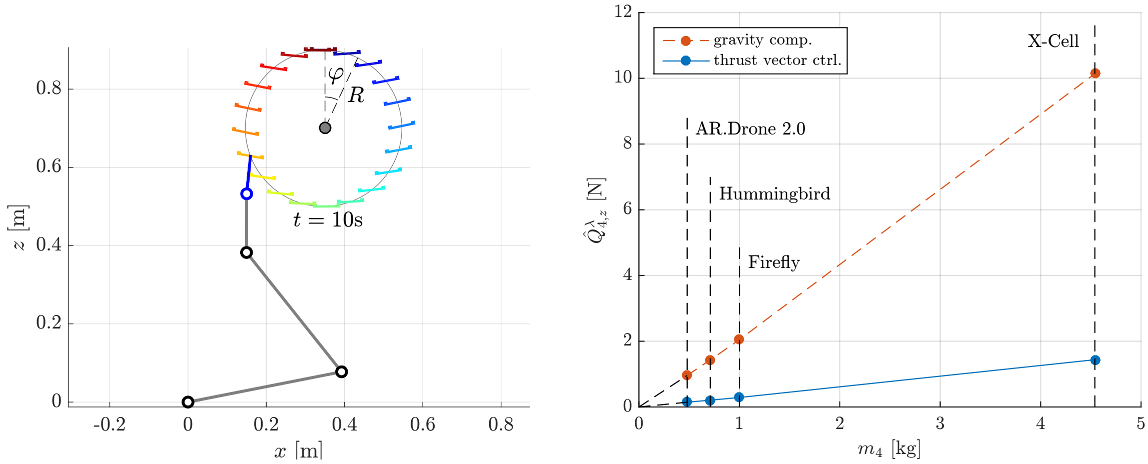
Figure 4.16: VTOL UAVs considered in the simulation case study.



(a) Circular trajectory defined in polar coordinates (φ, R) with bang-bang jerk profile.

(b) Circular trajectory in Cartesian space with area of maximum acceleration highlighted red.

Figure 4.17: Example of active thrust vector control on a circular trajectory. The thrust vector magnitude T w.r.t. time is shown in the bottom plot of Figure 4.17(a). Copyright © 2017 IEEE [212].



(a) Simulated circular trajectory with 0.2 m radius. The flying robot's orientation is illustrated using a colored contour.

(b) Comparison of constraint force $Q^{\lambda}_{4,z}$ w.r.t. the mass $m_4 = m_u$ of the flying robots in the simulation. The R-50 helicopter is omitted to increase readability.

Figure 4.18: Simulated trajectory (left) and resulting constraint force between manipulator and UAV (right). Copyright © 2017 IEEE [212].

according to the desired acceleration in Cartesian space, the flying robot needs to decelerate on the right and accelerate on the left side of the circle. Both is achieved successfully by tilting the thrust vector and decreasing, respectively increasing, the thrust force. Maximum thrust is produced at about $t = 10$ s. At that instant, the desired acceleration in x-direction is zero while maximum acceleration in z-direction is demanded. Therefore, the thrust vector of the flying robot points perfectly upright and produces maximum thrust. In addition to the tracking errors in position, velocity, and acceleration, defined as

$$e(\boldsymbol{x}) = \sqrt{(x_d - x)^2 + (z_d - z)^2}, \tag{4.112}$$

and of the errors in orientation θ_e of the flying robot, defined as in (4.103), the maximum constraint force $\boldsymbol{Q}_4^{\lambda}$ is examined in detail. It gives insight in the force applied by the robot manipulator in order to accelerate the flying robot along the desired trajectory.

No base motion: Maximum thrust vs. gravity compensation vs. active thrust control

For this simulation set, no base motion is assumed and the simulation is conducted as described above. The results for the cases i.) - iii.) are summarized in Table 4.5 for the five VTOL UAVs listed in Table 4.3. It can be seen, that in i.) and ii.) the position tracking errors are similar, which is due to the assumption of known flying robot properties. However, the maximum joint torques of the robot in i.) and ii.) are distributed differently. In iii.), compared to i.) and ii.), the performance in trajectory tracking is significantly increased. The flying robot is stabilizing the hover orientation in i.) and ii.) but has to follow an attitude trajectory in iii.), which results in an increased attitude error. For the R-50 helicopter, the torque limits of the manipulator are exceeded in i.) and ii.) (highlighted red in Table 4.5), whereas in iii.) the manipulator stays within its actuator constraints. The robot joint torques in iii.) tend to be below those in ii.), while the torque τ_u of the flying robot and its thrust T in iii.) are above those in ii.). This shows that under active thrust vector control, more workload is put on the aerial vehicle. Note that the VTOL UAV's thrust magnitude T complies with its limit $T_{\max}$, listed in Table 4.4, in all three cases and for all five UAVs.

Table 4.5: Simulation results without base motion.

	AR.Drone 2.0			Hummingbird			Firefly			X-Cell			R-50		
	i.)	ii.)	iii.)	i.)	ii.)	iii.)	i.)	ii.)	iii.)	i.)	ii.)	iii.)	i.)	ii.)	iii.)
$e(\boldsymbol{x})$ [cm] max	0.67	0.67	0.59	0.72	0.72	0.59	0.78	0.78	0.60	1.53	1.53	0.63	11.49	11.49	4.59
$e(\dot{\boldsymbol{x}})$ $[\frac{\text{cm}}{\text{s}}]$max	5.50	5.50	4.75	5.91	5.91	4.78	6.43	6.43	4.85	11.45	11.45	4.83	51.78	51.78	14.87
$e(\ddot{\boldsymbol{x}})$ $[\frac{\text{cm}}{\text{s}^2}]$max	58.9	58.9	51.0	62.5	62.5	50.9	67.0	67.0	50.9	107.0	107.0	44.3	206.8	206.8	77.4
$\lvert\theta_e\rvert$ [deg] max	0.01	0.01	0.48	0.02	0.02	1.31	0.02	0.02	0.59	0.12	0.12	0.10	2.29	2.29	2.29
$\lvert\tau_1\rvert$ [Nm] max	38.9	41.7	40.9	35.2	42.0	41.0	28.9	42.5	40.9	26.8	49.1	40.9	168.2	158.4	69.3
$\lvert\tau_2\rvert$ [Nm] max	10.1	11.4	10.8	8.6	11.7	10.9	5.9	12.1	10.8	19.2	15.7	10.6	76.6	121.1	36.5
$\lvert\tau_3\rvert$ [Nm] max	0.4	0.4	0.1	0.5	0.5	0.1	0.7	0.7	0.1	3.3	3.3	0.8	63.1	62.9	29.7
$\lvert\tau_u\rvert$ [Nm] max	2e-4	2e-4	0.50	3e-4	3e-4	0.53	8e-4	8e-4	0.56	0.03	0.03	1.77	7.90	7.90	32.61
T [N] max	10.0	4.7	5.6	20.0	7.0	8.4	36.0	9.8	11.8	162.0	44.5	53.4	754.0	431.4	516.9

In Figure 4.18b, the maximum constraint force $\hat{Q}_{4,z}^{\lambda}$ in z-direction acting in the universal hinge is depicted. For cases ii.) and iii.), a linear relation for the constraint forces w.r.t. the flying robot mass m_u is observed. The slope for case ii.) is identical with the maximum acceleration of the trajectory in z-direction, whereas the slope for iii.) is much smaller. It holds for the considered Cartesian trajectory independently from the considered robot manipulator parameters. For the given trajectory, the thrust is always helping in i.). This will not be the case for other trajectories, like a straight line. The plot of the constraint force in Figure 4.18b substantiates the low torques in ii.) and iii.), but the actual torques depend on the current configuration of the robot manipulator. However, it can be concluded, that active thrust vector control indeed increases the trajectory tracking performance while extending the range of flying robots a robot manipulator can cope with in the take-off or touch-down phase.

Without vs. with base motion compensation at three different sea states

The second part of the simulation case study is devoted to the validation of base motion compensation. In order to assess the practical potential of robot-assisted landing of VTOL UAVs on ships, a realistic ship motion simulation is needed. In-depth treatment of ocean vehicle dynamics is beyond the scope of this work. A detailed introduction can be found in [92] from which the marine systems simulator (MSS) MATLAB® toolbox is utilized. The MSS provides the simulation of a supply vessel with length 82.80 m for which the hydrodynamic coefficients are computed using the commercial program ShipX. Wave excitation at the three different sea states 3, 4, and 5 is considered. Probability, maximum observed wave height, and peak frequency for all sea states are shown in Table 4.6. Note that the probability of sea states 0, 1, and 2 is summarized. All values in Table 4.6 are from [92].

Table 4.6: Sea states.

Sea state	Maximum wave height [m]	Peak frequency [rad/s]	World wide probability [%]
0	0	0	11.2486
1	0.1	1.11	
2	0.5	0.93	
3	1.25	0.79	31.6851
4	2.5	0.68	40.1944
5	4.0	0.60	12.8005
6	6.0	0.53	3.0253
7	9.0	0.46	0.9263
8	14.0	0.39	0.1190
9	Over 14.0	-	0.0009

Table 4.7: Resulting maximum base motion (supply vessel, L=82.80 m).

Sea state	$z_{b,\max}$ [m]	$\dot{x}_{b,\max}$ [m/s]	$\dot{z}_{b,\max}$ [m/s]	$\ddot{x}_{b,\max}$ [m/s^2]	$\ddot{z}_{b,\max}$ [m/s^2]	$\dot{\beta}_{\max}$ [rad/s]	$\ddot{\beta}_{\max}$ [rad/s^2]
3	0.11	0.13	0.09	0.08	0.06	0.01	0.01
4	0.39	0.33	0.27	0.20	0.22	0.02	0.02
5	0.90	0.55	0.59	0.28	0.42	0.03	0.03

The JONSWAP wave spectrum ([92], p. 206) is a de facto standard for nonfully developed seas with wind generated waves under the assumption of finite water depth and, therefore, used in this simulation case study. The simulation results for sea states 3 to 5 can be found in Table 4.8, wherein results without base motion compensation are highlighted gray and results with base motion compensation are highlighted blue. For conciseness, only the maximum joint torques of the robot manipulator are included in Table 4.8. The joint torques tend to increase with increasing sea state. With active base motion compensation, more workload is put on the robot, which results in higher joint torques compared to no base motion compensation. Regarding the R-50 helicopter, the manipulator considered in this work can only comply with its actuator limits at sea state 3 and if active thrust vector control iii.)

Table 4.8: Results with simulated ship motion of a supply vessel (L=82.80 m).

	AR.Drone 2.0			Hummingbird			Firefly			X-Cell			R-50		
	i.)	ii.)	iii.)	i.)	ii.)	iii.)	i.)	ii.)	iii.)	i.)	ii.)	iii.)	i.)	ii.)	iii.)
SEA STATE 3 (max. heave $\hat{z}_0 = 0.11$m)															
$\lvert\tau_1\rvert$ [Nm] max.	38.8	41.6	40.8	35.1	41.9	40.8	28.7	42.3	40.7	27.7	48.6	40.5	160.8	161.4	63.3
$\lvert\tau_2\rvert$ [Nm] max.	10.3	11.5	10.9	8.8	11.9	11.0	6.1	12.3	11.0	18.6	16.2	11.1	71.2	120.2	37.9
$\lvert\tau_3\rvert$ [Nm] max.	0.3	0.3	0.1	0.5	0.5	0.1	0.7	0.7	0.2	3.1	3.1	0.8	62.4	62.2	27.6
$\lvert\tau_1\rvert$ [Nm] max.	38.7	41.5	40.7	35.0	41.8	40.7	28.7	42.2	40.7	27.7	48.5	40.4	160.9	161.9	65.5
$\lvert\tau_2\rvert$ [Nm] max.	10.3	11.6	10.9	8.8	11.9	11.1	6.1	12.3	11.1	18.5	16.3	11.2	71.4	120.5	40.3
$\lvert\tau_3\rvert$ [Nm] max.	0.4	0.4	0.1	0.5	0.5	0.1	0.7	0.7	0.2	3.1	3.1	0.8	62.6	62.4	29.4
SEA STATE 4 (max. heave $\hat{z}_0 = 0.39$m)															
$\lvert\tau_1\rvert$ [Nm] max.	38.7	41.5	40.8	34.9	41.8	40.9	28.6	42.2	41.0	29.7	48.9	42.4	157.2	173.9	77.5
$\lvert\tau_2\rvert$ [Nm] max.	10.2	11.5	10.9	8.7	11.8	11.0	6.0	12.3	11.0	21.0	16.2	11.8	91.3	127.8	56.8
$\lvert\tau_3\rvert$ [Nm] max.	0.4	0.4	0.1	0.6	0.6	0.2	0.8	0.8	0.3	3.5	3.5	1.4	60.9	60.7	35.1
$\lvert\tau_1\rvert$ [Nm] max.	38.6	41.4	40.8	34.9	41.7	40.9	28.6	42.2	41.0	29.7	48.7	42.4	160.1	175.5	80.6
$\lvert\tau_2\rvert$ [Nm] max.	10.2	11.5	10.8	8.7	11.8	10.9	6.0	12.2	10.9	21.1	16.2	11.9	93.1	127.8	57.0
$\lvert\tau_3\rvert$ [Nm] max.	0.4	0.4	0.1	0.6	0.6	0.2	0.8	0.8	0.3	3.5	3.5	1.4	59.5	59.3	38.3
SEA STATE 5 (max. heave $\hat{z}_0 = 0.90$m)															
$\lvert\tau_1\rvert$ [Nm] max.	39.5	42.3	41.6	35.8	42.7	41.7	29.4	43.3	41.8	32.3	51.4	43.7	172.1	181.7	101.5
$\lvert\tau_2\rvert$ [Nm] max.	9.9	11.2	10.8	8.3	11.5	10.9	5.7	11.9	10.9	19.0	16.6	12.1	81.3	135.3	57.7
$\lvert\tau_3\rvert$ [Nm] max.	0.5	0.5	0.2	0.7	0.7	0.3	1.0	1.0	0.4	4.3	4.3	2.0	62.9	62.7	38.0
$\lvert\tau_1\rvert$ [Nm] max.	39.1	42.0	41.4	35.5	42.4	41.6	29.3	42.9	41.7	32.2	50.7	43.7	171.5	186.5	112.3
$\lvert\tau_2\rvert$ [Nm] max.	9.9	11.1	10.7	8.3	11.4	10.8	5.7	11.8	10.8	19.0	16.7	11.9	85.4	135.2	80.0
$\lvert\tau_3\rvert$ [Nm] max.	0.5	0.5	0.2	0.7	0.7	0.3	1.0	1.0	0.4	4.4	4.4	2.1	63.7	63.5	50.0

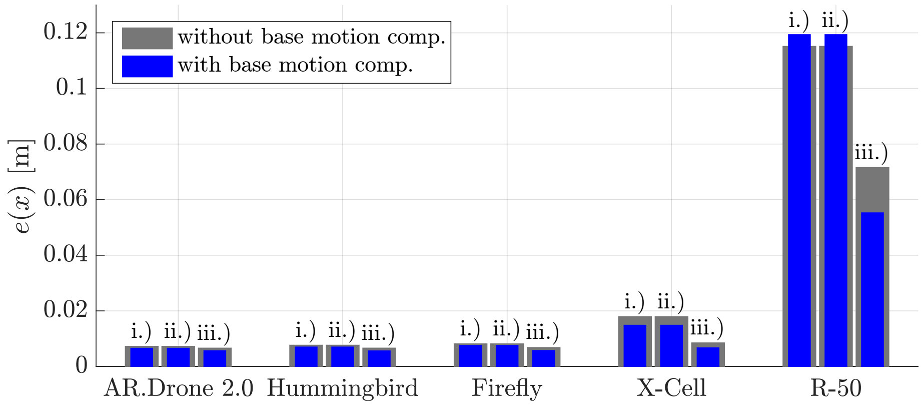


Figure 4.19: Position tracking error (see Eq. (4.112)) with ship motion at sea state 4. Copyright © 2017 IEEE [212].

is used. Analyzing the position tracking error at sea state 4, depicted in Figure 4.19, reveals that base motion compensation increases the tracking accuracy considerably, except for the R-50 helicopter in i.) and ii.), since the joint torque limits are exceeded.

4.5 Summary and conclusion

In this chapter, the separate models of flying robot and robot manipulator are considered and linear and nonlinear controllers are designed. The approaches are applicable especially for light weight flying robots, where the reaction forces on the robot manipulator are negligible or may be treated as external disturbances. At first, base motion is neglected and the flying robot's thrust is assumed constant. Subsequently, the approaches are extended to include base motion compensation and active control of the flying robot's thrust vector. The control approaches are evaluated in experiments with a robotic assistance system demonstrator composed of a DLR/KUKA Light Weight Robot and an off-the-shelf quadrocopter. Both base motion compensation and thrust vector control are evaluated in an extensive simulation case study. It reveals that thrust vector control is required for flying robots above 5 kg take-off weight in order to comply with the joint torque limits of the manipulator.

The advantage of the control approaches based on separate models mostly lies in their manageable complexity. Since they require little state information, communication between the two subsystems is reduced to a minimum. This in turn reduces the required hardware components and the implementation effort on middleware level. In addition, the impedance control strategies employed on both manipulator and flying robot side provide passive and, therefore, stable interaction between the subsystems. However, the presented approaches neglect the coupling terms present in the combined system dynamics. The latter as well as heavier flying robots are explicitly addressed in the following chapter.

"Coming together is a beginning, staying together is progress, and working together is success."
— Henry Ford

5 Controller designs using a combined model

In contrast to the previous Chapter 4, where separate models of the flying robot and the robot manipulator are considered, this chapter is devoted to the control of the combined system composed of a flying robot, a manipulator, and a base. Unlike in whole-body control [177], the base is not actuated, but its states are assumed to be measurable.

The most important feature is that both robot manipulator and flying robot are able to apply a wrench in the inertial frame by means of their actuators, i.e. joint motors and rotors, respectively. Hence, the system is overactuated w.r.t. the assistance task. Due to the coupling of forces and torques of common underactuated flying robots (i.e. with parallel rotor thrust vectors), a thrust force is always present, as long as torques are required to stabilize the attitude. Hence, in this section, solutions are presented which distribute the required force in task space between the robot manipulator and the flying robot. The assistance task then becomes a cooperative task, since both robot manipulator and flying robot contribute to it using their actuators. This is especially applicable to heavy UAVs for which otherwise the maximum payload of the robot manipulator would be exceeded.

The remainder of this chapter is structured as follows. First, the combined system model introduced in Section 2.4.3 is recapped in Section 5.1 and a task space controller for the combined system is presented in Section 5.2. Two possible options to incorporate the wrench of the flying robot in task space are highlighted. The first option is addressed in Section 5.3. Note that substantial parts of Section 5.1, Section 5.2, and Section 5.3 are taken from the author's paper [214]. In Section 5.3.1, a model decomposition based on the inertia of the subsystems is presented and transformed to task space. It is utilized to derive a heuristic for sharing the workload between the robot manipulator and the flying robot in Section 5.3.2. After additional remarks in Section 5.3.3 and Section 5.3.4 regarding the practical implementation, the workload sharing approach is evaluated in simulations and experiments in Section 5.3.5 and Section 5.3.6, respectively.

The second option to distribute the control effort between the flying robot and the manipulator is addressed in Section 5.4, where the control allocation is stated as an optimization problem. An analytical solution without inequality constraints and the quadratic problem, which can be solved numerically including linear equality and inequality constraints, are presented in Section 5.4.1 and Section 5.4.2, respectively. Furthermore, the optimal control allocation problem is extended to enable optimal reorientation of the flying robot in Section 5.4.3 and an adaptive parameter is added to the task space control law in order to account for uncertainties in the flying robot's thrust vector (Section 5.4.4). In Section 5.4.5, the adaptive task space controller for the combined system, as well as the optimal control allocation approach with optimal reorientation of the flying robot are evaluated in simulations including random but realistic base motions. To enable a practical implementation, Section 5.4.6 summarizes appropriate extensions of the combined control approach. Finally, Chapter 5 is summarized and concluded in Section 5.5.

5.1 Combined dynamics of flying robot and robot manipulator

Consider a generic robot manipulator r with n joints, as shown in Figure 5.1, with a flying robot (or UAV) u attached to the end-effector via a universal hinge with three rotational DoF (cf. Section 2.4.3). The latter can be modeled as an additional joint in the kinematic chain, i.e. the combined system with $n+3$ DoF in configuration space is obtained

$$\underbrace{\begin{bmatrix} \boldsymbol{M}_{rr} & \boldsymbol{M}_{ru} \\ \boldsymbol{M}_{ru}^T & \boldsymbol{M}_{uu} \end{bmatrix}}_{\boldsymbol{M}} \underbrace{\begin{pmatrix} \ddot{\boldsymbol{\phi}}_r \\ \ddot{\boldsymbol{\phi}}_u \end{pmatrix}}_{\ddot{\boldsymbol{\phi}}} + \underbrace{\begin{bmatrix} \boldsymbol{C}_{rr} & \boldsymbol{C}_{ru} \\ \boldsymbol{C}_{ur} & \boldsymbol{C}_{uu} \end{bmatrix}}_{\boldsymbol{C}} \underbrace{\begin{pmatrix} \dot{\boldsymbol{\phi}}_r \\ \dot{\boldsymbol{\phi}}_u \end{pmatrix}}_{\dot{\boldsymbol{\phi}}} + \boldsymbol{g} = \boldsymbol{\tau}. \tag{5.1}$$

Therein, $\boldsymbol{M}$ is the inertia matrix, $\boldsymbol{C}$ is the Coriolis matrix, $\boldsymbol{g}$ is the gravity vector, and $\boldsymbol{\phi}$ is the vector of generalized coordinates [37]. The input vector

$$\boldsymbol{\tau} = \underbrace{\begin{pmatrix} \boldsymbol{\tau}_r \\ \boldsymbol{\tau}_u \end{pmatrix}}_{\boldsymbol{\tau}_c} + \boldsymbol{J}^T \begin{pmatrix} \boldsymbol{F}_u \\ \boldsymbol{0} \end{pmatrix} + \boldsymbol{\tau}_{ext} \tag{5.2}$$

comprises the actuator torques $\boldsymbol{\tau}_c$ of both robot manipulator ($\boldsymbol{\tau}_r$) and UAV ($\boldsymbol{\tau}_u$), the force vector

$$\boldsymbol{F}_u = \boldsymbol{R}_{iu} \begin{pmatrix} \boldsymbol{0} \\ T \end{pmatrix} \tag{5.3}$$

of the UAV in the inertial frame Ψ_i, as well as external disturbances $\boldsymbol{\tau}_{ext}$. The thrust T always points along the z-axis of the UAV's body-fixed frame Ψ_u. Furthermore, $\boldsymbol{R}_{iu}$ is a rotation matrix, which transforms a vector given in the Ψ_u-frame to the Ψ_i-frame. The matrix $\boldsymbol{J} = \frac{\partial \boldsymbol{\chi}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}$ is the Jacobian, with $\boldsymbol{\chi}(\boldsymbol{\phi})$ being the forward kinematics of the combined system. Note that here the end-effector is the UAV, as shown in Figure 5.1.

The pose of the flying robot in the inertial frame is defined as $\boldsymbol{x} = (\boldsymbol{\xi}^T \quad \boldsymbol{\Phi}^T)^T$, where $\boldsymbol{\xi}$ is the position of the reference point (depicted red in Figure 5.1). The shape of the orientation vector $\boldsymbol{\Phi}$ depends on the considered attitude representation (see Section 2.4.1). Equation (5.2) reveals that the combined system (5.1) is overactuated for $\boldsymbol{F}_u \neq \boldsymbol{0}$ (and $\text{rank}(\boldsymbol{J}) > 0$), while the flying robot itself is underactuated. This fact is exploited for workload sharing in Section 5.3.

Since take-off and landing maneuvers are defined in task space [38], the dynamics (5.1) are transformed, as done in Section 2.4.4, to obtain

$$\boldsymbol{\Lambda}\ddot{\boldsymbol{x}} + \boldsymbol{\mu} + \boldsymbol{F}_g = \boldsymbol{J}^{\#T}\boldsymbol{\tau}_c + \begin{pmatrix} \boldsymbol{F}_u \\ \boldsymbol{0} \end{pmatrix} + \boldsymbol{F}_{ext} = \boldsymbol{F}_\tau + \boldsymbol{F}_{ext}, \tag{5.4}$$

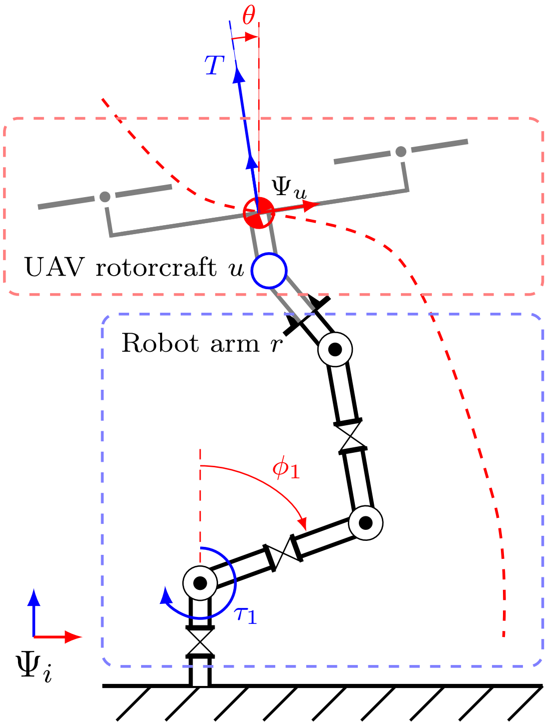


Figure 5.1: Visualization of the robot-assisted landing task and the considered system model consisting of a UAV rotorcraft and a robot manipulator. The goal is to tow the flying aerial vehicle to the ground by means of the robot manipulator mounted on the landing surface. The flying robot is attached to the robot manipulator via a universal hinge (or ball joint, blue) while the end-effector (red reference point) should follow a given trajectory. Copyright © 2017 IEEE [214].

with the generalized pseudoinverse (cf. Appendix A3)

$$\boldsymbol{J}^{\#T} = \left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\right)^{-1}\boldsymbol{J}\boldsymbol{M}^{-1}. \tag{5.5}$$

5.2 Task space control of the combined system

Utilizing the combined system model in task space (5.4), a straightforward choice for a controller that provides accurate and robust tracking is given by

$$\boldsymbol{F}_\tau = \boldsymbol{\Lambda}(\ddot{\boldsymbol{x}}_d - \boldsymbol{D}\dot{\boldsymbol{x}}_e - \boldsymbol{K}\boldsymbol{x}_e) + \boldsymbol{\mu}_d + \boldsymbol{F}_g, \tag{5.6}$$

which is an inverse dynamics controller (cf. (2.85) in Section 2.6), but an impedance controller (2.87) (with or without inertia shaping (2.95)) could be used as well. Two different options for controlling the combined system are considered in this chapter. They are summarized in Table 5.1. In the first case, the control torque is defined as the projection of the task space force applied by the robot manipulator minus the task space force already applied by the flying robot (i.e. its thrust force). This leads to

$$\boldsymbol{\tau}_c = \boldsymbol{J}^T \left\{ \boldsymbol{\Lambda}(\ddot{\boldsymbol{x}}_d - \boldsymbol{D}\dot{\boldsymbol{x}}_e - \boldsymbol{K}\boldsymbol{x}_e) + \boldsymbol{\mu}_d + \boldsymbol{F}_g - \begin{pmatrix} \boldsymbol{F}_u \\ \boldsymbol{0} \end{pmatrix} \right\} + \boldsymbol{\tau}_{nsp}, \tag{5.7}$$

Table 5.1: Considered options to control the combined system.

Case 1	Case 2
$\boldsymbol{\tau}_c = \boldsymbol{J}^T(\boldsymbol{F}_\tau - \boldsymbol{F}_u) + \boldsymbol{\tau}_{nsp}$	$\boldsymbol{Q}\boldsymbol{u} = \boldsymbol{J}^T\boldsymbol{F}_\tau + \boldsymbol{\tau}_{nsp}$
Allows to incorporate given $\boldsymbol{F}_u$, which may be designed such that the workload is shared as desired (refer to Section 5.3).	Allows to include the control allocation of the flying robot (see Section 3.4) and online optimization (refer to Section 5.4).

where $\boldsymbol{\tau}_{nsp}$ is a nullspace torque, $\boldsymbol{x}_e = \boldsymbol{x} - \boldsymbol{x}_d$ is the error vector, and $(\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \ddot{\boldsymbol{x}}_d)$ is the desired trajectory, which the flying robot should follow. The pose error is defined as $\boldsymbol{x}_e = (\boldsymbol{\xi}_e^T \quad \boldsymbol{e}_R^T)^T$, where $\boldsymbol{\xi}_e = \boldsymbol{\xi} - \boldsymbol{\xi}_d$ is the translational position error. Using (3.19), the attitude error $\boldsymbol{e}_R$ may be deduced directly from given rotation matrices. For the controller gains it holds that $\boldsymbol{D} > \boldsymbol{0}$ and $\boldsymbol{K} > \boldsymbol{0}$, which ensures stability for inverse dynamics control. For other PD control approaches without reshaping, a diagonal stiffness gain might be required for the orientation dynamics (e.g. see example 3 on p. 15 of the report [178]).

In the second case, the task space control force (5.6) is distributed optimally to the robot manipulator and the flying robot. This case is treated in Section 5.4. The closed-loop error dynamics are directly obtained for $\boldsymbol{F}_{ext} = \boldsymbol{0}$ by inserting (5.7) in (5.4) as

$$\ddot{\boldsymbol{x}}_e + \boldsymbol{D}\dot{\boldsymbol{x}}_e + \boldsymbol{K}\boldsymbol{x}_e = \boldsymbol{0}, \tag{5.8}$$

which is the well-known damped harmonic oscillator. The equilibrium of (5.8) is stable for a positive definite damping matrix $\boldsymbol{D}$ and a positive definite stiffness matrix $\boldsymbol{K}$. It can be concluded that for closed-loop stability, the force $\boldsymbol{F}_u$ is arbitrary as long as the controller (5.7) is aware of it and if at the same time the torque limits are not reached. Nevertheless, if $\boldsymbol{F}_u$ interferes with the task at hand, the torque $\boldsymbol{\tau}_r$ and therefore the workload (or control effort) of the robot manipulator is increased. Furthermore, $\boldsymbol{\tau}_r$ has to comply with the joint torque limits $\boldsymbol{\tau}_{\max}$, which can be exceeded for a given trajectory and a heavy flying robot, i.e. for large m_u compared to the maximum payload of the robot manipulator. This leads to the question how the thrust vector $\boldsymbol{F}_u$ should be chosen in order to support the assistance task instead of disturbing it.

5.3 Workload sharing

The aforementioned overactuation of the combined system can be used to the share the control effort, or workload, of the assistance task between the robot manipulator and the flying robot. This is especially useful, if heavy flying robots are considered, which would otherwise exceed the maximum payload of the manipulator. In this section, a heuristic approach based on a decomposition of the combined system is presented.

5.3.1 Task space control based on the decomposed model

In order to independently treat the influence of the flying robot u on the robot manipulator r and vice versa, it is convenient to decompose the system dynamics (5.1) w.r.t. inertial effects. This can be done as shown in Section 2.4.5, using

$$\boldsymbol{M} = \tilde{\boldsymbol{M}}(\boldsymbol{m}_r) + \hat{\boldsymbol{M}}(\boldsymbol{m}_u), \tag{5.9}$$

$$\boldsymbol{C} = \tilde{\boldsymbol{C}}(\boldsymbol{m}_r) + \hat{\boldsymbol{C}}(\boldsymbol{m}_u), \tag{5.10}$$

$$\boldsymbol{g} = \tilde{\boldsymbol{g}}(\boldsymbol{m}_r) + \hat{\boldsymbol{g}}(\boldsymbol{m}_u), \tag{5.11}$$

with the mass of the robot manipulator $\boldsymbol{m}_r = (m_1 \quad \cdots \quad m_n)^T$ and the mass of the flying robot $\boldsymbol{m}_u = m_u$. Here, $\tilde{(\cdot)}$ denotes the parts related to the manipulator and $\hat{(\cdot)}$ the parts related to the flying robot. This yields

$$\tilde{\boldsymbol{M}}\ddot{\boldsymbol{\phi}} + \tilde{\boldsymbol{C}}\dot{\boldsymbol{\phi}} + \tilde{\boldsymbol{g}} + \hat{\boldsymbol{M}}\ddot{\boldsymbol{\phi}} + \hat{\boldsymbol{C}}\dot{\boldsymbol{\phi}} + \hat{\boldsymbol{g}} = \boldsymbol{\tau} \tag{5.12}$$

in configuration space and

$$\tilde{\boldsymbol{\Lambda}}\ddot{\boldsymbol{x}} + \tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{F}}_g + \hat{\boldsymbol{\Lambda}}\ddot{\boldsymbol{x}} + \hat{\boldsymbol{\mu}} + \hat{\boldsymbol{F}}_g = \boldsymbol{J}^{\#T}\boldsymbol{\tau} \tag{5.13}$$

in task space. This is applicable to the non-redundant combined system case, as pointed out in Section 2.4.5.

Other decompositions of the combined system dynamics may be applied as well, such as the transformation of coordinates considered in [47, 179, 180] or the passive decomposition presented in [49, 181]. However, both require the knowledge of the center of mass position of the combined system for controller design.

The decomposed system dynamics in task space (5.13) are utilized to design a control law that enables distributing the workload between the robot manipulator r and the flying robot u. Substituting the terms corresponding to the flying robot u in the controller (5.7), yields

$$\begin{aligned}\boldsymbol{\tau}_c = \boldsymbol{J}^T \Bigg\{ &\left(\tilde{\boldsymbol{\Lambda}} + \begin{bmatrix} \hat{\boldsymbol{\Lambda}}_{\xi\xi} & \hat{\boldsymbol{\Lambda}}_{\xi\Phi} \\ \hat{\boldsymbol{\Lambda}}_{\xi\Phi}^T & \hat{\boldsymbol{\Lambda}}_{\Phi\Phi} \end{bmatrix}\right) \left(\begin{pmatrix} \ddot{\boldsymbol{\xi}}_d \\ \dot{\boldsymbol{\omega}}_d \end{pmatrix} - \boldsymbol{D}\dot{\boldsymbol{x}}_e - \boldsymbol{K}\boldsymbol{x}_e \right) \\ &+ \left(\tilde{\boldsymbol{\mu}} + \begin{bmatrix} \hat{\boldsymbol{\mu}}_{\xi\xi} & \hat{\boldsymbol{\mu}}_{\xi\Phi} \\ \hat{\boldsymbol{\mu}}_{\Phi\xi} & \hat{\boldsymbol{\mu}}_{\Phi\Phi} \end{bmatrix}\right) + \tilde{\boldsymbol{F}}_g + \begin{pmatrix} \hat{\boldsymbol{F}}_{g,\xi} \\ \hat{\boldsymbol{F}}_{g,\Phi} \end{pmatrix} - \begin{pmatrix} \boldsymbol{F}_u \\ \boldsymbol{0} \end{pmatrix} \Bigg\}. \end{aligned} \tag{5.14}$$

5.3.2 Heuristic force distribution

Obviously, $\boldsymbol{F}_u$ could be used to directly apply the feed-forward acceleration terms and to compensate for parts of the Coriolis/centrifugal terms as well as for the flying robot's gravity vector with

$$\boldsymbol{F}_u = \hat{\boldsymbol{\Lambda}}_{\xi\xi}\ddot{\boldsymbol{\xi}}_d + \hat{\boldsymbol{\Lambda}}_{\xi\Phi}\dot{\boldsymbol{\omega}}_d + \hat{\boldsymbol{\mu}}_{\xi\xi} + \hat{\boldsymbol{\mu}}_{\xi\Phi} + \hat{\boldsymbol{F}}_{g,\xi}. \tag{5.15}$$

Moreover, $\boldsymbol{F}_u$ affects the rotational dynamics in task space via the coupling terms in $\boldsymbol{\Lambda}$ and $\boldsymbol{\mu}$. Note that the trivial solution $\boldsymbol{F}_u = \boldsymbol{0}$ is infeasible in practice, because $T = 0$ implies

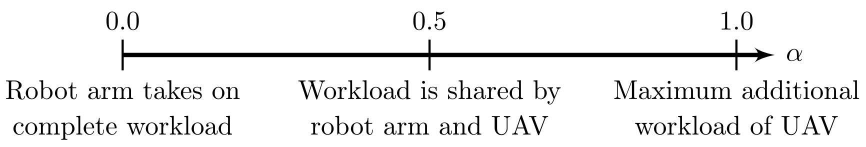

0.0
0.5
1.0
α
Robot arm takes on complete workload
Workload is shared by robot arm and UAV
Maximum additional workload of UAV

Figure 5.2: Illustration of the workload distribution factor $\alpha \in [0, 1]$. The workload of the robot-assisted take-off and landing tasks can be distributed dynamically between the robot manipulator and the flying robot, for instance, to comply with the robot's joint torque limits and to enable take-off and landing of heavy flying robots. Copyright © 2017 IEEE [214].

$\boldsymbol{\tau}_u = \mathbf{0}$. This is true for multicopters as well as for autonomous helicopters. For example, the cumulative thrust T of a multicopter is the sum of all rotor forces, while the multicopter's torque $\boldsymbol{\tau}_u$ results from the difference between the rotor forces. In comparison with the latter, the thrust T of a helicopter is the main rotor's force component along the body-fixed z-axis, while the helicopter's torque $\boldsymbol{\tau}_u$ depends on the main rotor force acting perpendicular to the tip-path-plane (TPP) and on the actual tilt angles of the TPP [153].

Therefore, from the possible choices of $\boldsymbol{F}_u$ the most practical one is selected, which does not require state feedback except for the vector of generalized coordinates $\boldsymbol{\phi}$:

$$\boldsymbol{F}_u = \alpha \hat{\boldsymbol{\Lambda}}_{\xi\xi} \ddot{\boldsymbol{\xi}}_d + \hat{\boldsymbol{F}}_{g,\xi}. \tag{5.16}$$

A scalar gain $\alpha \in [0, 1]$ is introduced, as defined in Figure 5.2, which scales the feed-forward acceleration and therefore allows to define the workload distribution between the robot manipulator and the flying robot.

5.3.3 Flying robot thrust vector control

Again, the desired thrust T in (5.3) is directly computed by projection onto the vertical axis of the flying robot $\boldsymbol{z}_u = \boldsymbol{R}_{iu}\boldsymbol{e}_3$, expressed in the inertial frame, via

$$T = \boldsymbol{z}_u^T \boldsymbol{F}_u. \tag{5.17}$$

In order to enable the flying robot to apply the force $\boldsymbol{F}_u$ in the inertial frame, the desired orientation $\boldsymbol{\Phi}_d$ of the combined system in task space has to be assigned accordingly. In

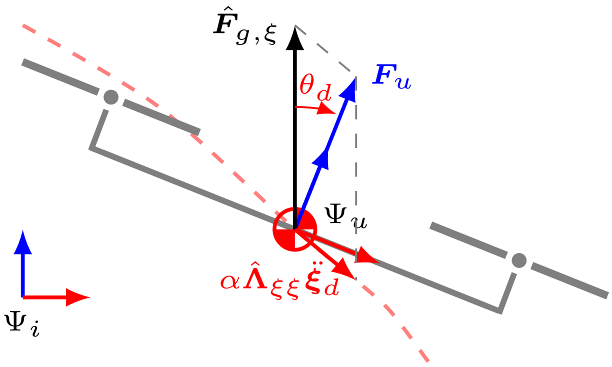

$\hat{\boldsymbol{F}}_{g,\xi}$
θ_d
$\boldsymbol{F}_u$
Ψ_u
$\alpha\hat{\boldsymbol{\Lambda}}_{\xi\xi}\ddot{\boldsymbol{\xi}}_d$
Ψ_i

Figure 5.3: Representation of the transformation (5.18) for thrust vector control (5.16). Copyright © 2017 IEEE [214].

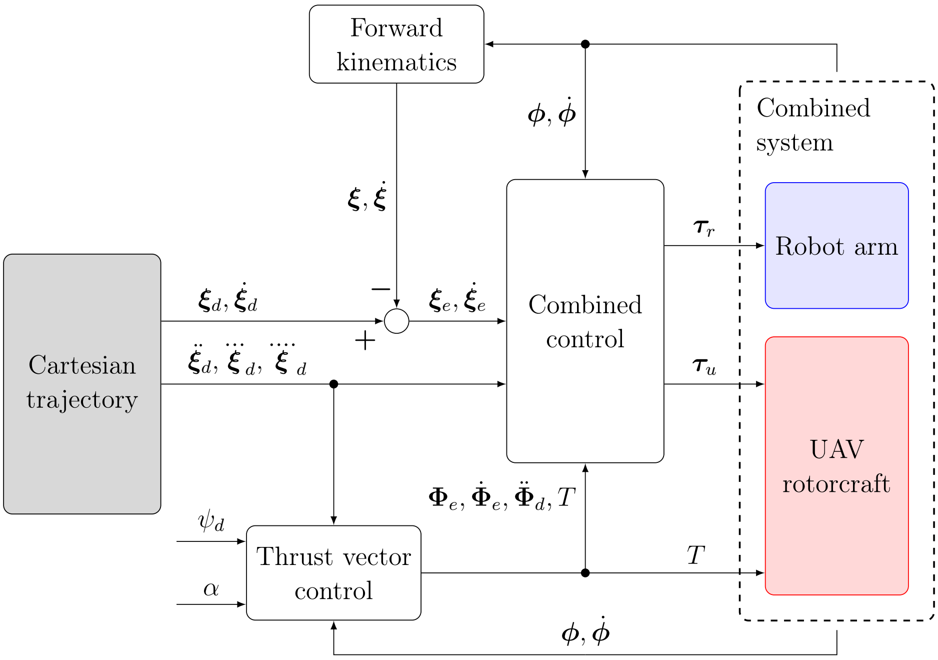


Figure 5.4: Final workload sharing control approach. Copyright © 2017 IEEE [214].

general, this is achieved using a transformation that aligns the aerial vehicle's thrust direction in the body frame $\boldsymbol{e}_3 = (0 \quad 0 \quad 1)^T$ with the desired force vector $\boldsymbol{F}_u$ (see Figure 5.3). The orientation about the vector $\boldsymbol{F}_u$, i.e. the heading ψ_d of the flying robot, can be chosen independently. Using Euler angles $\boldsymbol{\Phi}_d = (\varphi_d \quad \theta_d \quad \psi_d)^T$ and the XYZ convention, this can be expressed as [126]

$$\theta_d = \arcsin\left(\frac{\boldsymbol{F}_u^T \boldsymbol{e}_1}{||\boldsymbol{F}_u||_2}\right), \tag{5.18}$$

$$\varphi_d = -\arcsin\left(\frac{\boldsymbol{F}_u^T \boldsymbol{e}_2}{||\boldsymbol{F}_u||_2 \cos(\theta_d)}\right). \tag{5.19}$$

Alternative solutions using rotation matrices or unit quaternions for attitude representation are summarized briefly in Appendix A4. In order to compute $\dot{\boldsymbol{\Phi}}_d$ and $\ddot{\boldsymbol{\Phi}}_d$ via (5.16), (5.18), and (5.19), the given trajectory $\boldsymbol{\xi}_d$ has to be $\mathcal{C}^4$, i.e. four times continuously differentiable, which can be accomplished in practice using fifth order polynomials for example.

The computed desired orientation trajectory as well as the scalar thrust from (5.17) is then fed into the combined controller (5.7). Figure 5.4 illustrates the final workload sharing control approach presented in this section. Eq. (5.16), (5.17), (5.18), and (5.19) constitute the *thrust control* block, while the *combined control* block contains the control law (5.7).

5.3.4 Remarks for the practical implementation

The combined control law (5.7) requires the knowledge of the combined flying robot and robot manipulator model. It is assumed that the propulsion model of the flying robot, i.e. motor dynamics and propeller aerodynamics, is well identified [153]. In practice, aerial vehicles and manipulators have independent control systems. Therefore, a communication channel and open interfaces have to be present which allow to exchange states and model parameters.

The flying robot dependent terms in (5.7) can be included directly in the flying robot's attitude controller. With active gravity compensation, only the remaining parts of (5.7), (5.16), and (5.17), respectively, are transmitted and applied during task execution. This eliminates the risk of a crash caused by a complete communication loss.

The formulation of the thrust control law (5.16) is chosen because a minimum amount of states is needed for realization. In the current form, gravity is always compensated. The factor α regulates the force distribution between the aerial vehicle and the manipulator, but the total amount of force remains constant. Robustness against wind disturbances while satisfying maximum torque constraints can be increased by adding the translational parts of the stiffness and damping terms from (5.7) in (5.16). Nevertheless, the tracking performance will still depend on the total amount of stiffness and damping.

It is assumed that α is provided prior to task execution or during runtime by a higher level component. For instance, the joint torques can be monitored and if a defined limit is reached, α can be set accordingly. To generate a continuous attitude reference, α has to be at least two times continuously differentiable.

Since the task space controller (5.7) can include the inverse Jacobian, it is prone to configuration singularities, which can be avoided in practice e.g. by using damped least squares [182]. Additionally, torque optimization [77] can be used to resolve possible redundancy by generating corresponding nullspace torques.

One problem that arises for the heuristic force distribution, is the design of the workload sharing factor α. As mentioned before, it should be chosen, such that the actuator limits of the flying robot and the manipulator are not exceeded. Additionally, α needs to be differentiable in order to result in a sufficiently smooth reference attitude and control torque. One solution is to filter (5.16) using a higher order blend (e.g. a second-order filter), as shown

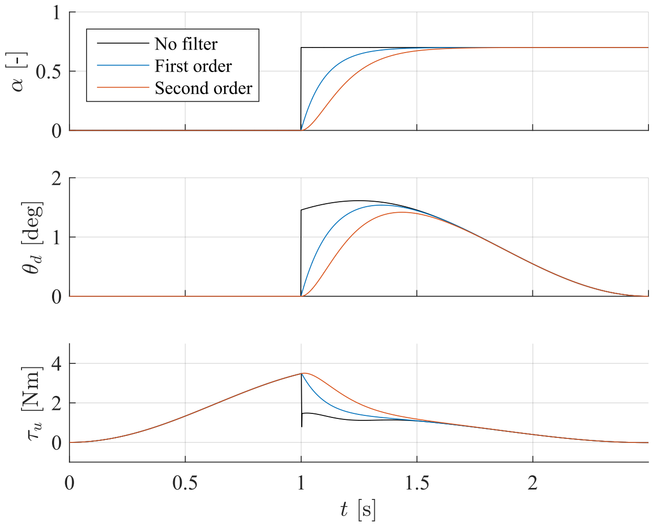


Figure 5.5: First and second order filters applied to α and resulting reference pitch angle θ_d and torque τ_u.

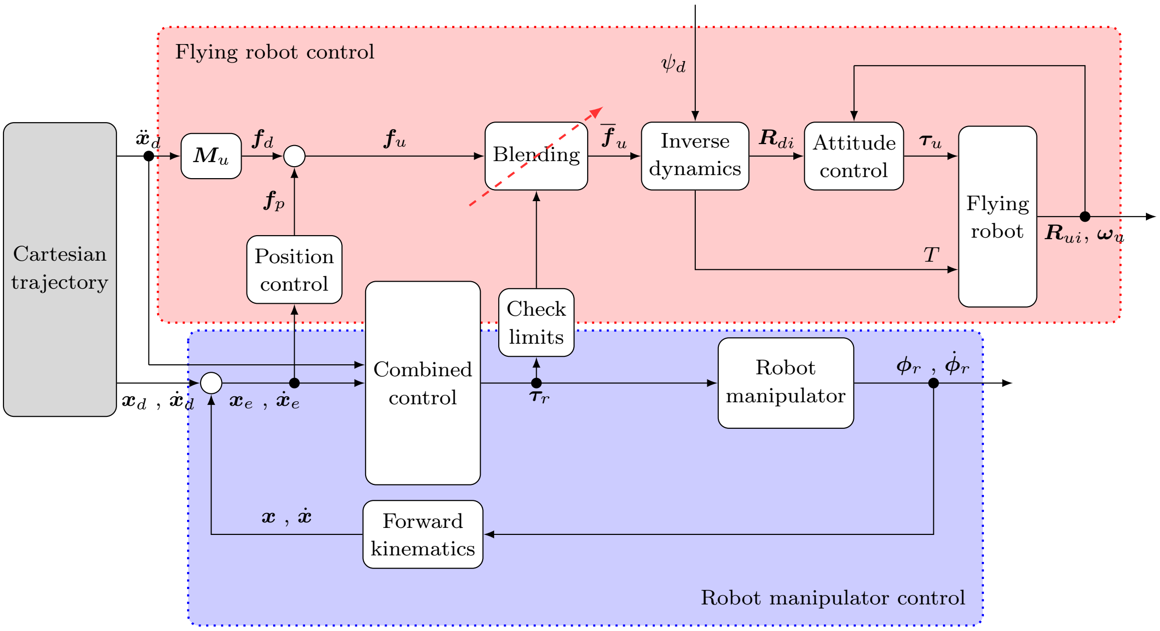


Figure 5.6: Heuristic workload sharing with a higher order blend of the control force. The blend is activated depending on a check of the actuator limits of both the flying robot and the robot manipulator.

in Figure 5.5. The latter is activated depending on an actuator limit check of both the flying robot and the robot manipulator. The final workload sharing approach is illustrated in Figure 5.6.

5.3.5 Simulation results

For numerical simulations, the dynamical model of a three link robot system in the $\xi_1\xi_3$-plane as depicted in Figure 5.7 is used. Here, the robot manipulator is composed of link l_1 and l_2, while the UAV is modeled as a point mass m_u at a distance l_3 from the universal hinge. All model parameters of the non-redundant system can be found in Table 5.2. In the plane, the UAV's orientation in task space is directly obtained as $\theta = \phi_{r,1} + \phi_{r,2} + \phi_u$. Moreover, a wind gust

$$\boldsymbol{F}_{wind} = \begin{pmatrix} -W \exp\left(-5(t - t_w)^2\right) \\ \boldsymbol{0}_{2\times 1} \end{pmatrix} \tag{5.20}$$

is introduced acting in negative ξ_1-direction with maximum amplitude W at time t_w. In (5.20), $\exp(\cdot)$ denotes the exponential function.

Using MATLAB/Simulink®, a straight-line landing trajectory as illustrated in Figure 5.9a and Figure 5.8 is simulated with and without wind disturbance and for different workload distribution factors $\alpha \in [0, 1]$. As can be seen in Figure 5.9a, the trajectory is accurately tracked for $\boldsymbol{K} = 200\,\boldsymbol{E}$ and $\boldsymbol{D} = \sqrt{200}\,\boldsymbol{E}$, with $\boldsymbol{E}$ being a 3×3 identity matrix. In terms of position tracking, $\alpha \to 1$ does not result in better performance, since (5.16) only includes feed-forward terms. However, $\alpha \to 1$ definitely results in a reduction of the manipulator's

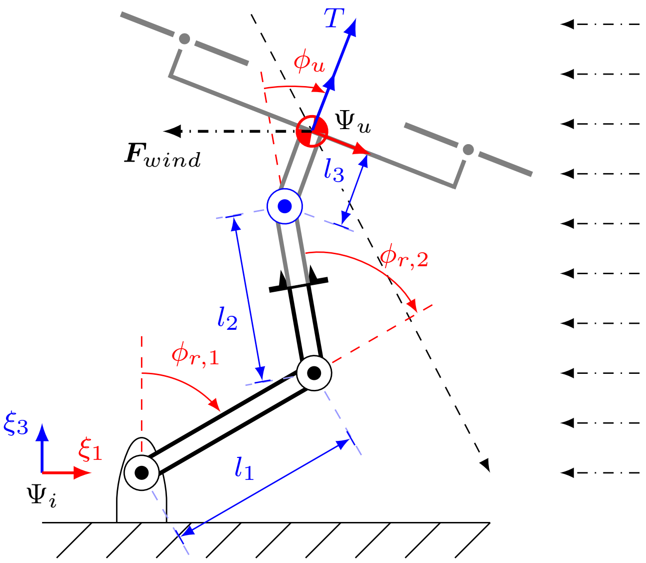


Figure 5.7: Planar three link robot system model. Copyright © 2017 IEEE [214].

Table 5.2: Model parameters used in the simulation.

l_1	l_2	l_3	$m_{r,1}$	$m_{r,2}$	m_u	g
0.6 m	0.6 m	0.25 m	8 kg	8 kg	40 kg	9.81 m/s^2

control effort. This can be seen from the controls depicted in Figure 5.9b and from the results in Table 5.3, which summarizes for $\alpha = 0$ and $\alpha = 1$

- mean squared error $e_{MSE} = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_e^2(t_k)$,
- standard deviation $\sigma = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N} |\boldsymbol{x}_e(t_k) - \mu|^2}$, with mean $\mu = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_e(t_k)$,
- maximum torque $\max(|\tau_j(t_k)|)$ in Nm,
- maximum thrust $\max(|T(t_k)|)$ in N,
- control effort of the robot manipulator $\Gamma_r = \boldsymbol{\tau}_{r,1}^T\boldsymbol{\tau}_{r,1} + \boldsymbol{\tau}_{r,2}^T\boldsymbol{\tau}_{r,2}$, and
- of the flying robot $\Gamma_u = \boldsymbol{\tau}_u^T\boldsymbol{\tau}_u + \boldsymbol{\tau}_{u,1}^{*T}\boldsymbol{\tau}_{u,1}^* + \boldsymbol{\tau}_{u,2}^{*T}\boldsymbol{\tau}_{u,2}^*$.

Therein, N is the number of logged data points from the simulation using a Runge-Kutta integrator with a fixed time step $\Delta t = 0.001\,\text{s}$, t_k is the discrete simulation time, and $\tau_j \in \{\tau_{r,1}, \tau_{r,2}, \tau_u\}$ denotes the torque acting at the jth joint of the combined system. The additional torques $\tau_{u,1}^*$ and $\tau_{u,2}^*$ are obtained from the mapping

$$\begin{pmatrix} \tau_{u,1}^* \\ \tau_{u,2}^* \end{pmatrix} = \boldsymbol{J}_{\xi\xi}^T \boldsymbol{R}_{iu} \begin{pmatrix} 0 \\ T \end{pmatrix}, \tag{5.21}$$

where $\boldsymbol{J}_{\xi\xi}$ is the translational part of the Jacobian and τ_u is the pitch torque of the UAV.

As expected, the position error is low with and without workload sharing and unaffected by the wind gust. The disturbance causes a deviation in orientation (see Figure 5.9a) and

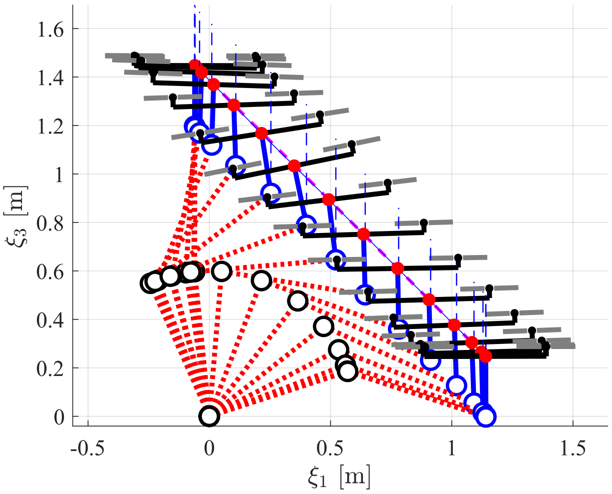
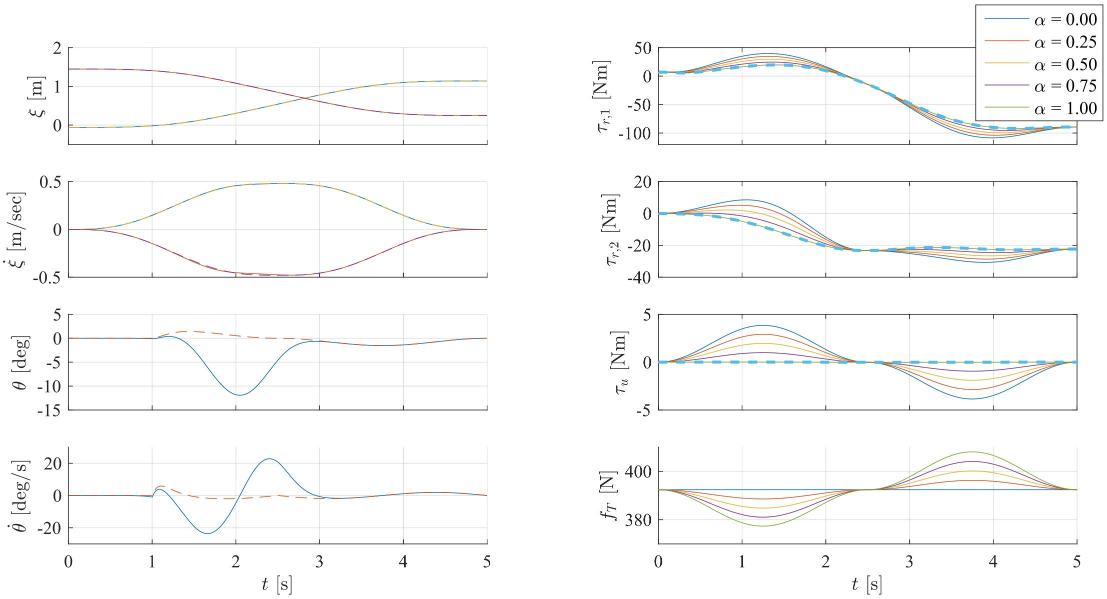


Figure 5.8: Visualization of the simulated landing maneuver using the workload sharing control approach with $\alpha = 1$ from $(\xi_1, \xi_3) = (-0.06\,\text{m}, 1.45\,\text{m})$ to $(\xi_1, \xi_3) = (1.14\,\text{m}, 0.25\,\text{m})$ within 5 seconds disturbed by a side wind gust with $W = 110\,\text{N}$ at $t_w = 2\,\text{s}$. The system is plotted every 0.3 s. Copyright © 2017 IEEE [214].



(a) Desired (dashed) and actual trajectory (solid) of the UAV's center of gravity with workload sharing ($\alpha = 1$) and under a side wind gust with $W = 110\,\text{N}$ at $t_w = 2\,\text{s}$. Note that the position trajectory (top) is accurately tracked despite the wind gust.

(b) Torques $\boldsymbol{\tau}_c = (\tau_{r,1}\ \tau_{r,2}\ \tau_u)^T$ of combined system and thrust force f_T of the flying robot for a given trajectory and different workload distribution factors α under no side wind ($W = 0$). Note that for $\alpha \to 1$ the torques decrease toward the case $m_u = 0$ (dashed), while the thrust increases.

Figure 5.9: Simulation results with workload sharing. (a) depicts the trajectory in the $\xi_1\xi_3$-plane and (b) shows the time series of the control inputs. Copyright © 2017 IEEE [214].

Table 5.3: Summary of the simulation results.

W	0 N		110 N	
α	0	1	0	1
	$\begin{pmatrix}\xi_1[\mathrm{m}^2] & \xi_3[\mathrm{m}^2] & \theta[\mathrm{rad}^2]\end{pmatrix}^T$		$\begin{pmatrix}\xi_1[\mathrm{m}^2] & \xi_3[\mathrm{m}^2] & \theta[\mathrm{rad}^2]\end{pmatrix}^T$	
e_{MSE}	$\begin{pmatrix}0.16\\0.13\\0.00\end{pmatrix}\cdot 10^{-10}$	$\begin{pmatrix}0.16\\0.10\\0.62\end{pmatrix}\cdot 10^{-10}$	$\begin{pmatrix}0.00\\0.00\\0.01\end{pmatrix}$	$\begin{pmatrix}0.00\\0.00\\0.01\end{pmatrix}$
	$\begin{pmatrix}\xi_1[\mathrm{m}] & \xi_3[\mathrm{m}] & \theta[\mathrm{deg}]\end{pmatrix}^T$		$\begin{pmatrix}\xi_1[\mathrm{m}] & \xi_3[\mathrm{m}] & \theta[\mathrm{deg}]\end{pmatrix}^T$	
σ	$\begin{pmatrix}0.04\\0.04\\0.00\end{pmatrix}\cdot 10^{-4}$	$\begin{pmatrix}0.04\\0.03\\4.49\end{pmatrix}\cdot 10^{-4}$	$\begin{pmatrix}0.00\\0.00\\3.76\end{pmatrix}$	$\begin{pmatrix}0.00\\0.00\\3.81\end{pmatrix}$
	$\begin{pmatrix}\lvert\tau_{r,1}\rvert[\mathrm{Nm}] & \lvert\tau_{r,2}\rvert[\mathrm{Nm}] & \lvert\tau_u\rvert[\mathrm{Nm}] & \lvert T\rvert[\mathrm{N}]\end{pmatrix}^T$		$\begin{pmatrix}\lvert\tau_{r,1}\rvert[\mathrm{Nm}] & \lvert\tau_{r,2}\rvert[\mathrm{Nm}] & \lvert\tau_u\rvert[\mathrm{Nm}] & \lvert T\rvert[\mathrm{N}]\end{pmatrix}^T$	
max	$\begin{pmatrix}108.42\\30.70\\3.85\\392.40\end{pmatrix}$	$\begin{pmatrix}91.95\\23.31\\0.01\\408.05\end{pmatrix}$	$\begin{pmatrix}108.42\\30.70\\21.74\\392.40\end{pmatrix}$	$\begin{pmatrix}91.95\\22.82\\20.74\\408.05\end{pmatrix}$
	$\begin{pmatrix}\Gamma_r[\mathrm{N}^2\mathrm{m}^2] & \Gamma_u[\mathrm{N}^2\mathrm{m}^2] & \Gamma_r+\Gamma_u[\mathrm{N}^2\mathrm{m}^2]\end{pmatrix}^T$		$\begin{pmatrix}\Gamma_r[\mathrm{N}^2\mathrm{m}^2] & \Gamma_u[\mathrm{N}^2\mathrm{m}^2] & \Gamma_r+\Gamma_u[\mathrm{N}^2\mathrm{m}^2]\end{pmatrix}^T$	
	$\begin{pmatrix}0.23\\5.63\\5.85\end{pmatrix}\cdot 10^{8}$	$\begin{pmatrix}0.16\\5.84\\6.00\end{pmatrix}\cdot 10^{8}$	$\begin{pmatrix}0.22\\5.92\\6.14\end{pmatrix}\cdot 10^{8}$	$\begin{pmatrix}0.16\\6.12\\6.28\end{pmatrix}\cdot 10^{8}$

leads to higher torques of the UAV. The effect of the wind gust on the orientation in task space depends on the corresponding controller gains. The simulations also reveal that the wind gust acting at the center of gravity of the UAV rotorcraft (see Figure 5.7) has minor influence on the robot manipulator due to the rotational hinge. Workload sharing decreases the maximum joint torques of the robot manipulator, which results in a lower control effort for the manipulator and a higher workload for the aerial vehicle. This also holds under side wind conditions. For $\alpha \to 1$ and $W = 0$ the torques decrease towards the case where no UAV is present, i.e. $m_u = 0$.

5.3.6 Experimental results

Experiments on a robot-assisted landing testbed as shown in Figure 5.11 are conducted. It consists of a custom-built hexacopter in a starlike configuration, as shown in Figure 5.10 with $m_u = 2\,\mathrm{kg}$ and 0.25 m rotor diameter, and a KUKA/DLR Light Weight Robot (LWR) manipulator with seven DoF and $m_r = 16\,\mathrm{kg}$. The hexacopter is fixed to the robot via a custom-built universal hinge (see Figure 5.10), which can be attached and detached using an electromagnet mounted at the end-effector of the manipulator. Details about the dynamical model of the LWR can for instance be found in [183]. The manipulator and the hexacopter are controlled at 1 kHz and 500 Hz, respectively. Data transfer between the robot manipulator and the UAV rotorcraft is realized using 2.4 GHz IEEE 802.11 WLAN.

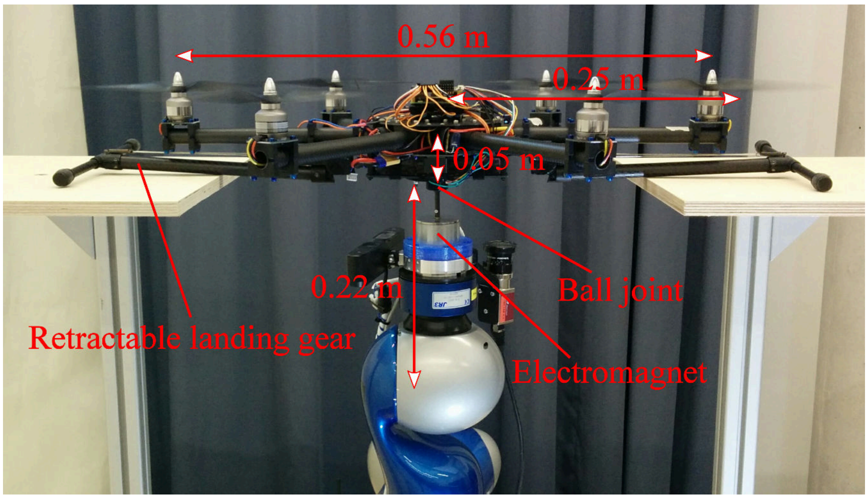


Figure 5.10: Robot-assisted landing experimental setup. Copyright © 2017 IEEE [214].

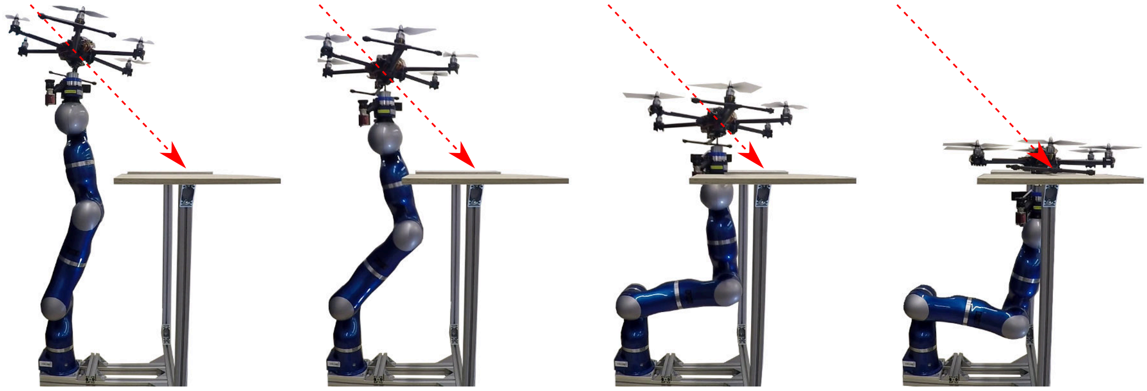
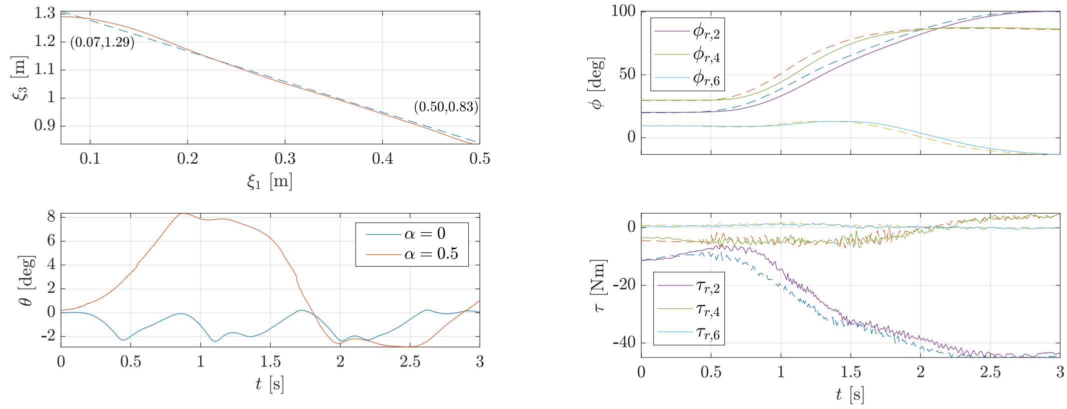


Figure 5.11: Exemplary landing maneuver performed with the robot-assisted landing testbed consisting of a hexacopter with $m_u = 2\,\mathrm{kg}$ and a LWR manipulator with $m_r = 16\,\mathrm{kg}$. The workload sharing control approach is used with $\alpha = 0.5$. Copyright © 2017 IEEE [214].

Again, a straight-line landing trajectory as shown in Figure 5.11 as well as in Figure 5.12a is used. Suitable controller gains found in the experiments are $\boldsymbol{K}_r = 1000\,\boldsymbol{E}$ and $\boldsymbol{D}_r = 0.5\,\boldsymbol{E}$ for the manipulator and $\boldsymbol{K}_u = 300\,\boldsymbol{E}$ and $\boldsymbol{D}_u = 20\,\boldsymbol{E}$ for the UAV rotorcraft, where $\boldsymbol{E}$ is an identity matrix of appropriate size. The differences in the gains are due to friction in the robot's joints as well as uncertainties in the control allocation of the UAV (i.e. the mapping of motor commands to applied forces and torques), for which a static thrust identification using a force-torque sensor was performed prior to the experiments. There, the hexacopters maximum thrust was identified to be 36 N. No nullspace resolution was used in the experiments because of the planar trajectory and friction in the joints.

Figure 5.12 shows the results of two experiments without ($\alpha = 0$) and with workload sharing ($\alpha = 0.5$). The desired position trajectory is tracked sufficiently well in both cases. For the case $\alpha = 0.5$, the orientation θ depicted in Figure 5.12b (bottom) illustrates how the UAV tilts backwards approximately in the middle of the trajectory in order to decelerate. The robot joint torques in both experiments in Figure 5.12b confirm that workload sharing reduces the control effort of the manipulator. The maximum torque difference in joint



(a) Desired (dashed) and actual (solid) landing trajectory from $(\xi_1, \xi_3) = (0.07\,\text{m}, 1.29\,\text{m})$ to $(\xi_1, \xi_3) = (0.50\,\text{m}, 0.83\,\text{m})$ within 3 seconds (top). The bottom plot shows the UAV's pitch angle θ for experiments with $(\alpha = 0.5)$ and without workload sharing $(\alpha = 0)$.

(b) Measured robot joint angles $q_{r,j}$ and torques $\tau_{r,j}$ with $j \in \{2, 4, 6\}$ for experiments with $(\alpha = 0.5$, solid) and without workload sharing $(\alpha = 0$, dashed). Because of the in-plane motion, the other joints are not affected and therefore omitted for clarity.

Figure 5.12: Experimental results with workload sharing. (a) depicts the trajectory in the $\xi_1\xi_3$-plane and (b) shows the robot's measured joint angles and torques. Copyright © 2017 IEEE [214].

two of the LWR is 6 Nm. The control effort of joints 2, 4, and 6 of the manipulator are $\Gamma_r(\alpha = 0) = 3.28 \cdot 10^5$ and $\Gamma_r(\alpha = 0.5) = 2.92 \cdot 10^5$. This is a reduction of 11 %.

Due to the lab environment, the UAV rotorcraft's mass in the experiment was limited to $m_u = 2\,\text{kg}$. Nevertheless, the results underline the findings of the simulations. For future investigations in the lab, the admissible torque of the LWR could be constrained artificially and wind disturbances may be included.

5.4 Optimal control allocation

Instead of leaving the choice of the workload distribution factor α to the designer or a higher level algorithm, it is more convenient to perform an online optimization. For this, different

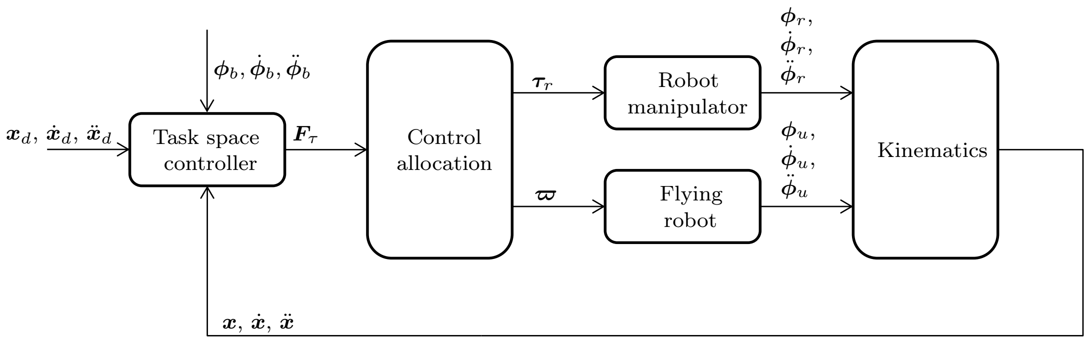

$\boldsymbol{\phi}_b, \dot{\boldsymbol{\phi}}_b, \ddot{\boldsymbol{\phi}}_b$
$\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \ddot{\boldsymbol{x}}_d$
Task space controller
$\boldsymbol{F}_\mathcal{T}$
Control allocation
$\boldsymbol{\tau}_r$
$\boldsymbol{\varpi}$
Robot manipulator
Flying robot
$\boldsymbol{\phi}_r, \dot{\boldsymbol{\phi}}_r, \ddot{\boldsymbol{\phi}}_r$
$\boldsymbol{\phi}_u, \dot{\boldsymbol{\phi}}_u, \ddot{\boldsymbol{\phi}}_u$
Kinematics
$\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}$

Figure 5.13: General overview of task space controller for the combined system with control allocation for the subsystems robot manipulator and flying robot.

possibilities are presented in the following. The control allocation problem is formulated as

$$\boldsymbol{Q}\boldsymbol{u} = \boldsymbol{\tau}_c, \tag{5.22}$$

with $\boldsymbol{\tau}_c = \boldsymbol{J}^T\boldsymbol{F}_\tau + \boldsymbol{\tau}_{nsp}$ for conciseness. The control allocation matrix $\boldsymbol{Q}$ is not invertible in general. Furthermore, $\boldsymbol{u}$ is the generalized control input to the combined system and

$$\boldsymbol{F}_\tau = \boldsymbol{\Lambda}(\ddot{\boldsymbol{x}}_d - \boldsymbol{D}\dot{\boldsymbol{x}}_e - \boldsymbol{K}\boldsymbol{x}_e) + \boldsymbol{F}_g + \boldsymbol{\mu}_d \tag{5.23}$$

is the output of the task space controller (cf. Figure 5.13). Nullspace resolution is realized via $\boldsymbol{\tau}_{nsp}$ and gravity compensation is included in $\boldsymbol{F}_\tau$ on the right-hand side of (5.22). Equation (5.22) represents the equality constraint of the optimization problem. It is important to note, that the actuator limits of the combined system are heterogeneous, i.e. for manipulators the joint torques are limited whereas for flying robots the rotor speeds, and hence, the rotor thrust forces are limited. Thus, the control input is $\boldsymbol{u} = (\boldsymbol{\tau}_r^T \quad \boldsymbol{\varpi}^T)^T$, where $\boldsymbol{\varpi}$ may be a vector of squared rotor speeds, as before, or contain generic control surface deflections. Let the force of the flying robot in the inertial frame be $\boldsymbol{F}_u$ and the produced torque in joint space be $\boldsymbol{\tau}_u$. Then, the general control allocation of a flying robot with N controls is given by

$$\begin{pmatrix} \boldsymbol{F}_{u_{3\times1}} \\ \boldsymbol{\tau}_{u_{3\times1}} \end{pmatrix} = \begin{bmatrix} \boldsymbol{R}_{iu_{3\times3}}\boldsymbol{B}_{f_{3\times N}} \\ \boldsymbol{B}_{\tau_{3\times N}} \end{bmatrix} \boldsymbol{\varpi}_{N\times1}. \tag{5.24}$$

Consider a manipulator with n joints. As mentioned before, the flying robot attached at the end-effector of the manipulator via a ball joint results in a joint space of the combined system with $n+3$ DoF. In order to derive the control allocation matrix $\boldsymbol{Q}$, (5.22) may be reformulated as

$$\begin{pmatrix} \boldsymbol{\tau}_{r_{n\times1}} \\ \boldsymbol{0}_{3\times1} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0}_{n\times1} \\ \boldsymbol{\tau}_{u_{3\times1}} \end{pmatrix} + \boldsymbol{J}^T_{(n+3)\times6} \begin{pmatrix} \boldsymbol{F}_{u_{3\times1}} \\ \boldsymbol{0}_{3\times1} \end{pmatrix} = \boldsymbol{J}^T_{(n+3)\times6}\boldsymbol{F}_{\tau_{6\times1}} + \boldsymbol{\tau}_{nsp_{(n+3)\times1}}. \tag{5.25}$$

Inserting (5.24) and rearranging yields

$$\begin{bmatrix} \boldsymbol{E}_{n\times n} \\ \boldsymbol{0}_{3\times n} \end{bmatrix} \boldsymbol{\tau}_{r_{n\times1}} + \underbrace{\left(\begin{bmatrix} \boldsymbol{0}_{n\times N} \\ \boldsymbol{B}_{\tau_{3\times N}} \end{bmatrix} + \boldsymbol{J}^T_{(n+3)\times6} \begin{bmatrix} \boldsymbol{R}_{iu_{3\times3}}\boldsymbol{B}_{f_{3\times N}} \\ \boldsymbol{0}_{3\times N} \end{bmatrix} \right)}_{\boldsymbol{\Sigma}_{(n+3)\times N}} \boldsymbol{\varpi}_{N\times1} \tag{5.26}$$

for the left-hand side, from which the control allocation matrix is obtained as

$$\boldsymbol{Q}_{(n+3)\times(n+N)} = \begin{bmatrix} \boldsymbol{E}_{n\times n} & \boldsymbol{\Sigma}_{(n+3)\times N} \\ \boldsymbol{0}_{3\times n} & \end{bmatrix}. \tag{5.27}$$

5.4.1 Analytical solution via Lagrange multipliers

In order to realize an optimal control allocation, consider the minimization problem

$$\min_{\boldsymbol{u}} \frac{1}{2}\boldsymbol{u}^T\boldsymbol{H}\boldsymbol{u} \quad \text{s. t. } \boldsymbol{\tau}_c - \boldsymbol{Q}\boldsymbol{u} = \boldsymbol{0}. \tag{5.28}$$

A solution can be found by applying the method of Lagrange multipliers [184]. The Lagrangian function G is defined as

$$G = \frac{1}{2}\boldsymbol{u}^T \boldsymbol{H}\boldsymbol{u} + \boldsymbol{\lambda}^T \boldsymbol{\tau}_c - \boldsymbol{\lambda}^T \boldsymbol{Q}\boldsymbol{u}, \tag{5.29}$$

where $\boldsymbol{\lambda}$ denotes the Lagrange multipliers. Setting the partial derivative of the Lagrangian w.r.t. the control input to zero, i.e.

$$\frac{\partial G}{\partial \boldsymbol{u}} = \boldsymbol{u}^T \boldsymbol{H} - \boldsymbol{\lambda}^T \boldsymbol{Q} = \boldsymbol{H}^T \boldsymbol{u} - \boldsymbol{Q}^T \boldsymbol{\lambda} = 0, \tag{5.30}$$

yields

$$\boldsymbol{u} = \boldsymbol{H}^{-T} \boldsymbol{Q}^T \boldsymbol{\lambda}. \tag{5.31}$$

Analogously, setting the partial derivative of the Lagrangian w.r.t. the Lagrange multipliers to zero, i.e.

$$\frac{\partial G}{\partial \boldsymbol{\lambda}} = \boldsymbol{\tau}_c - \boldsymbol{Q}\boldsymbol{u} = 0, \tag{5.32}$$

recovers the control allocation mapping (5.22). Inserting (5.31) in (5.22) and rearranging gives the expression for the Lagrange multipliers

$$\boldsymbol{\lambda} = \left(\boldsymbol{Q}\boldsymbol{H}^{-T}\boldsymbol{Q}^T\right)^{-1} \boldsymbol{\tau}_c. \tag{5.33}$$

Finally, by inserting (5.33) in (5.31), the control input $\boldsymbol{u}$ minimizing the above objective function is obtained as

$$\boldsymbol{u} = \underbrace{\boldsymbol{H}^{-T}\boldsymbol{Q}^T\left(\boldsymbol{Q}\boldsymbol{H}^{-T}\boldsymbol{Q}^T\right)^{-1}}_{\boldsymbol{Q}^{\#}} \boldsymbol{\tau}_c. \tag{5.34}$$

Therein, $\boldsymbol{Q}^{\#}$ is the weighted, right pseudoinverse of $\boldsymbol{Q}$, i.e. $\boldsymbol{Q}\boldsymbol{Q}^{\#} = \boldsymbol{E}$.

Least squares solution

For $\boldsymbol{H} = \boldsymbol{E}$, $\boldsymbol{Q}^{\#}$ in (5.34) is the Moore-Penrose pseudoinverse $\boldsymbol{Q}^+$ and $\boldsymbol{u} = \boldsymbol{Q}^+\boldsymbol{\tau}_c$ gives the least squares solution that minimizes $\frac{1}{2}\boldsymbol{u}^T\boldsymbol{u}$, i.e. the Euclidean norm of the control input. However, the least squares solution does not individually reduce the control effort of the manipulator or the flying robot, since it treats all control inputs equally.

Weighted least squares solution

The weighted least squares solution (5.34) involves the selection of an appropriate matrix $\boldsymbol{H}$ that distributes the control effort between the robot manipulator and the flying robot. In other words, instead of the workload distribution factor α, now $\boldsymbol{H}$ is the design parameter. However, it can not be assured that the control inputs do not exceed the actuator limits.

5.4.2 Constrained quadratic programming solution

In order to include upper and lower bounds

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}, \tag{5.35}$$

of the generalized control input, a numerical optimization can be performed. Equation (5.35) may be expressed as an inequality constraint, i.e.

$$\begin{bmatrix} \boldsymbol{E} \\ -\boldsymbol{E} \end{bmatrix} \boldsymbol{u} \leq \begin{pmatrix} \boldsymbol{u}_{\max} \\ -\boldsymbol{u}_{\min} \end{pmatrix}. \tag{5.36}$$

Thus, the minimization problem is given by

$$\min_{\boldsymbol{u}} \frac{1}{2} \boldsymbol{u}^T \boldsymbol{H} \boldsymbol{u} \quad \text{s. t.} \begin{cases} \boldsymbol{Q}\boldsymbol{u} = \boldsymbol{\tau}_c, \\ \begin{bmatrix} \boldsymbol{E} \\ -\boldsymbol{E} \end{bmatrix} \boldsymbol{u} \leq \begin{pmatrix} \boldsymbol{u}_{\max} \\ -\boldsymbol{u}_{\min} \end{pmatrix}, \end{cases} \tag{5.37}$$

which is obviously a quadratic problem with linear equality and inequality constraints.

The optimization problem stated above may be solved for instance using the open-source parametric quadratic programming (QP) software qpOASES [156], which uses an active set method. The software is highly efficient, especially for Model Predictive Control applications [164]. Other algorithms that can be used for numerical optimization are, for example, the interior point or trust region reflective methods [155].

5.4.3 Optimal reorientation of the flying robot

In the above control allocation (5.27), the instantaneous configuration $\boldsymbol{\phi}$ (included in $\boldsymbol{J}(\boldsymbol{\phi})$) of the combined system is used. Since the desired orientation of the end-effector $\boldsymbol{R}_{id}$ is not optimized, only the magnitude of the flying robot's thrust vector will vary, but not its direction. That also implies that the flying robot can only contribute to the assistance task in this direction. This is a fundamental difference compared to the workload sharing approach in Section 5.3.1, where the thrust vector direction changes depending on the workload sharing factor α.

However, reorientation of the flying robot can be achieved by adding a second optimization prior to (5.37) in the control loop, resulting in a sequential quadratic programming (SQP) approach. Instead of using (5.27), the control allocation problem is formulated for the force $\boldsymbol{F}_u$ in the inertial frame, such that

$$\boldsymbol{Q}_f \boldsymbol{u}_f = \boldsymbol{J}^T \boldsymbol{F}_\tau + \boldsymbol{\tau}_{nsp}, \tag{5.38}$$

with $\boldsymbol{u}_f^T = \begin{pmatrix} \boldsymbol{\tau}_r^T & \boldsymbol{\tau}_u^T & \boldsymbol{F}_u^T \end{pmatrix}$, $\boldsymbol{u}_{f,\min} \leq \boldsymbol{u}_f \leq \boldsymbol{u}_{f,\max}$, $\boldsymbol{F}_\tau$ given by (5.23), and

$$\boldsymbol{Q}_f = \begin{bmatrix} \begin{matrix} \boldsymbol{E}_{n\times n} & \boldsymbol{0}_{n\times 3} \\ \boldsymbol{0}_{3\times n} & \boldsymbol{E}_{3\times 3} \end{matrix} & \boldsymbol{J}^T \begin{bmatrix} \boldsymbol{E}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} \end{bmatrix} \end{bmatrix}. \tag{5.39}$$

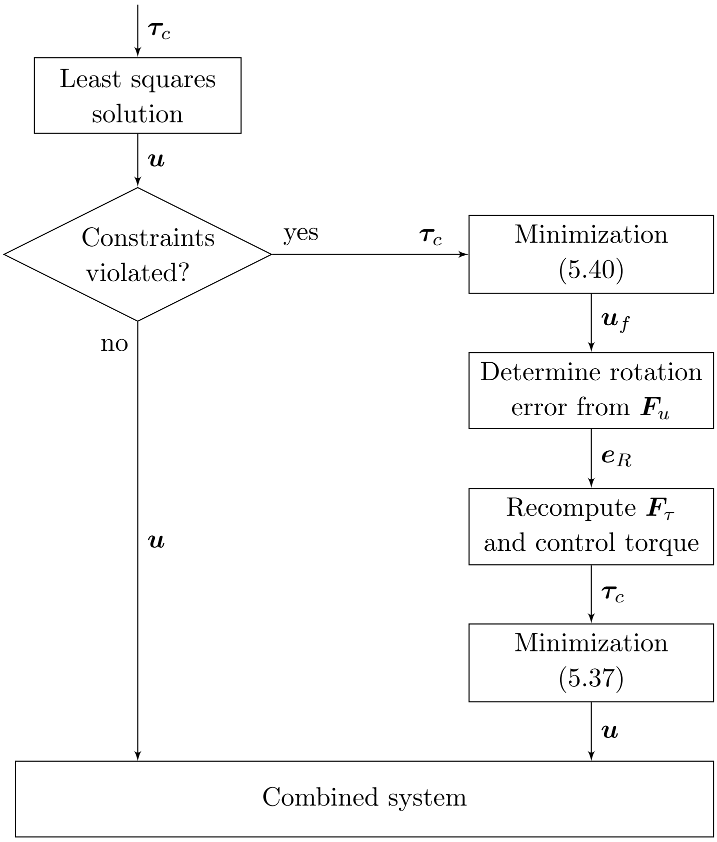


Figure 5.14: Sequential quadratic program for optimal control allocation.

The additional minimization problem can be stated as

$$\min_{\boldsymbol{u}_f} \frac{1}{2}\boldsymbol{u}_f^T \boldsymbol{H} \boldsymbol{u}_f \quad \text{s. t.} \begin{cases} \boldsymbol{Q}_f \boldsymbol{u}_f = \boldsymbol{\tau}_c, \\ \begin{bmatrix} \boldsymbol{E} \\ -\boldsymbol{E} \end{bmatrix} \boldsymbol{u}_f \leq \begin{pmatrix} \boldsymbol{u}_{f,\max} \\ -\boldsymbol{u}_{f,\min} \end{pmatrix} \end{cases}. \tag{5.40}$$

It is assumed that the constraints on $\boldsymbol{F}_u$ and $\boldsymbol{\tau}_u$ are independent, which is not the case in practice, as can be seen directly from (5.24). The actual constraints are imposed by subsequently solving the optimization problem (5.37). Figure 5.14 illustrates the resulting sequential quadratic program (SQP). The new desired orientation is computed from $\boldsymbol{F}_u$, which is the solution of (5.40), as described in Appendix A4. Note that the obtained desired force $\boldsymbol{F}_u$ might not be continuous, which could lead to problems stabilizing the orientation of the flying robot. This needs to be considered in the implementation. For example, the solution of the constrained problem can be mixed (see equation (27) in [185]) with the unconstrained least squares solution, which is known to be continuous. The attitude error is, again, determined by (3.19), which in turn is used in (5.23) to recompute $\boldsymbol{F}_\tau$. As long as the actuator limits are not exceeded, the least squares solution is used, since it is computationally cheaper and equal to the SQP solution without inequality constraints.

5.4.4 Adaptive control extension

As discussed extensively in Chapter 3, the thrust of flying robots includes uncertainties and is usually not feedback-controlled. To cope with this, the adaptive control approach presented in Section 3.5 is extended to the combined system (2.43).

Assume that the input of the combined system (5.4) is subject to a multiplicative uncertainty $\frac{1}{1+\varepsilon}$ (cf. (3.39) in Section 3.4 and (3.64) in Section 3.5.1), with constant $\varepsilon > -1$, since $(1+\varepsilon) \leq 0$ is physically irrelevant, such that [1]

$$(1+\varepsilon)\left(\boldsymbol{\Lambda}\ddot{\boldsymbol{x}} + \boldsymbol{\mu} + \boldsymbol{F}_g\right) = \boldsymbol{F}_\tau + (1+\varepsilon)\boldsymbol{F}_{ext}. \tag{5.41}$$

It is reasonable to assume a scalar parameter ε, because the uncertainty of the rotor thrust equally affects all 6 DoF. Recall the generalized passivity-based adaptive control law (3.76), which is now used to compute the desired force in task space

$$\boldsymbol{F}_\tau = (1+\hat{\varepsilon})\boldsymbol{\nu}, \tag{5.42}$$

where

$$\boldsymbol{\nu} = \boldsymbol{\Lambda}(\ddot{\boldsymbol{x}}_d - \boldsymbol{P}\dot{\tilde{\boldsymbol{x}}} - \boldsymbol{K}\dot{\tilde{\boldsymbol{x}}} - \boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{x}}) + \boldsymbol{F}_g + \boldsymbol{\mu}_d, \qquad \text{(inverse dynamics control)} \tag{5.43}$$

$$\boldsymbol{\nu} = \boldsymbol{\Lambda}\ddot{\boldsymbol{x}}_d - \boldsymbol{\Lambda}\boldsymbol{P}\dot{\tilde{\boldsymbol{x}}} - \boldsymbol{K}\dot{\tilde{\boldsymbol{x}}} - \boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{x}} + \boldsymbol{F}_g + \boldsymbol{\mu}_d, \qquad \text{(impedance control)} \tag{5.44}$$

with $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_d$ and positive definite $\boldsymbol{P} = \boldsymbol{P}^T$, positive definite $\boldsymbol{K} = \boldsymbol{K}^T$, and $\boldsymbol{\mu}_d = \boldsymbol{\mu} - \frac{1}{2}\dot{\boldsymbol{\Lambda}}\boldsymbol{s}$. Again, the final control input follows as $\boldsymbol{\tau}_c = \boldsymbol{J}^T\boldsymbol{F}_\tau + \boldsymbol{\tau}_n$ and the adaptation law is defined as

$$\dot{\hat{\varepsilon}} = -\gamma \boldsymbol{s}^T\boldsymbol{\nu}, \tag{5.45}$$

where $\boldsymbol{s} = \dot{\tilde{\boldsymbol{x}}} + \boldsymbol{P}\tilde{\boldsymbol{x}}$ is the combined tracking error. Passivity of the closed-loop dynamics w.r.t. the power port $(1+\varepsilon)\boldsymbol{s}^T\boldsymbol{F}_{ext}$ may be verified similar to Section 3.5.3. The proof including the derivation of the term $\boldsymbol{\mu}_d$ is omitted for brevity but can be found in Appendix A10.

5.4.5 Simulation results

In order to validate the methods presented in this section, a simulation of the multibody system composed of the base b, the LWR manipulator r, and the flying robot u is implemented as described below. The simulation is carried out using MATLAB/Simulink®, the Simulink implementation of the multibody dynamics algorithm of [32], and the MATLAB interface of qpOASES [156].

To simulate the interaction between the robot manipulator, the flying robot, and the floating base, a free-floating base model is introduced, that mimics the dynamics of a rigid body floating in water. It consists of a single rigid body fixed to the origin by spatial and rotational springs and dampers, as shown in Figure 5.15, and is excited by wave forces. Again, the JONSWAP wave spectrum ([92], p. 206) is used with wave height 1.5 m. In

[1] Note the scaling of the external force, due to the input uncertainty. For a pure model uncertainty, the scaling of the external force would not be present.

Table 5.4: Moving base model translational and rotational stiffness and damping.

k_{trans} [N/m]	k_{rot} [N/rad]	d_{trans} [N s/m]	d_{rot} [N s/rad]
400	40	20	4

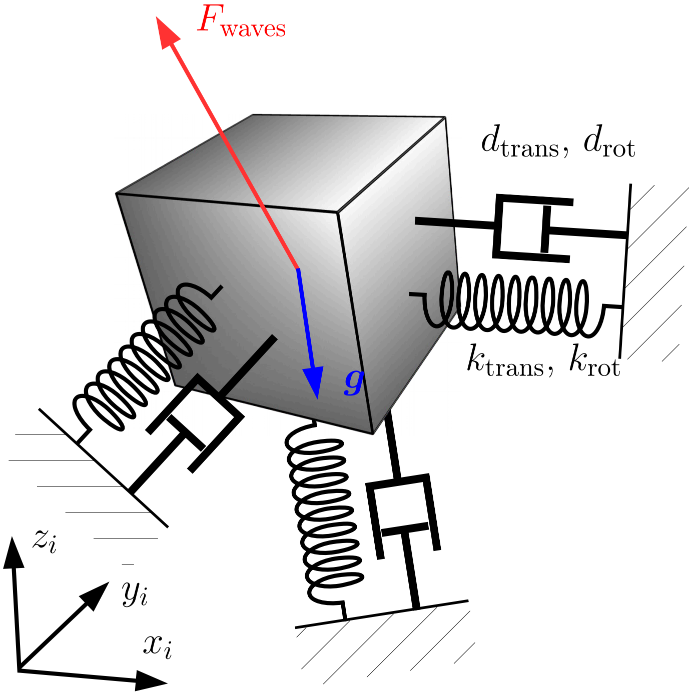

F_{waves}
d_{trans}, d_{rot}
k_{trans}, k_{rot}
$\boldsymbol{g}$
z_i
y_i
x_i

Figure 5.15: Analogous 3D floating base spring-damper model with wave excitation F_{waves}.

order to increase its effect on the base, the force in x_i-, y_i-, and z_i-direction is amplified by a factor of 20 and the moment about the z_b-axis is amplified by a factor of 2. The equilibrium between spring-damper reaction forces and gravity $\boldsymbol{g}$ is shifted by constantly applying $0.95\boldsymbol{g}$ to simulate buoyancy. The stiffness and damping values are given in Table 5.4 and are similar for the x_i-, y_i-, and z_i-directions.

The desired position trajectory is generated using fifth-order polynomials and describes a rectangle in Cartesian space, as shown in Figure 5.16. To assess the robustness of the closed-loop control, the multibody system is dropped into the water at the start of the simulation, as can be seen in Figure 5.16, Figure 5.18(a), and Figure 5.19(a),(b). The duration of the reference trajectory is 30 s. It is assumed that the motion of the base, its mass $m_b = 40\,\text{kg}$, and its inertia $I_b^{xx} = I_b^{yy} = I_b^{zz} = 6.67\,\text{kg}\,\text{m}^2$ are known exactly.

In simulation the following points are evaluated:

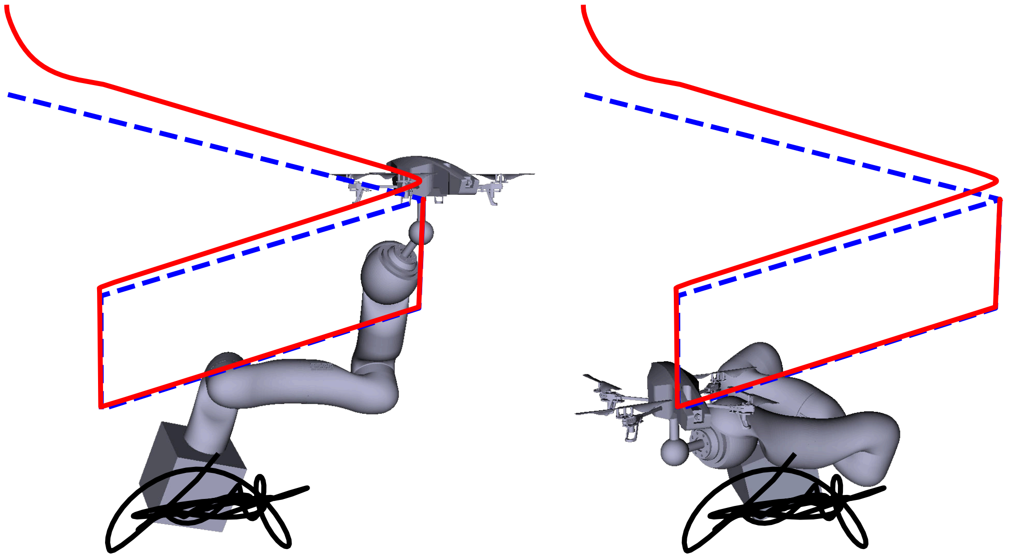
- inverse dynamics control or impedance control of the combined system,
- QP and SQP for optimal control allocation and reorientation of the flying robot, respectively, and
- adaptive control to cope with the uncertainty in the thrust of the flying robot.

Table 5.5: Actuator limits used in the simulation, with $\boldsymbol{\tau}_{r,\min} = -\boldsymbol{\tau}_{r,\max}$ and $\boldsymbol{T}_{\min} = \mathbf{0}$.

	$\tau_{r,\max}$ [Nm]							$T_{\max}$ [N]			
Joint / rotor number	1	2	3	4	5	6	7	1	2	3	4
Limit	165	100	70	70	70	30	30	150	150	150	150

Table 5.6: Artificial limits for optimal reorientation, with $\boldsymbol{\tau}_{u,\min} = -\boldsymbol{\tau}_{u,\max}$ and $\boldsymbol{F}_{u,\min} = \mathbf{0}$.

	$\tau_{u,\max}$ [Nm]			$F_{u,\max}$ [N]		
Axis	x_i	y_i	z_i	x_i	y_i	z_i
Limit	24	24	12	200	200	400



(a) $t = 6\,\mathrm{s}$

(b) $t = 14\,\mathrm{s}$

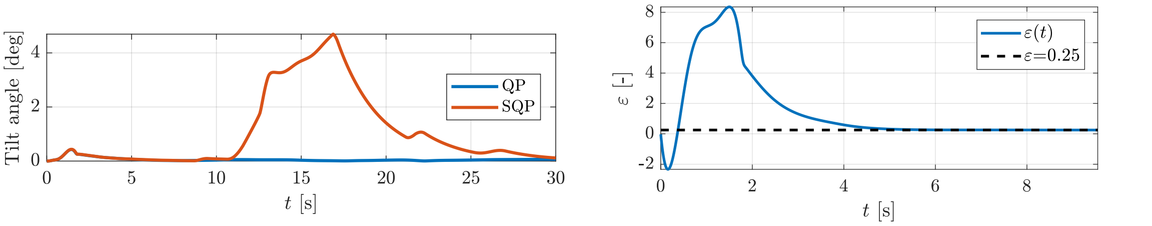
Figure 5.16: Combined system and rectangular reference trajectory (desired $--$, actual $-$, base movement $-$).

Table 5.7: Root Mean Square Error (RMSE) of the flying robot's position in simulation.

RMSE [m] without adaptive control			RMSE [m] with adaptive control		
x_i	y_i	z_i	x_i	y_i	z_i
0.0034	0.0033	0.0919	0.0030	0.0029	0.0859

The actuator limits considered for optimal control allocation are listed in Table 5.5 for the single QP approach and in Table 5.6 for the additional optimal reorientation of the flying robot (SQP approach). For both optimizations, $\boldsymbol{H}$ is a square identity matrix of appropriate size. The maximum rotor thrust forces and the parameters $m_u = 40\,\mathrm{kg}$, $I_u^{xx} = I_u^{yy} = I_u^{zz} = 6.67\,\mathrm{kg\,m^2}$ of the flying robot result in a realistic thrust to weight ratio of 1.5 (cf. Table 5.5). The dynamic properties of the LWR can be found for example in [183].

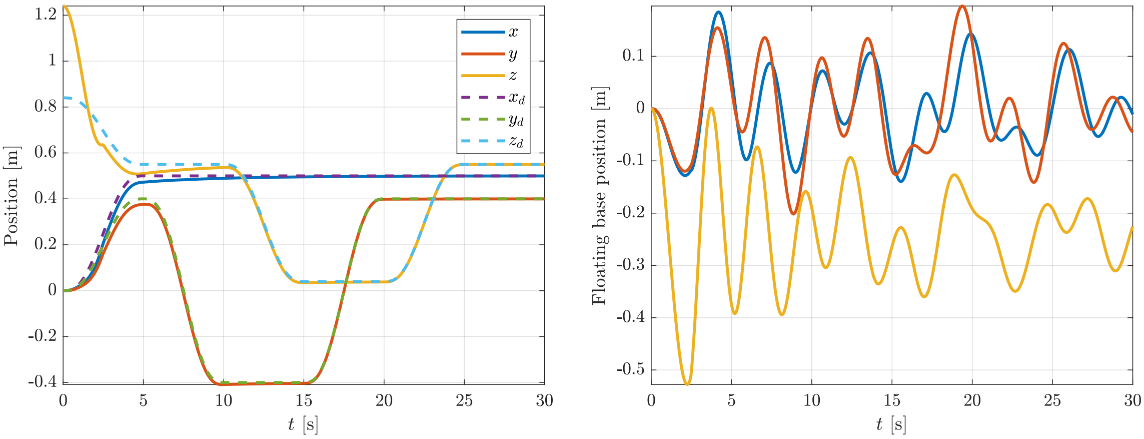
In Figure 5.18, Figure 5.19, and Figure 5.17 the results of various simulation runs are depicted. Figure 5.18(a) shows that the position converges using the impedance controller and that the reference trajectory is tracked well even under large random base motion



(a) Comparison of tilt angle with single QP and SQP, i.e. constant orientation and optimal reorientation of the flying robot.

(b) Convergence of adaptive parameter $\varepsilon \to 0.25$ due to artificial uncertainty of thrust $0.8\boldsymbol{F}_u = \frac{1}{1+\varepsilon}\boldsymbol{F}_u$ and with adaptation gain $\gamma = 0.1$.

Figure 5.17: Tilt angle of flying robot (a) and adaptive parameter ε (b).

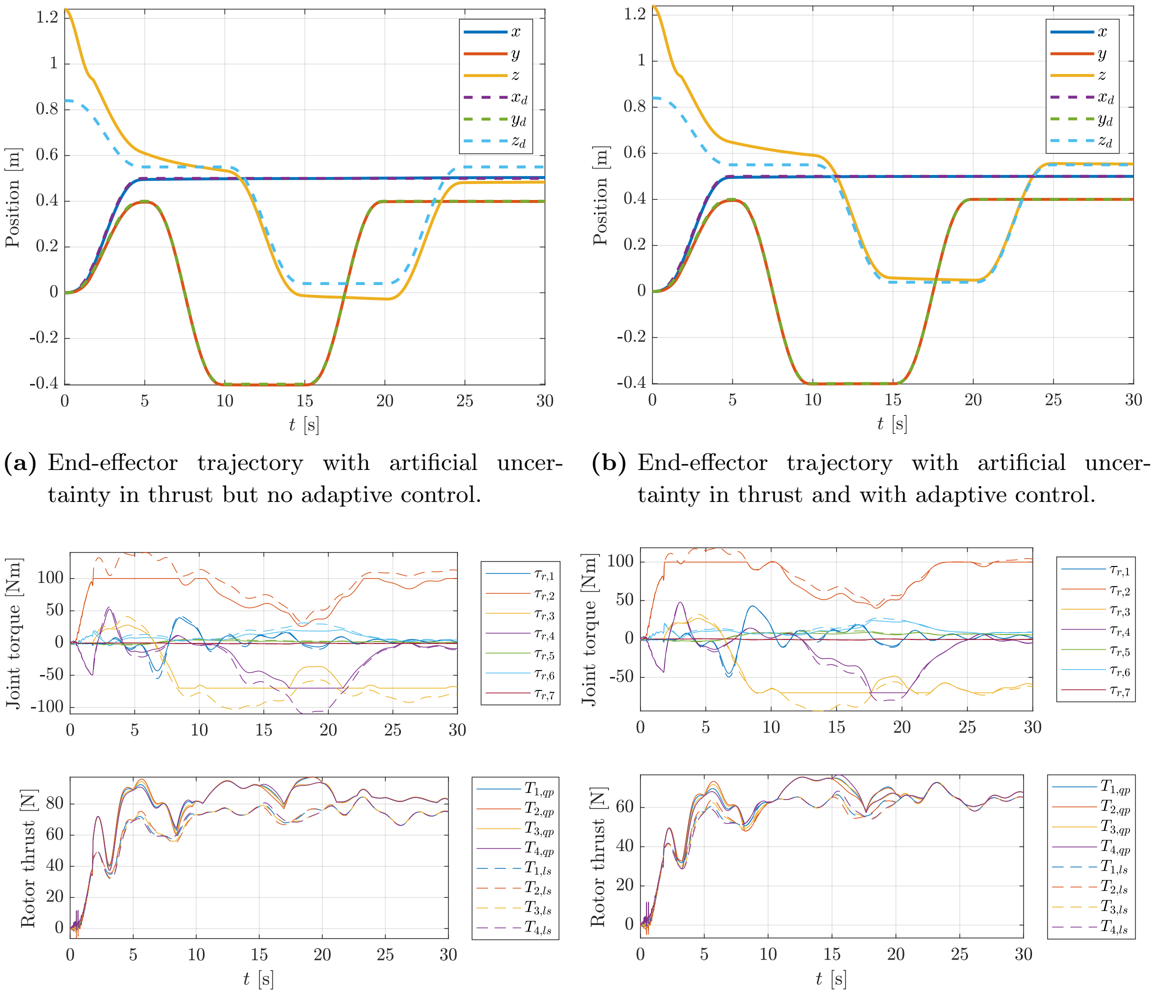


(a) Reference and actual Cartesian end-effector trajectory without uncertainties.

(b) 3D position of floating base excited by spring-damper and wave forces.

Figure 5.18: Results of the multibody simulation without artificial uncertainties.

(cf. Figure 5.18(b)) but without uncertainties. Figure 5.19(a) and (b) compare the position tracking accuracy under artificial uncertainty of the rotor thrust without and with adaptive control (see Section 5.4.4), respectively. As expected, the reference trajectory is tracked more precisely with adaptive control (see the RMSE in Table 5.7 for comparison) at the cost of increased rotor thrust, as indicated by the joint torques and rotor thrust forces shown in Figure 5.19(c) and (d). From the latter, it can also be seen that the control allocation using QP (see Section 5.4.2) results in actuator commands that comply with the limits defined in Table 5.5. In Figure 5.17(a) the tilt angle with QP control allocation is compared to optimal reorientation of the flying robot using SQP (see Section 5.4.3 and Figure 5.14). The tilt angle computed by the SQP approach is rather small and, hence, likely to be indistinguishable from noise or disturbances. This suggests that the SQP may be omitted in practice resulting in a computationally cheaper implementation. Finally, Figure 5.17(b) depicts the adaptive parameter ε of the controller (5.42), which converges exactly to the value of the simulated uncertainty of the thrust force.



(a) End-effector trajectory with artificial uncertainty in thrust but no adaptive control.

(b) End-effector trajectory with artificial uncertainty in thrust and with adaptive control.

(c) Actuator commands using least squares (dashed, index ls) and constrained QP solution (solid, index qp).

(d) Actuator commands with adaptive control using least squares (dashed, index ls) and constrained QP solution (solid, index qp).

Figure 5.19: Results of the multibody simulation with artificial uncertainties.

5.4.6 Extensions for practical implementation

Some assumptions made in the derivation of the combined control approach presented above might prevent a practical implementation. They are addressed and suitable extensions are discussed in this section.

Identifying or neglecting the base inertia

The inertia of the base is included in the combined system model and, so far, it is assumed to be known. For a robot manipulator operating on a ship, this is rather unrealistic. In order to still apply the combined control approach, an online parameter estimation ([86], p. 358) may be performed or the adaptive control approach [186], ([88], p. 277) can be applied. The latter additionally allows to consider uncertainty in the manipulator's or the flying robot's inertia. Another possibility is to omit the base inertia in the combined model, to assume that the base motion is not influenced by the manipulator or the flying robot, and instead to measure and compensate the base acceleration, as done in Section 4.4.2 and [55].

Estimation of the base motion

Similar to its inertia, the motion of the base is assumed to be known. It is reasonable to assume, that the base motion can be measured by off-the-shelf sensor systems, such as IMUs and Differential GPS (DGPS or DGNSS) with Real Time Kinematics (RTK). Furthermore, the ship motion can be estimated and predicted using auto-regression [56] or Kalman filters [187, 188]. The latter requires only acceleration measurements and no vessel specific parameters.

Admittance interface to the flying robot

The aforementioned uncertainty in rotor thrust also holds for the torques produced by the flying robot. Since they are often not feedback-controlled, the torques commanded by the combined control approach might not be applied exactly by the flying robot. To overcome this issue, an admittance interface can be used, which accepts torque inputs and generates a corresponding angular velocity reference. The latter is commanded to a low-level closed-loop rate controller. This approach is inspired by [177], where a torque-controlled humanoid robot on a velocity-controlled base is considered. The main difference here is that the flying robot is the last element in the kinematic chain. The parameters of the admittance model can be chosen arbitrarily, but have to be included in the combined system dynamics instead of the real model of the flying robot. This also allows to mimic a heavier or lighter flying robot.

Estimating or bypassing the transformation between the UAV and the manipulator

Another issue unconsidered so far is the estimation of the transformation between the flying robot and the end-effector of the manipulator. Depending on how the connection between the two systems is established, e.g. using a gripper, an electromagnet, or a docking interface, this transformation is unknown a priori and needs to be measured. For this, different techniques can be used, for example using fiducial markers at the end-effector and a downward facing camera mounted on the flying robot, as proposed in Chapter 6, or vice versa. Alternatively, the difference between absolute end-effector and absolute flying robot orientation estimates can be used, but their accuracy might be insufficient depending on the precision of the magnetometer for heading estimation. A completely different solution is to reformulate the problem by choosing the end-effector of the robot manipulator as reference point, command a constant orientation of the flying robot, and only vary the magnitude of the vertical thrust force, which is independent from the orientation. The last approach is motivated by the fact that the tilt angle of the flying robot, as observed in the simulations, is negligibly small and might not be distinguishable from noise. This observation is substantiated especially by the simulation results with optimal reorientation presented in Section 5.3.5 and Section 5.4.5.

5.5 Summary and conclusion

In this chapter, a task space control approach for robot manipulator assisted take-off and landing of flying robots is presented. Both inverse dynamics and impedance control are directly applicable to the redundant system. The combined model of the flying robot, the robot manipulator, and the free-floating base considers all coupling terms in the dynamics. This implies that an accurate model is required. Moreover, it is assumed that the motion of the base, i.e. its pose, velocity, and acceleration, can be measured or estimated. Two options for controlling the combined system are presented, which allow to share the control effort between the flying robot and the robot manipulator: a heuristic workload sharing approach, which uses an inertia decomposition and can be solved analytically, and a numerical solution, for which the control allocation is formulated as a quadratic minimization problem with equality and inequality constraints.

The performance of both solutions is evaluated in simulations and experiments with a light weight manipulator and a custom-built hexacopter. Both approaches are especially useful if heavy flying robots are considered, since the required actuator torque of the manipulator is decreased significantly. The first solution is computationally less expensive and requires less state information to be exchanged between the flying robot and the robot manipulator, but it is restricted to non-redundant systems and inequality constraints are difficult to enforce. The second solution is also applicable to redundant systems, allows to include inequality constraints, and is optimal w.r.t. a quadratic cost function. However, the optimization is computationally expensive, which could potentially affect the real-time performance. So far, it is evaluated using MATLAB/Simulink® on a desktop computer with Intel Xeon CPU E5-1620 @ 3.60 GHz. A second optimization is proposed for optimal reorientation of the flying robot instead of commanding a constant attitude. This results in a SQP approach. However, in simulation only a negligibly small reference tilt angle is computed. This substantiates the conjecture, that the second optimization can be omitted for the sake of a less complex implementation.

To cope with uncertainties of the flying robot's rotor thrust, an adaptive parameter is added to the task space controller. In a simulation including artificial uncertainties and large, random base motions, the position error converges to zero with the adaptive controller, whereas with the nominal impedance controller, a steady-state error of approximately 8 cm remains. In addition, with the adaptive control approach, the closed-loop dynamics are shown to be passive. Finally, possible extensions are summarized, which enable a practical implementation of the presented combined control approaches. Further experimental evaluation is part of future work.

"Computers are able to see, hear and learn.
Welcome to the future."
— Dave Waters

6 In-flight capturing of a flying robot

This chapter is devoted to autonomous in-flight capturing of a flying robot by means of a robot manipulator fixed to the landing surface. Hence, the initial phase of the three phases of robot-assisted landing presented in Section 2.3 is considered. In order to robustly catch the flying robot, the relative distance between the end-effector of the manipulator and the docking interface of the UAV needs to be measured accurately. Moreover, the state estimation should be reliable and has to account for time delays in the acquisition and processing of the sensor data.

To achieve this, an object tracking approach based on computer vision is proposed in Section 6.1. It overcomes the major limitations of external tracking systems discussed in Section 6.1.1. By using multiple fiducial (or artificial) markers as presented in Section 6.1.2, robustness against occlusion of single markers is achieved. Furthermore, the well-known Kalman filter is used in Section 6.1.3 for fusing the distance measurements with the visual inertial odometry of the flying robot. This enables to estimate the pose of the flying robot at a higher update rate compared to the frame rate of the camera used for marker tracking.

Combining marker tracking with the Cartesian impedance controller (2.87) and (4.100) presented in Section 2.6 and Section 4.4.2, respectively, results in visual servo control as shown in Section 6.2. In Section 6.2.1, the maximum attainable translational velocities of the DLR Light Weight Robot (LWR) are computed based on an optimization. The latter allows to visualize and assess the capabilities of a given manipulator, i.e. at what speed the flying robot may be caught by the manipulator. The Cartesian impedance controller is extended in Section 6.2.2 to include the relative pose estimate acquired using visual marker tracking. The workspace of the manipulator is constrained artificially in order to avoid singular configurations, like the outstretched configuration.

For completeness, different docking interfaces between flying robot and manipulator are discussed briefly in Section 6.3. An electromagnet is selected due to its generality and robustness. In Section 6.4, the multi-marker tracking algorithm and the visual servo control approach are evaluated in experiments with a DLR/KUKA LWR, a custom docking interface, an off-the-shelf camera, and a quadrocopter.

6.1 Robust visual tracking of the flying robot

In this section, an overview of available pose measurement methods, the selected visual tracking approach, and the implemented Kalman filter for sensor fusion are presented.

6.1.1 Overview of pose measurement approaches

Different pose, i.e. position and orientation, estimation methods are available today. The considered systems with main properties and disadvantages are listed in Table 6.1. The well-known Global Positioning System (GPS) in combination with inertial measurements of an IMU enables pose estimation of flying robots [100]. However, GPS is only available outdoors and at a low rate. Its accuracy is deemed insufficient for in-flight capturing of a flying robot. On the other hand, external tracking systems, like Vicon [100], A.R.T. [109], and the HTC Vive Lighthouse [189] provide precise pose estimates at a high rate, but they are usually only available indoors. Moreover, they require additional external components, which need to be installed and calibrated. The same holds for Ultra Wide Band (UWB) localization [190]. The maximum update rate of UWB is comparable to external tracking systems, but its accuracy is lower and it only provides position measurements.

Visual odometry (VO) [217], i.e. determining the position and orientation of a vehicle based on camera images, overcomes the main drawbacks of the systems mentioned above. It may be used indoors and outdoors and can achieve high accuracy. The update rate is limited by camera frequency and processing time, but can be increased using a sufficiently fast camera or sensor fusion with an IMU. The sensitivity w.r.t. lighting conditions is a problem, for example at night, but can be relaxed by using spotlights for illumination. Objects in the

Table 6.1: List of established pose measurement systems.

System	Description	Max. rate	Accuracy	Drawbacks
Vicon, A.R.T.	Motion capture systems using external infrared cameras and passive markers	250 Hz	1 mm	not available outdoors
HTC Vive Lighthouse	Tracking system for virtual reality devices using infrared emitters and receiver diods	250-1 kHz	1 mm	only area of 4 m by 4 m
Ultra Wide Band (UWB)	High frequency radio messages and runtime measurements between anchors	1 kHz	10-30 cm	provides only position
GPS, Galileo, GLONASS, BeiDou	Global positioning systems using runtime of radio signals emitted by satellites	1-10 Hz	1-5 m	only position with low accuracy
Visual odometry	Feature tracking in mono or stereo camera images to obtain a velocity estimate	1-200 Hz	1 cm	sensitive to illumination

camera images can be tracked using fiducial markers [191]. Recent advances in computer vision, e.g. using deep neural networks, also enable tracking of generic objects without the need for artificial markers [192]. However, the robustness and reliability of marker-less approaches is difficult to assess. Since the installation of fiducial markers on the ground and on the flying robot seems reasonable, visual marker tracking is selected for measuring the relative distance between the end-effector of the manipulator and the flying robot. The camera may be installed at the end-effector or on the ground. These two options only differ in the required extrinsic calibration.

6.1.2 Multi-marker tracking algorithm

Today, many software implementations of fiducial marker tracking algorithms are readily available that can be impplemented on the ground station computer. In [191], a comparison of different marker types, called tags, is presented. Due to their fast computation, the ARToolKitPlus [193] and AprilTags [194] are considered in this work. Artificial markers are a reliable and accurate method to detect and localize an object using a monocular camera. However, the tags have to be fully visible in the image in order to be successfully detected and localized [191]. The probability of one marker on the flying robot being outside the field of view of the camera increases with decreasing distance between flying robot and camera.

To increase the probability of a marker being detected, multiple markers are used. They are attached at the bottom of the flying robot, as shown in Figure 6.1. During the initial approach phase, described in Section 2.3, all markers can be localized. One marker is defined as a reference and the relative positions and orientations of the other markers with respect to this marker are calculated and stored. Using these relative positions and orientations together with the current poses of one or more previously localized markers, the pose of

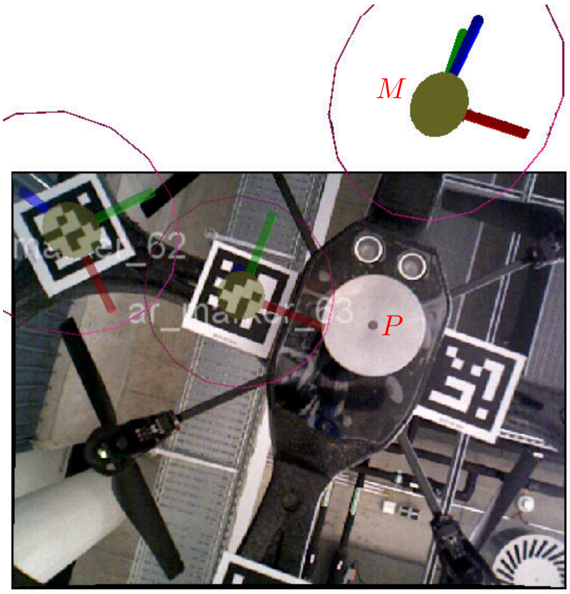
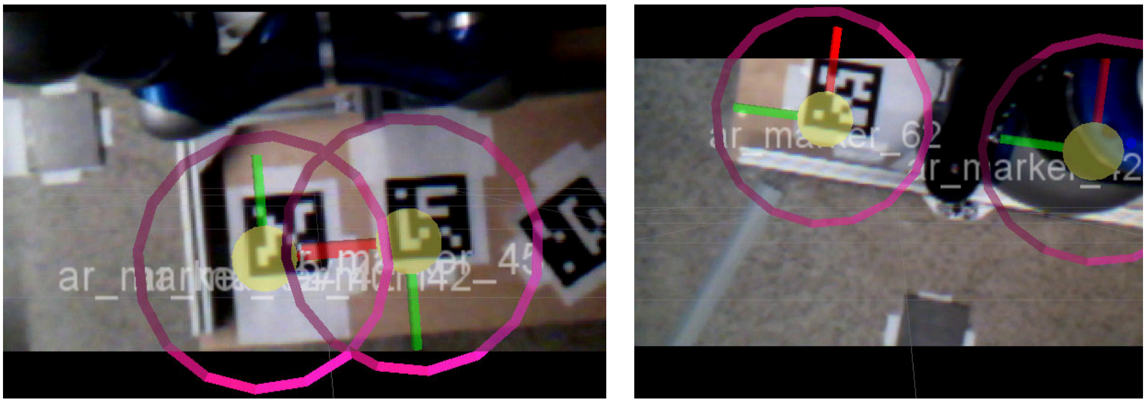


Figure 6.1: Tracked markers on the bottom of the flying robot as seen by the camera at the end-effector of the robot arm. The coordinate systems of two localized markers and of the reference marker M, which is localized although it is outside the field of view, are visualized.



(a) Two markers are fully visible, one is not detected. **(b)** One marker is occluded by the robot arm.

Figure 6.2: Tracked markers on the ground as seen by the camera on the bottom of the flying robot.

the reference marker can be computed, even if it is outside the field of view. The detected markers and the marker coordinate systems are shown in Figure 6.1. It can be seen that the reference marker outside of the field of view of the camera is also localized. If more than two markers are detected simultaneously, a RANSAC [195] algorithm is used to identify and delete outliers in the marker localization. To position the flying robot relative to the robot manipulator, e.g. during the approach phase (see Section 2.3), the same multi-marker tracking algorithm can be used for flying robot control. An example is shown in Figure 6.2.

6.1.3 Sensor fusion using Kalman filter

The update rate of the distance measurement obtained from the marker tracking algorithm on the ground station computer is limited by the camera frequency. Usually, the on-board state estimation of the flying robot is significantly faster. To increase the rate of the marker estimate, all available sensor measurements can be combined, as shown in Figure 6.3, using a Kalman filter [196]. This also attenuates noise and provides a prediction of the marker pose. In general, the pose estimation problem is nonlinear, hence, an Extended Kalman Filter (EKF) [196] is used. In general, the EKF uses a process model and a measurement model linearized at the current estimate, sensor and process noise, as well as sensor measurements to update the pose estimate. The filter algorithm adopted from ([196], p. 51) is given by

$$\text{Prediction:} \tag{6.1}$$

$$\overline{\boldsymbol{x}}_k = \boldsymbol{g}(\hat{\boldsymbol{x}}_{k-1}) \tag{6.2}$$

$$\overline{\boldsymbol{P}}_k = \boldsymbol{G}_k \boldsymbol{P}_{k-1} \boldsymbol{G}_k^T + \boldsymbol{X}_k \tag{6.3}$$

$$\text{Correction:} \tag{6.4}$$

$$\boldsymbol{K}_k = \overline{\boldsymbol{P}}_k \boldsymbol{H}_k^T (\boldsymbol{H}_k \overline{\boldsymbol{P}}_k \boldsymbol{H}_k^T + \boldsymbol{Z}_k)^{-1} \tag{6.5}$$

$$\hat{\boldsymbol{x}}_k = \overline{\boldsymbol{x}}_k + \boldsymbol{K}_k (\boldsymbol{z}_k - \boldsymbol{h}(\overline{\boldsymbol{x}}_k)) \tag{6.6}$$

$$\boldsymbol{P}_k = (\boldsymbol{E} - \boldsymbol{K}_k \boldsymbol{H}_k) \overline{\boldsymbol{P}}_k \tag{6.7}$$

where k is the current time step, $\hat{\boldsymbol{x}}_k$ is the estimate, $\boldsymbol{P}_k$ is the estimate covariance matrix, and $\boldsymbol{G}_k = \frac{\partial \boldsymbol{g}(\hat{\boldsymbol{x}}_{k-1})}{\partial \hat{\boldsymbol{x}}_{k-1}}$ and $\boldsymbol{H}_k = \frac{\partial \boldsymbol{h}(\overline{\boldsymbol{x}}_k)}{\partial \overline{\boldsymbol{x}}_k}$. The $\overline{(\cdot)}$ notation denotes values before the correction step.

Furthermore, $\boldsymbol{K}_k$ is the Kalman gain, $\boldsymbol{z}_k$ is the measurement vector, $\boldsymbol{X}_k$ is the covariance of the process noise, $\boldsymbol{Z}_k$ is the covariance of the measurement noise, and $\boldsymbol{E}$ is an identity matrix with appropriate size. If $\boldsymbol{X}$ is large, the Kalman filter tracks large changes in the data more closely and vice versa. And for smaller $\boldsymbol{Z}$, the estimated state will follow the measurement more closely and vice versa.

A Kalman filter allows to incorporate measurements at different rates. However, delayed measurements can significantly deteriorate the estimate. Hence, the update strategy illustrated in Figure 6.4 is implemented. It rejects measurements whose delay is above a threshold. Here, measurements with time stamp smaller than the previous filter update are rejected. The varying time delay of the wireless LAN communication channel is computed from the difference between computer clock and message time stamps. This requires clock synchronization of ground station computer and on-board flight control computer.

The Precision Time Protocol (PTP) defined in the standards IEEE 1588 and IEC 61588 [197] is used to synchronize the clocks. The PTP algorithm is illustrated in Figure 6.5. Figure 6.6 shows round trip delay (RTD) measurements recorded for the custom-built quadrocopter Sparrow and the off-the-shelf AR.Drone 2.0 (see Figure 2.2b in Section 2.1). The observed RTD, which is most of the time about 4 ms for Sparrow and about 1 ms for the AR.Drone, shows that the IMU measurements can be used in the Kalman filter. The occurrence of very delayed messages confirms that the proposed update strategy depicted in Figure 6.4 is needed to handle outliers. Note that the Kalman filter also rejects marker pose measurements which arrive too late. Therefore, the estimate is not corrupted, if the tags are not detected.

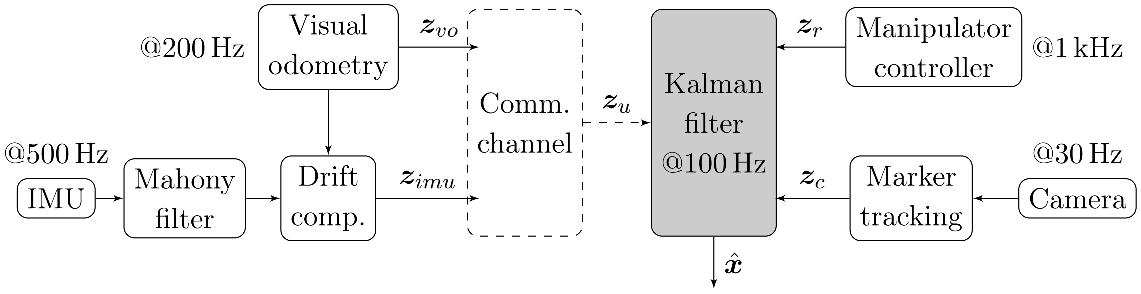


Figure 6.3: Overview of the sensor fusion implementation which provides a prediction of the marker pose.

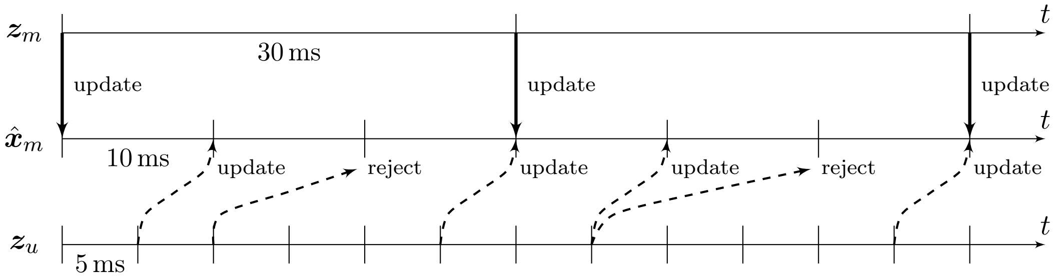


Figure 6.4: Kalman filter update strategy. Measurements of the Visual Inertial Odometry (VIO) of the flying robot that arrive too late due to varying time delay of the communication channel are rejected.

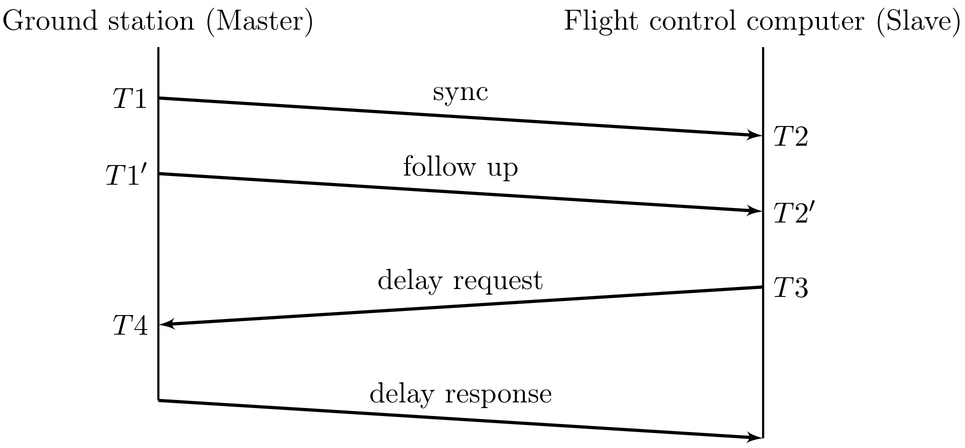


Figure 6.5: Illustration of the Precision Time Protocol (PTP). It uses a hierarchical master-slave architecture. The masters selects the most accurate clock as reference clock. To determine the delay, the master sends its time stamp $T1$ to the slave. Additionally, $T1'$ is measured at the ethernet interface to eliminate the delay in the pipeline from stack to hardware. The slave records the time at which the messages are received using its own clock as $T2$ and $T2'$, respectively. Then, it repeatedly sends delay request messages with time stamps $T3$ to the master. The master responds with the time stamp $T4$ at which the messages are received. The master-to-slave and slave-to-master delays are determined from the differences between the time stamps. The mean value provides the time deviation from the master, which the slave uses for synchronization.

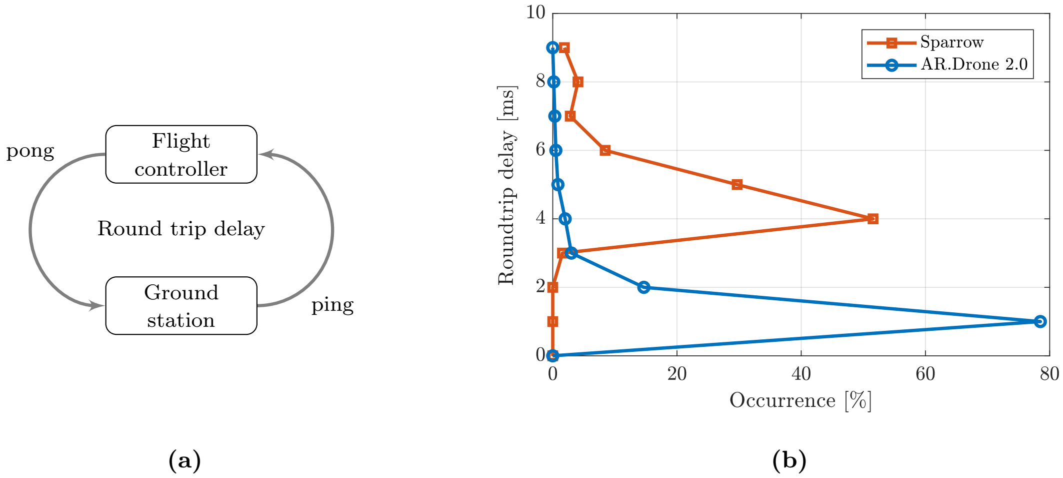


Figure 6.6: Round trip delay (RTD) measurements for the quadrocopters Sparrow and AR.Drone 2.0 acquired over ten minutes using ping command. RTD for AR.Drone is 1 ms in 80 percent of the time. RTD for Sparrow, which is based on a Raspberry Pi 3B+, is 4 ms in 50 percent of the time.

An extended Kalman filter for estimating the marker pose that incorporates on-board pose estimates of the flying robot can be designed as follows. A pose may be described by a translation $\boldsymbol{r}$ and a quaternion $\boldsymbol{q}$ which represents the orientation. Using quaternions in the Kalman filter is inspired by [198]. The advantage of quaternions is that they have no singularities and that their derivative w.r.t. time can be written very concisely (see Section 2.4.1). Without loss of generality, some important assumptions are made, as they

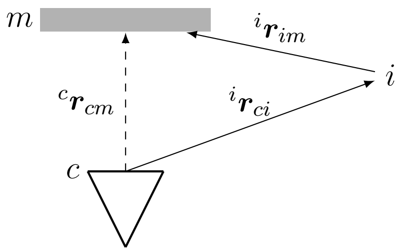

m
${}^{i}\boldsymbol{r}_{im}$
i
${}^{c}\boldsymbol{r}_{cm}$
${}^{i}\boldsymbol{r}_{ci}$
c

Figure 6.7: Considered reference frames and translations.

significantly reduce the size of the process model. The dynamic transformation from camera frame c to world frame w of the robot manipulator are performed a priori on the manipulator side. The marker pose in the camera frame c can always be transformed to the world frame w using the static extrinsic camera calibration, the forward kinematics $\boldsymbol{\chi}(\boldsymbol{\phi})$ of the manipulator (available at 1 kHz), and the estimated pose of the moving base. Hence, it is assumed that w coincides with c. The constant transformation from body-fixed frame u of the flying robot to the marker frame m is also applied a priori, i.e. the flying robot publishes pose, velocity, and acceleration of the marker expressed in the world frame i. Note that the world frame w of robot manipulator and the world frame i of the flying robot do not coincide in general. Summing up, the frames c, m, and i, as shown in Figure 6.7, and the static transformation between c and i are considered in the Kalman filter. Considering a varying transformation between c and i might be more general, but would considerably increase the size of the state vector as well as the complexity of the presentation here and of the final implementation.

The state vector $\hat{\boldsymbol{x}}$ of the Kalman filter is defined as

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{\boldsymbol{x}}_r^T & \hat{\boldsymbol{x}}_q^T \end{bmatrix}^T \tag{6.8}$$

where

$$\hat{\boldsymbol{x}}_r = \begin{pmatrix} {}^{c}\boldsymbol{r}_{cm}^T & {}^{c}\boldsymbol{v}_{cm}^T & {}^{c}\boldsymbol{a}_{cm}^T & {}^{i}\boldsymbol{r}_{im}^T & {}^{i}\boldsymbol{v}_{im}^T & {}^{i}\boldsymbol{a}_{im}^T & {}^{i}\boldsymbol{r}_{ci}^T \end{pmatrix}^T, \tag{6.9}$$

with linear velocities $\boldsymbol{v}$ and linear accelerations $\boldsymbol{a}$ and

$$\hat{\boldsymbol{x}}_q = \begin{pmatrix} \boldsymbol{q}_{cm}^T & \dot{\boldsymbol{q}}_{cm}^T & \ddot{\boldsymbol{q}}_{cm}^T & \boldsymbol{q}_{im}^T & \dot{\boldsymbol{q}}_{im}^T & \ddot{\boldsymbol{q}}_{im}^T & \boldsymbol{q}_{ci}^T \end{pmatrix}^T. \tag{6.10}$$

For better readability, the hat notation $\hat{(\cdot)}$ is omitted for the elements in $\hat{\boldsymbol{x}}$ and the discrete process model is written independently for the translational and for the rotational part as

$$\hat{\boldsymbol{x}}_{r,k} = \boldsymbol{g}_r(\hat{\boldsymbol{x}}_{k-1}) = \boldsymbol{A}_{r,k-1}(\hat{\boldsymbol{x}}_{k-1})\hat{\boldsymbol{x}}_{r,k-1}, \tag{6.11}$$

$$\hat{\boldsymbol{x}}_{q,k} = \boldsymbol{g}_q(\hat{\boldsymbol{x}}_{k-1}) = \boldsymbol{A}_{q,k-1}(\hat{\boldsymbol{x}}_{k-1})\hat{\boldsymbol{x}}_{q,k-1}, \tag{6.12}$$

respectively. Note that (6.11) and (6.12) are nonlinear since $\boldsymbol{A}_{r,k-1}$ and $\boldsymbol{A}_{q,k-1}$ depend on the states $\boldsymbol{q}_{ci,k-1}$ and $\boldsymbol{q}_{im,k-1}$. If the sample time Δt is small enough, accelerations can be assumed constant in the interval $[(k-1)\Delta t, k\Delta t]$. For convenience, transformation of the translations is carried out using rotation matrices, which can be constructed from quaternions using the Euler-Rodrigues formula (2.20) from Section 2.4.1. The marker tracking algorithm from Section 6.1.2 provides the quaternion $\boldsymbol{q}_{cm}$ and the state estimation of the flying robot is assumed to provide the quaternion $\boldsymbol{q}_{im}$ and its first and second derivatives w.r.t. time. Recall that the quaternion multiplication is defined using the quaternion matrix $\boldsymbol{Q}(\boldsymbol{q})$ (2.24) and that the quaternion kinematics are given by (2.25) (see both in Section 2.4.1). Thus, the process model follows as

$$\boldsymbol{A}_{r,k-1} = \begin{bmatrix} \boldsymbol{E} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \Delta t\boldsymbol{R}(\boldsymbol{q}_{ci,k-1}) & \frac{1}{2}\Delta t^2\boldsymbol{R}(\boldsymbol{q}_{ci,k-1}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{R}(\boldsymbol{q}_{ci,k-1}) & \Delta t\boldsymbol{R}(\boldsymbol{q}_{ci,k-1}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{R}(\boldsymbol{q}_{ci,k-1}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{E} & \Delta t\boldsymbol{E} & \frac{1}{2}\Delta t^2\boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{E} & \Delta t\boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{R}(\boldsymbol{q}_{ci,k-1}^{-1}) & \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{E} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{6.13}$$

and

$$\boldsymbol{A}_{q,k-1} = \begin{bmatrix} \boldsymbol{E} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \Delta t\boldsymbol{Q}(\boldsymbol{q}_{ci,k-1}) & \frac{1}{2}\Delta t\boldsymbol{Q}(\boldsymbol{q}_{ci,k-1}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Q}(\boldsymbol{q}_{ci,k-1}) & \Delta t\boldsymbol{Q}(\boldsymbol{q}_{ci,k-1}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Q}(\boldsymbol{q}_{ci,k-1}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{E} & \Delta t\boldsymbol{E} & \frac{1}{2}\Delta t^2\boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{E} & \Delta t\boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{U}(\boldsymbol{q}_{im,k-1}^{-1}) & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \tag{6.14}$$

where $\boldsymbol{E}$ is an identity matrix. In the prediction step, (6.13) and (6.14) are evaluated using the estimates $\boldsymbol{q}_{ci,k-1}$ and $\boldsymbol{q}_{im,k-1}$ at time step $k-1$. To predict the covariance, the Jacobian $\boldsymbol{G}_k$ is required ([196], p. 50), which is computed using $\boldsymbol{A}_{k-1} = \mathrm{diag}(\boldsymbol{A}_{r,k-1}, \boldsymbol{A}_{q,k-1})$ as

$$\boldsymbol{G}_k = \frac{\partial(\boldsymbol{A}_{k-1}\hat{\boldsymbol{x}}_{k-1})}{\partial\hat{\boldsymbol{x}}_{k-1}} = \tag{6.15}$$

$$\left[\begin{array}{c|ccccc} & & & & & \frac{\partial(\boldsymbol{R}(\boldsymbol{q}_{ci,k-1})({}^i\boldsymbol{v}_{im,k-1}\Delta t + {}^i\boldsymbol{a}_{im,k-1}\frac{1}{2}\Delta t^2))}{\partial\boldsymbol{q}_{ci,k-1}} \\ & & & & & \frac{\partial(\boldsymbol{R}(\boldsymbol{q}_{ci,k-1})({}^i\boldsymbol{v}_{im,k-1} + {}^i\boldsymbol{a}_{im,k-1}\Delta t))}{\partial\boldsymbol{q}_{ci,k-1}} \\ \boldsymbol{A}_{r,k-1} & & & \boldsymbol{0}_{21\times24} & & \frac{\partial(\boldsymbol{R}(\boldsymbol{q}_{ci,k-1})\,{}^i\boldsymbol{a}_{im,k-1})}{\partial\boldsymbol{q}_{ci,k-1}} \\ & & & & & \boldsymbol{0}_{9\times3} \\ & & & & & \frac{\partial(\boldsymbol{R}(\boldsymbol{q}_{ci,k-1}^{-1})\,{}^c\boldsymbol{r}_{cm,k-1})}{\partial\boldsymbol{q}_{ci,k-1}} \\ \hline & & & \text{Rows 1-24,} & & \boldsymbol{U}(\dot{q}_{im,k-1})\Delta t + \boldsymbol{U}(\ddot{q}_{im,k-1})\frac{1}{2}\Delta t^2 \\ & & & \text{columns 1-24} & & \boldsymbol{U}(\dot{q}_{im,k-1}) + \boldsymbol{U}(\ddot{q}_{im,k-1})\Delta t \\ \boldsymbol{0}_{28\times21} & & & \text{of } \boldsymbol{A}_{q,k-1} & & \boldsymbol{U}(\ddot{q}_{im,k-1}) \\ & & & & & \boldsymbol{0}_{12\times4} \\ & \boldsymbol{U}(\boldsymbol{q}_{im,k-1}^{-1}) & \boldsymbol{0}_{4\times8} & \frac{\partial(\boldsymbol{U}(\boldsymbol{q}_{im,k-1}^{-1})\boldsymbol{q}_{cm,k-1})}{\partial\boldsymbol{q}_{im,k-1}} & \boldsymbol{0}_{4\times8} & \boldsymbol{0}_{4\times4} \end{array}\right] \tag{6.16}$$

The following identities are needed to derive the matrix (6.16). For the partial derivatives of the rotation matrices, it holds that

$$\frac{\partial(\boldsymbol{R}(\boldsymbol{q})\boldsymbol{r})}{\partial\boldsymbol{r}} = \boldsymbol{R}(\boldsymbol{q}),$$

$$\frac{\partial(\boldsymbol{R}(\boldsymbol{q})\boldsymbol{r})}{\partial\boldsymbol{q}} =$$

$$\begin{bmatrix} 2q_w r_x + 2q_y r_z - 2q_z r_y & 2q_x r_x + 2q_y r_y + 2q_z r_z & 2q_w r_z + 2q_x r_y - 2q_y r_x & 2q_x r_z - 2q_w r_y - 2q_z r_x \\ 2q_w r_y - 2q_x r_z + 2q_z r_x & 2q_y r_x - 2q_x r_y - 2q_w r_z & 2q_x r_x + 2q_y r_y + 2q_z r_z & 2q_w r_x + 2q_y r_z - 2q_z r_y \\ 2q_w r_z + 2q_x r_y - 2q_y r_x & 2q_w r_y - 2q_x r_z + 2q_z r_x & 2q_z r_y - 2q_y r_z - 2q_w r_x & 2q_x r_x + 2q_y r_y + 2q_z r_z \end{bmatrix}.$$

And for the partial derivatives of the quaternion multiplication it holds that

$$\frac{\partial(\boldsymbol{Q}(\boldsymbol{q}_1)\boldsymbol{q}_2)}{\partial\boldsymbol{q}_2} = \boldsymbol{Q}(\boldsymbol{q}_1), \tag{6.17}$$

$$\frac{\partial(\boldsymbol{Q}(\boldsymbol{q}_1)\boldsymbol{q}_2)}{\partial\boldsymbol{q}_1} = \frac{\partial(\boldsymbol{U}(\boldsymbol{q}_2)\boldsymbol{q}_1)}{\partial\boldsymbol{q}_1} = \boldsymbol{U}(\boldsymbol{q}_2), \tag{6.18}$$

$$\frac{\partial(\boldsymbol{U}(\boldsymbol{q}_1^{-1})\boldsymbol{q}_2)}{\partial\boldsymbol{q}_1} = \frac{\partial(\boldsymbol{Q}(\boldsymbol{q}_2)\boldsymbol{q}_1^{-1})}{\partial\boldsymbol{q}_1} = \boldsymbol{W}(\boldsymbol{q}_2), \tag{6.19}$$

where $\boldsymbol{U}(\boldsymbol{q})$ is the conjugate quaternion matrix defined in (2.24) and

$$\boldsymbol{W}(\boldsymbol{q}) = \begin{bmatrix} \eta & \boldsymbol{\epsilon}^T \\ \boldsymbol{\epsilon} & -\eta\boldsymbol{E} - \boldsymbol{S}(\boldsymbol{\epsilon}) \end{bmatrix}. \tag{6.20}$$

Since the visual inertial odometry of the flying robot and the marker pose obtained from the multi-marker tracking algorithm have different update rates, they are incorporated in the Kalman filter using two different measurement models. Fortunately, the models are linear, such that line (6.6) of the EKF becomes

$$\hat{\boldsymbol{x}}_k = \overline{\boldsymbol{x}}_k + \boldsymbol{K}_k(\boldsymbol{z}_{u,k} - \boldsymbol{H}_u\overline{\boldsymbol{x}}_k), \tag{6.21}$$

$$\hat{\boldsymbol{x}}_k = \overline{\boldsymbol{x}}_k + \boldsymbol{K}_k(\boldsymbol{z}_{c,k} - \boldsymbol{H}_c\overline{\boldsymbol{x}}_k), \tag{6.22}$$

for the odometry and camera measurements, respectively, where

$$\boldsymbol{H}_u = \begin{bmatrix} \boldsymbol{0}_{9\times 9} & \boldsymbol{E}_{9\times 9} & \boldsymbol{0}_{9\times 31} \\ \boldsymbol{0}_{12\times 33} & \boldsymbol{E}_{12\times 12} & \boldsymbol{0}_{12\times 4} \end{bmatrix},$$

$$\boldsymbol{H}_c = \begin{bmatrix} \boldsymbol{E}_{3\times 3} & \multicolumn{2}{c}{\boldsymbol{0}_{3\times 46}} \\ \boldsymbol{0}_{4\times 21} & \boldsymbol{E}_{4\times 4} & \boldsymbol{0}_{4\times 24} \end{bmatrix}.$$

Using the sensor fusion approach as described above, an estimation of the pose of the flying robot relative to the end-effector of the manipulator is obtained as visualized in Figure 6.8. Note that once the static transformation between frame c and frame i is estimated, the pose of the marker can be estimated using this transformation and the odometry of the flying robot. That means that the estimate is available even if the markers are occluded or outside of the field of view of the camera.

6.2 Visual servo control of the robot manipulator

Visual servo control uses computer vision for motion control of a robot [199, 200, 201]. In this section, a straightforward approach for visual servo control of the robot manipulator is proposed. The pose estimate obtained from the Kalman filter is directly used in the Cartesian impedance controller (2.87) and (4.100) presented in Section 2.6 and Section 4.4.2, respectively. But first, the maximum attainable end-effector velocity of the DLR LWR is assessed numerically.

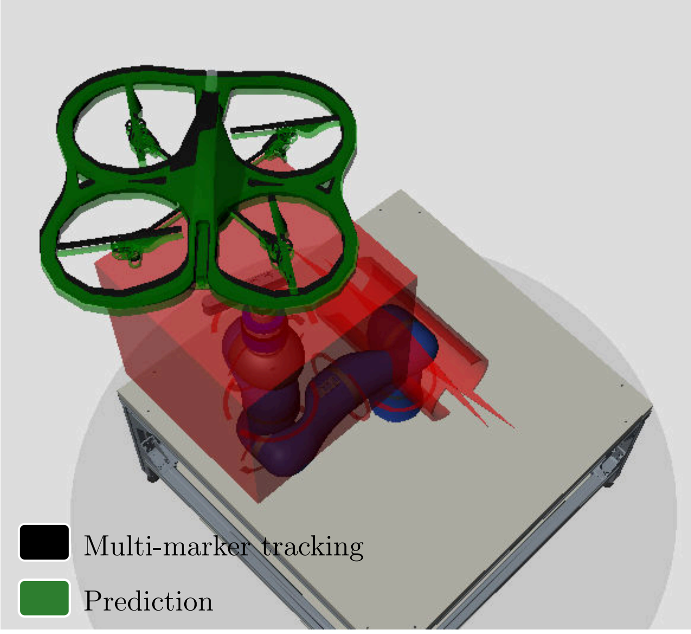


Figure 6.8: Visualization of the estimated pose of the flying robot obtained from multi-marker tracking (black) and predicted (green) using a Kalman filter approach for sensor fusion.

6.2.1 Assessment of the maximum end-effector velocity

In order to assess in which velocity region the robot manipulator is able to compensate the base motion and catch the flying robot, a rough estimation based on the maximum joint speeds $\dot{\boldsymbol{\phi}}_{\max}$ is presented. A manipulator's Cartesian velocity capabilities for a given configuration can be analyzed using so-called manipulability ellipsoids [202, 203] and polytopes [204, 205]. Since the translational motion is dominant for the capturing task, only the maximum linear velocities $\boldsymbol{v}$ are considered here. One can distinguish between an analysis in the weak sense, where $|\dot{\boldsymbol{\phi}}| \leq \dot{\boldsymbol{\phi}}_{\max}$, and in the strong sense, where additionally the angular velocity of the end-effector is constrained to be zero. In order to obtain a conservative estimation, an analysis in the strong sense is performed.

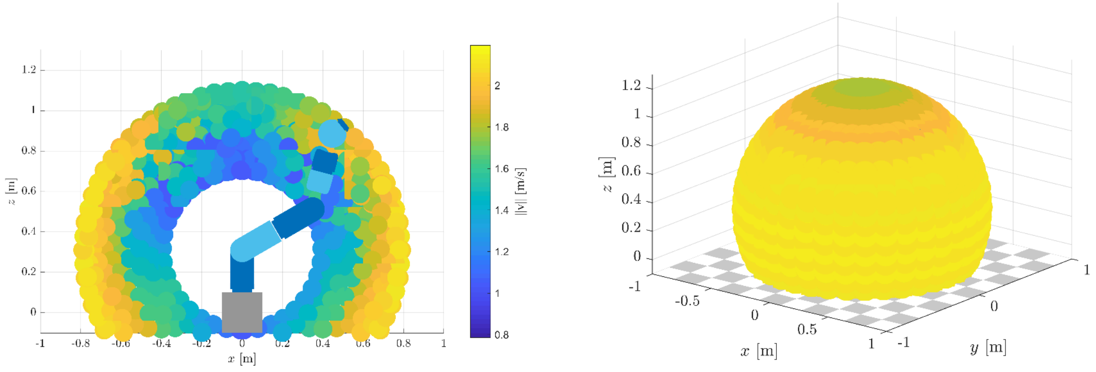
Instead of manipulability ellipsoids and polytopes, which are computationally demanding, a trivial solution is presented. The joint space is sampled using steps of 10 deg and the optimization problem defined below is solved numerically using the MATLAB® implementation of qpOASES [156].

$$\min_{\dot{\boldsymbol{\phi}}} \quad -\boldsymbol{v}^T\boldsymbol{v} = \dot{\boldsymbol{\phi}}^T\boldsymbol{J}^T\boldsymbol{J}\dot{\boldsymbol{\phi}} \quad \text{s.t.} \begin{cases} -\dot{\boldsymbol{\phi}}_{\max} \leq \dot{\boldsymbol{\phi}} \leq \dot{\boldsymbol{\phi}}_{\max}, \\ \boldsymbol{J}_\omega\dot{\boldsymbol{\phi}} = \boldsymbol{0}. \end{cases} \tag{6.23}$$

Note that minimizing $-\boldsymbol{v}^T\boldsymbol{v}$ is equal to maximizing $\boldsymbol{v}^T\boldsymbol{v}$. It holds for the Jacobian that

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_v \\ \boldsymbol{J}_\omega \end{bmatrix}$$

and the Cartesian end-effector position is obtained via the forward kinematics $\boldsymbol{\chi}(\boldsymbol{\phi})$. Similar to manipulability ellipsoids and polytopes, the dynamics of the manipulator are neglected,



(a) x-z-plane. **(b)** Side view.

Figure 6.9: Maximum attainable velocity of the manipulator LWR 4+.

i.e. it is assumed that the manipulator can reach its maximum joint velocities instantaneously. The results for the LWR 4+, whose maximum joint velocities are given in Table 2.1 in Section 2.2, are shown in Figure 6.9. Note that the colors correspond to the Euclidean norm of the linear velocity $||\boldsymbol{v}||$. Theoretically, the LWR 4+ could reach velocities up to $2\,\mathrm{m/s}$. Obviously, it is not able to reach the maximum velocity in the outstretched configuration. It is reasonable to conclude a maximum velocity of approximately $1\,\mathrm{m/s}$ in the entire workspace.

6.2.2 Cartesian impedance visual servo control

For tracking the motion of the flying robot by means of the robot's end-effector and finally capturing the aerial vehicle, the Cartesian impedance controller (2.87) is utilized. This allows to realize compliant contacts and avoid switching between controller modes. A straight-forward approach to visual servo control is chosen. The relative distance measurement between c and m is interpreted as a desired velocity and directly incorporated in the controller. Velocity limits and workspace boundaries of the robot manipulator are considered using a saturation function. Another possibility is to use virtual walls [206], i.e. virtual spring-damper-systems, which generate a repelling force at the end-effector.

It is clear, that the relative distance between the camera c mounted at the end-effector and the marker m can be transformed to a desired pose $\boldsymbol{x}_d$ in the world frame of the manipulator using the forward kinematics $\boldsymbol{\chi}(\boldsymbol{\phi})$. Next, recall the Cartesian impedance controller (2.87)

$$\boldsymbol{\tau}_d = \boldsymbol{g}(\boldsymbol{\phi}) - \boldsymbol{J}(\boldsymbol{\phi})^T \left(\boldsymbol{K}_d \boldsymbol{x}_e + \boldsymbol{D}_d \dot{\boldsymbol{x}}_e \right) + \boldsymbol{\tau}_{nsp} \tag{6.24}$$

with the position error defined as $\boldsymbol{x}_e = \boldsymbol{x} - \boldsymbol{x}_d$ and using the Jacobian $\boldsymbol{J}(\boldsymbol{\phi})$ of the manipulator. Furthermore, $\boldsymbol{K}_d$, and $\boldsymbol{D}_d$ are the desired Cartesian stiffness and the desired Cartesian damping, with $\boldsymbol{K}_d = \mathrm{diag}(K_1, \ldots, K_6)$ and $\boldsymbol{D}_d = \mathrm{diag}(D_1, \ldots, D_6)$, respectively. Joint torques within the nullspace of the manipulator, i.e. which do not influence the end-effector motion but the robot's posture, are comprised in $\boldsymbol{\tau}_{nsp}$. The motor dynamics of the manipulator are neglected, i.e. it is assumed that the joint torques $\boldsymbol{\tau}_j$ are controlled directly.

The low level torque controller of the manipulator assures that $\boldsymbol{\tau}_j$ follows the desired joint torque $\boldsymbol{\tau}_d$.

Virtual workspace boundaries

In order to account for workspace boundaries and to avoid configuration singularities, a limit on the admissible distance measurement, denoted as $\boldsymbol{d}$, is added. The desired cubic workspace in the base frame b is defined by center ${}^{b}\boldsymbol{c}$ and length $\boldsymbol{l} = (l_x \quad l_y \quad l_z)^T$, as depicted in Figure 6.10. Then the limit is defined as

$$\boldsymbol{d} = \operatorname{sat}\left({}^{b}\boldsymbol{r}_{bm}, {}^{b}\boldsymbol{c} - \frac{\boldsymbol{l}}{2}, {}^{b}\boldsymbol{c} + \frac{\boldsymbol{l}}{2}\right) - \boldsymbol{\chi}(\boldsymbol{\phi}), \tag{6.25}$$

with

$${}^{b}\boldsymbol{r}_{bm} = \boldsymbol{\chi}(\boldsymbol{\phi}) + \boldsymbol{R}_{bc}\,{}^{c}\boldsymbol{r}_{cm} \tag{6.26}$$

and using a saturation function defined as

$$\operatorname{sat}(a, \underline{a}, \overline{a}) = \begin{cases} \underline{a}, & a \leq \underline{a} \\ a, & \underline{a} < a < \overline{a} \\ \overline{a}, & a \geq \overline{a} \end{cases}. \tag{6.27}$$

Therein, a is the value to saturate, $\underline{a}$ is its lower and $\overline{a}$ its upper limit. Note that in (6.25), each element of the vector argument is saturated independently. The augmented distance $\boldsymbol{d}$ is interpreted as a desired velocity of the end-effector

$$\dot{\boldsymbol{x}}_d = \operatorname{sat}\left(k_\delta \cdot \boldsymbol{d}, \dot{\boldsymbol{x}}_{\min}, \dot{\boldsymbol{x}}_{\max}\right), \tag{6.28}$$

wherein $k_\delta > 0$ is a scalar gain. It follows from (6.28), that

$$\boldsymbol{x}_d = \boldsymbol{x}_d(t_0) + \int_{t_0}^{t_\infty} \operatorname{sat}\left(k_\delta \cdot \boldsymbol{d}, \dot{\boldsymbol{x}}_{\min}, \dot{\boldsymbol{x}}_{\max}\right) \mathrm{d}t, \tag{6.29}$$

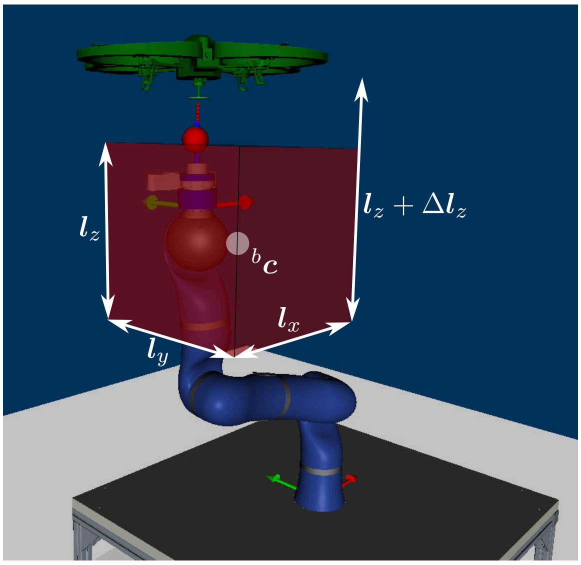


Figure 6.10: Exemplary virtual workspace limits for a cubic robot arm workspace with center ${}^{b}\boldsymbol{c}$. It may be implemented using a saturation function for the end-effector velocity or via a virtual wall.

wherein t_0 is the start and t_∞ is the end time. (6.28) and (6.29) are inserted in (6.24) and enable to account for the minimum and maximum velocities $\dot{\boldsymbol{x}}_{\min}$ and $\dot{\boldsymbol{x}}_{\max}$ of the manipulators end-effector.

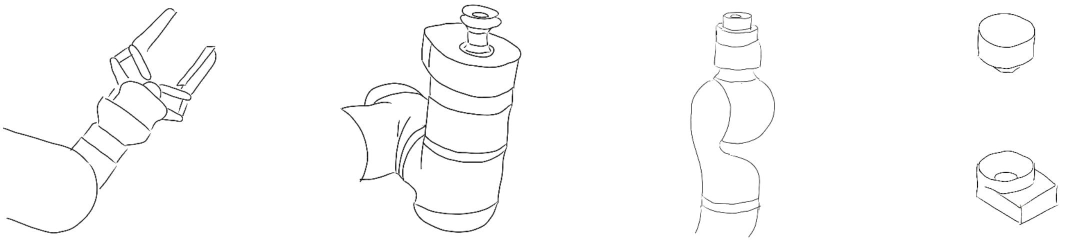
The transition from tracking to capturing is realized by increasing the workspace limits $\boldsymbol{l}$ until the docking point lies within the workspace boundaries. This can be done either with a step of $\Delta\boldsymbol{l}$, as indicated in Figure 6.10, or using linear interpolation from $\boldsymbol{l}$ to $\boldsymbol{l}+\Delta\boldsymbol{l}$ in order to slow down the transition. The start of the transition can be triggered autonomously, e.g. as soon as the flying robot reaches a defined rendezvous and stays there for a defined duration, or manually by an operator.

6.3 Discussion of docking interfaces

In this section, different docking interfaces are discussed in order to select a suitable one for the experiments. The considered designs are depicted in Figure 6.11.

A gripper (Figure 6.11 (a)) is a very general tool and usually available directly. No special interface is required on the UAV, except for an area where it can be gripped. However, a gripper is not necessarily reliable or robust and the area between the gripper jaws is usually small.

A (Figure 6.11 (b)) suction cup is more specialized and usually more reliable than a common gripper. However, it requires compressed air and a corresponding flat counterpart on the UAV to provide a firm grip.



(a) Gripper **(b)** Suction cup **(c)** Electromagnet **(d)** Cylindrical peg-in-hole

Figure 6.11: Illustration of different mechanical docking interfaces.

Electromagnets (Figure 6.11 (c)) are widely used and directly available. They require direct current (usually 12 V or 24 V) and can be switched on and off via a relay. A magnetic counterpart is required on the UAV. They are very robust and reliable, as the magnetic forces can be very strong and also act at some distance (about 0-5 cm).

Due to the mentioned advantages, a 12 V electromagnet is used in this work. A round steel plate with a diameter of 5 cm is attached to the UAV via the custom universal hinge (see Section 4.2.6, 4.3.6, and 5.3.6). A disadvantage of the used magnet is that holding forces remain even after the voltage is switched off and the UAV is difficult to disconnect.

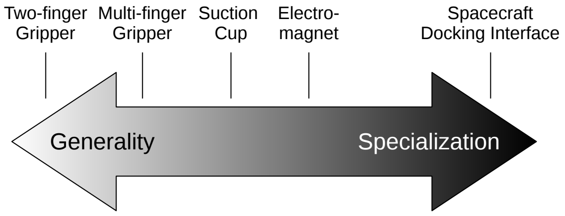


Figure 6.12: Gripper designs: generality vs. specialization.

Table 6.2: Advantages and disadvantages of general or specialized docking interfaces.

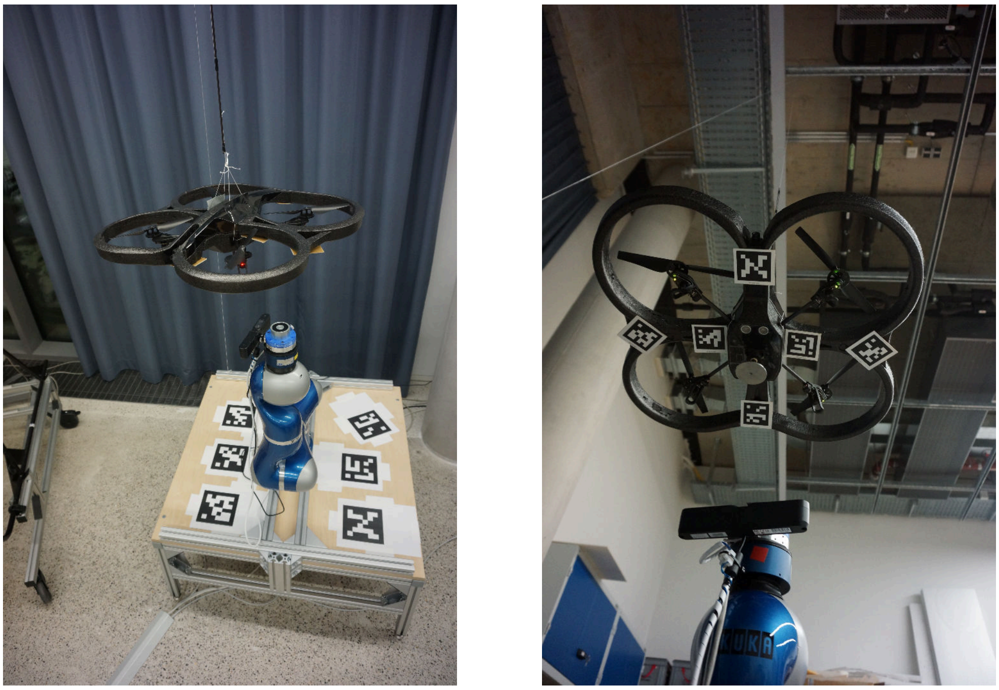
General gripper	Specialized docking interface
⊕ high availability	⊖ low or no availability
⊕ low cost	⊖ high cost
⊖ low accuracy and robustness	⊕ increased accuracy and robustness
⊖ need to consider flying robot design	⊖ increased weight of flying robot

A cylindrical peg-in-hole interface as shown in Figure 6.11 (d) might increase the reliability of the docking maneuver. It could be used in addition to the aforementioned docking interfaces as well. Figure 6.12 and Table 6.2 categorize the interface designs and sum up the advantages and disadvantages.

6.4 Experimental results

In order to evaluate the visual servo control approach, several experiments are conducted using the demonstrator for robot-assisted take-off and landing of flying robots already presented in Section 4.2.6, 4.3.6, and 5.3.6. The manipulator is mounted on a fixed base and has a camera and an electromagnet attached at its end-effector (see Figure 6.13). The camera is the Asus Xtion Pro Live camera with a resolution of 640 × 480 pixels at a rate of 30 frames per second (see Figure 6.13 (b)).

The Parrot AR.Drone 2.0 quadrocopter [171] is controlled via 2.4 GHz wireless LAN. Several markers are attached at the bottom of the quadrocopter as shown in Figure 6.13. Due to safety reasons, the quadrocopter is attached to the lab ceiling with a rope. Its position above a reference is stabilized using a PID controller and the multi-marker tracking algorithm described in Section 6.1.2. Multi-marker tracking is found to be more reliable than single-marker tracking, because the single reference marker is often occluded by the elbow of the manipulator. The orientation of the quadrocopter expressed in Euler angles is depicted in the upper plot in Figure 6.15a. Its heading is varying within ±8 deg, which is irrelevant due to the symmetry of the quadrocopter and because the docking interface is centered below the quadrocopter. The lower plot in Figure 6.15a shows that the quadrocopter is stabilized at a desired position with respect to the reference marker on the ground. All parameters used in the experiments are summarized in Table 6.3. Figure 6.16 and Figure 6.17 show image sequences of a visual tracking experiment and an in-flight capturing experiment, respectively. Furthermore, a successful capturing attempt and the trajectory of quadrocopter and manipulator is illustrated in the picture sequence in 6.14. The physical



(a) **(b)**

Figure 6.13: Setup of the in-flight capturing experiments.

Table 6.3: Parameters used for the in-flight capturing experiments.

Parameter	**Value**
Controller gain k_δ, see (6.28), (6.29)	4.0
Impedance controller stiffness $\boldsymbol{K}_d$ in N/m	$K_{1\ldots3} = 1000$, $K_{4\ldots6} = 100$
Impedance controller damping $\boldsymbol{D}_d$ in in Ns/m	$D_{1\ldots6} = 0.5$
Minimum / maximum linear velocity $\boldsymbol{v}_{\min/\max}$	∓1.0 m/s
Kalman filter time step Δt	0.01 s
Workspace center ${}^b\boldsymbol{c}$	$(-0.4\ 0\ 0.8)^T$ m
Workspace length $\boldsymbol{l}$	$(0.4\ 0.4\ 0.4)^T$ m
Workspace prolongation $\Delta\boldsymbol{l}$	$(0\ 0\ 0.3)^T$ m
Multi-marker tracking rate	30 Hz
Visual inertial odometry rate	200 Hz

connection between both is established using an electromagnet and a metal plate attached to the quadrocopter via a universal hinge. The position of the manipulator's end-effector and its relative distance to the metal plate during the capturing attempt is depicted in Figure 6.15b. It can be seen that the relative distance stays within a range of 5 cm and converges to zero as the electromagnet reaches the metal plate. Capturing is realized with a step of $\Delta l_z = 0.3$ m and initiated as soon as δ_x and δ_y are below 2.0 cm for 2.0 s (i.e. at time $t = 18$ s in Figure 6.15b).

In Section 1.3, the need for coordinated control is motivated by the assumption that the robot controller has to consider the UAV's orientation in order to prevent the UAV from

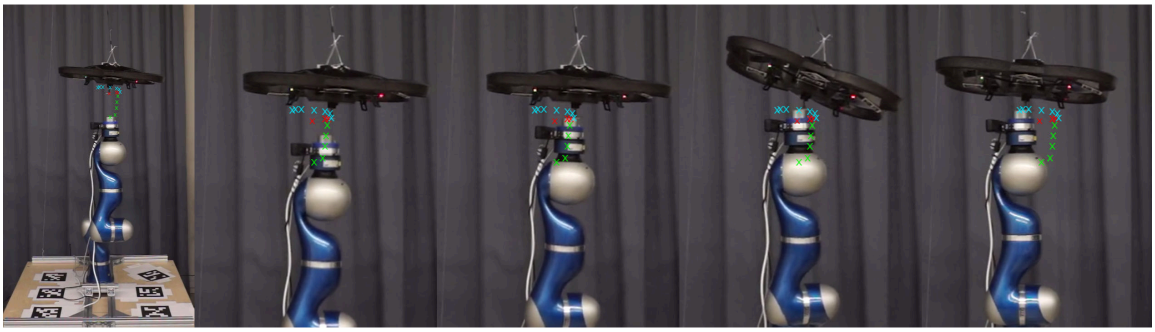
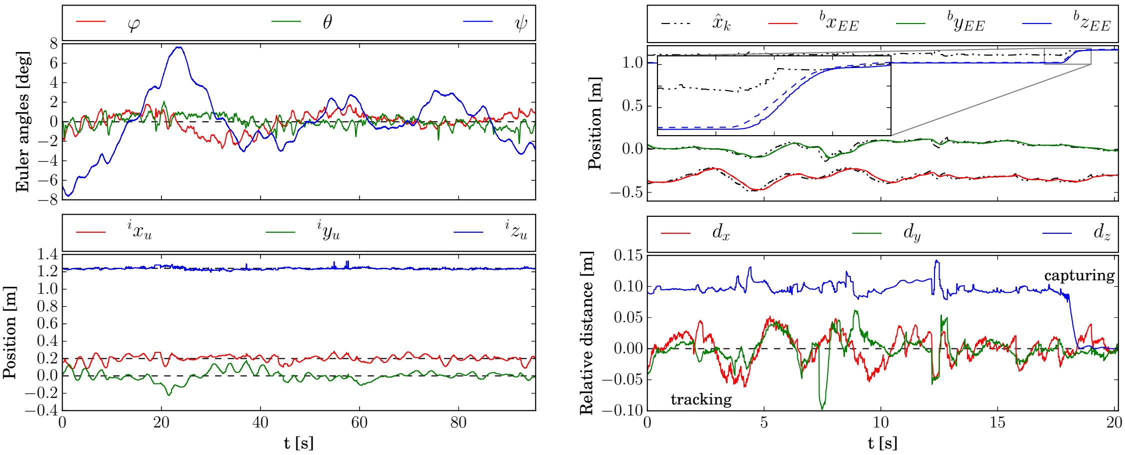


Figure 6.14: Exemplary successful in-flight capturing experiment. The trajectories of the docking of the end-effector, and both together are visualized with red, green, and blue crosses, respectively.



(a) Quadrocopter states during hovering over reference marker, with desired position and orientation (dashed) and estimated position and orientation (solid).

(b) End-effector position of the robot arm during a successful in-flight capturing attempt. The relative distance between end-effector and docking point is shown in the lower plot.

Figure 6.15: Log data plots of successful in-flight capturing experiment.

tilting sideways in an undesired way. The results of the capturing experiments clearly justify this assumption. As can be seen in Figure 6.14, the quadrocopter is significantly tilted sideways in the instance the contact is established and shortly afterwards. The robot is stopped manually after the UAV is captured. Obviously, an additional sensor is needed in order to determine if the capturing attempt is successful or not. A sensor would enable to react correctly to a failed attempt or to switch automatically to the next phase of the landing maneuver.

In summary, the motion of the docking point could be tracked by the manipulator arm with an accuracy below 5 cm. Furthermore, the flying robot could be captured autonomously and reliably. The reliability depends on the docking interface and may be increased by choosing a large, conical, and self-centering mechanism. Different options for the docking interface are discussed in the previous Section 6.3.

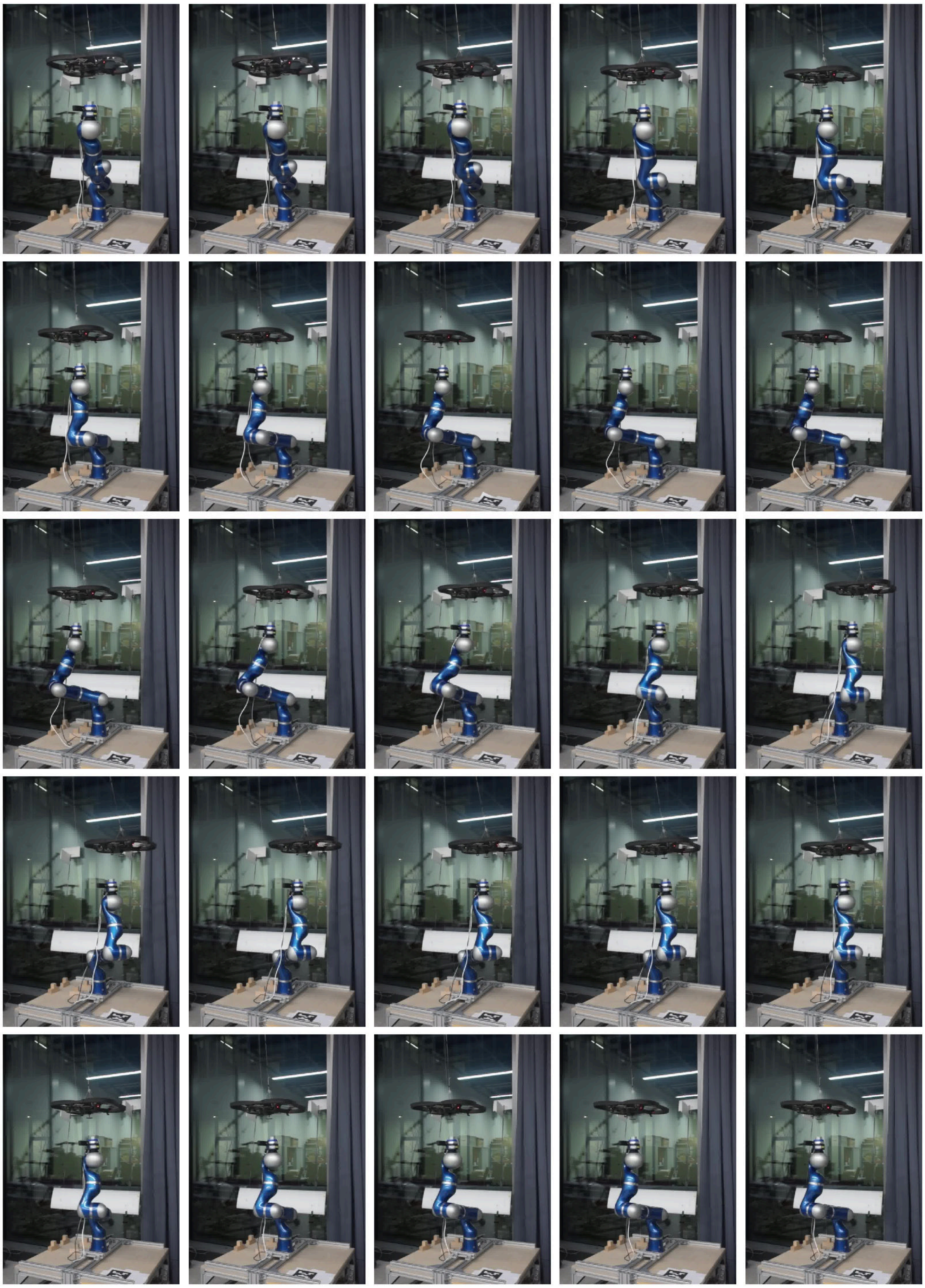


Figure 6.16: Image sequence of a visual tracking experiment. The end-effector of the manipulator constantly follows the quadrocopter. The duration of the depicted motion is 7 s.

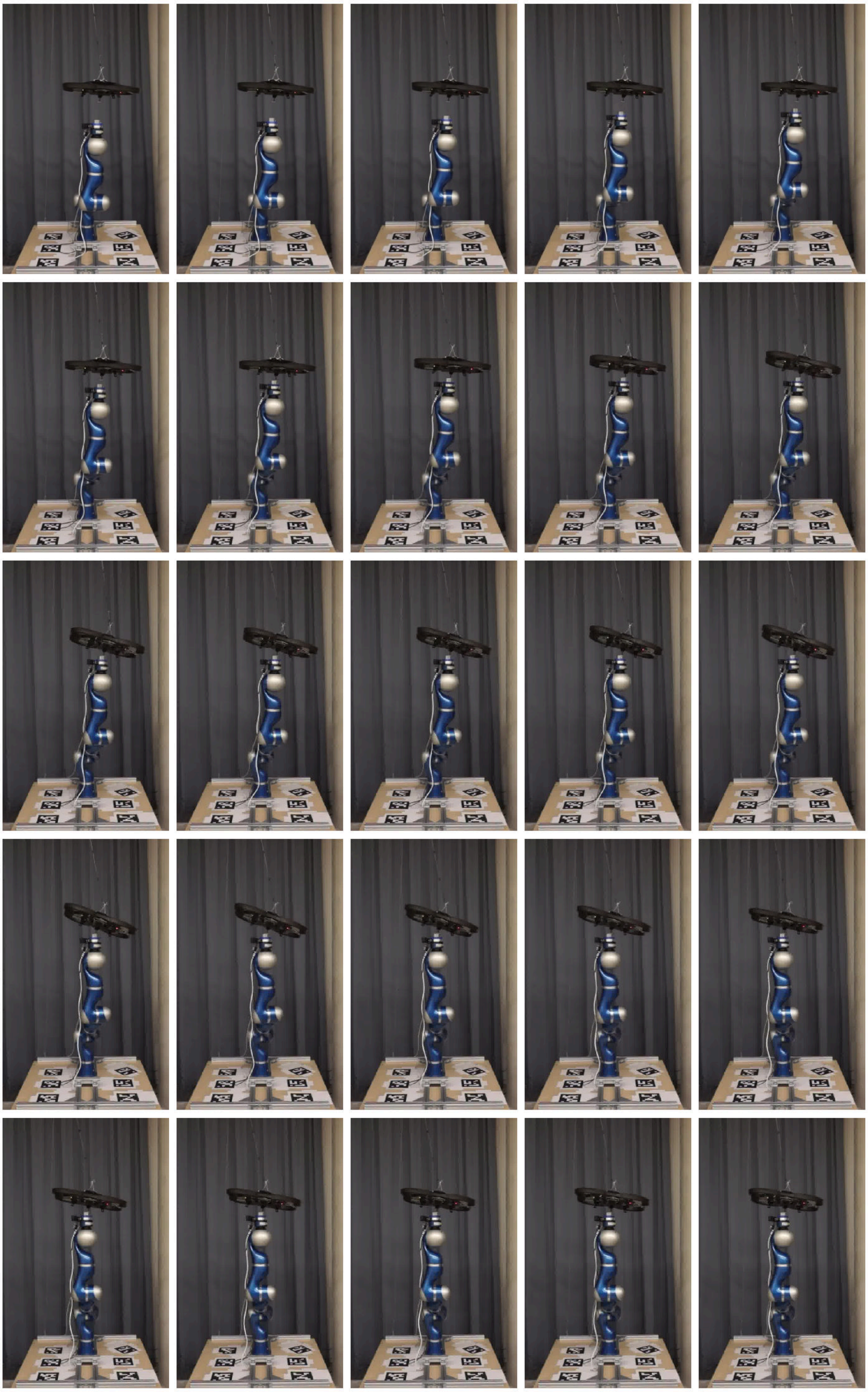


Figure 6.17: Image sequence of an in-flight capturing experiment. The duration of the maneuver is 2 s.

"There is an art to flying, or rather a knack. The knack lies in learning how to throw yourself at the ground and miss."
— Douglas Adams

7 Conclusion

7.1 Summary

In this thesis, coordinated control approaches for robot-assisted take-off and landing of flying robots are developed and tested in simulations and experiments. The evaluation of the presented controllers is summarized in Table 7.1. All coordinated controllers use the fundamental combined model of the flying robot attached to the manipulator via an universal hinge presented in Section 2.4 and the task space formulation introduced in Section 2.4.4.

Chapter 3 is devoted to independent modeling and control of flying robots. A generalized control approach including bidirectional thrust, control allocation procedures considering actuator saturation, and an adaptive control approach accounting for uncertainties of the rotor thrust are presented. This results in robust and accurate trajectory tracking as well as increased precision of an external wrench estimator. Additionally, the system under the adaptive controller is shown to be passive. Indoor and outdoor experiments performed with the hexacopter Ardea and the quadrocopter Sparrow prove the performance of the presented control approach. The experiments with Sparrow include the first ever autonomous transition from upright to inverted flight of a quadrocopter with fixed-pitch propellers.

In Chapter 4, the separate models of flying robot and robot manipulator are used for controller design. As shown, studying the linearized models allows an estimation of the stability limits. Nonlinearities of the attitude dynamics are considered in the backstepping design. However, their effect will only be visible at some distance from the hovering state and at high rotational speeds. The obtained controllers are especially applicable to light UAVs, whose influence on the robot manipulator is negligible. Low complexity of the approaches does not necessarily result in worse performance, both in simulations and experiments (see Figure 7.1). The advantage of the controllers is that they require little state information, such that communication and implementation effort are reduced. The impedance control strategy provides passive and, thus, stable interaction between the subsystems. The task space control approach is extended to account for accurately estimated base motion. It is evaluated successfully in simulation.

Chapter 5 considers the combined model of flying robot and robot manipulator. The presented inertial decomposition allows for workload sharing based on a heuristic (see Section 5.3). If required, the control effort of the manipulator can be reduced, such that it can

Table 7.1: Comparison of control approaches presented in this work.

Section	Description	Assumptions	Complexity	Sim. / Exp.	Performance
Independent control of flying robot					
3.3	PD control	No external disturbances	○ ○ ○	✓/ ✓	● ○ ○
3.5	Adaptive control	Accurate pose estimate	● ○ ○	✓/ ✓	● ● ●
Coordinated control based on separate models					
4.2.3	Task space control without UAV model	Influence of UAV is negligible	● ○ ○	✓/ ✓	● ○ ○
4.2.4	As above with linearized UAV model and pole placement	Light flying robot	● ○ ○	✓/ ✓	● ● ○
4.2.5	As above using modal decoupling	Light flying robot	● ● ●	✓/ ✓	● ● ○
4.3	As above with nonlinear UAV model and backstepping	Light flying robot	● ● ○	✓/ ✓	● ● ○
4.4.2	Base motion compensation	Accurate estimate of base motion	● ● ○	✓/ ✗	● ● ○
4.4.4	As above with active thrust vector control	Accurate estimate of base motion	● ● ●	✓/ ✗	● ● ●
Coordinated control based on combined model					
5.3	Workload sharing using decomposed model	Desired trajectory available	● ● ○	✓/ ✓	● ● ○
5.4.2	Optimal control allocation	Sufficient computing capacity	● ● ●	✓/ ✗	● ● ○
5.4.4	As above with adaptive control	Sufficient computing capacity	● ● ●	✓/ ✗	● ● ●
Independent control of robot manipulator					
6.2.2	Cartesian impedance control	Visual distance measurement	● ○ ○	✗/ ✓	● ● ○

also assist heavy flying robots. The same holds for the optimal control allocation presented in Section 5.4, which is more effective but computationally much more expensive. In short, the control approaches in Chapter 4 neglect the coupling between the subsystems but do not depend so heavily on information exchange with low latency as the control approaches in Chapter 5.

In-flight capturing of the flying robot is considered in Chapter 6. A visual tracking algorithm is presented which uses multiple fiducial markers. The pose estimate as well as the visual inertial odometry is fed into an EKF, which results in an increased update rate and robustness against occlusions of the estimate. The latter combined with a Cartesian impedance controller results in visual servo control of the robot manipulator. It is shown experimentally that a quadrocopter could be captured repeatedly using an electromagnet mounted at the end-effector of the manipulator.

7.2 Outlook

As summarized in Table 7.2, the research questions stated in Section 1.3 could be answered adequately in this thesis. A broad spectrum of coordinated control approaches for robot-assisted take-off and landing of flying robots is provided. Nevertheless, several open points are worth investigating in the future.

First of all, further experiments are required to entirely assess the capabilities of the task space controllers. This includes in-flight capturing of the flying robot as well as cooperative take-off and landing. Especially heavier flying robots have to be considered. In addition, suitable sensors and approaches to accurately estimate the base motion need to be investigated. Also, the stability margins w.r.t. time delay and the sensitivity of the control approaches to uncertainties in the base motion estimation need to be evaluated more thoroughly. Stability independent of the time delay could be shown for the linearized model under certain conditions (see Appendix A9). For general passivation of the distributed nonlinear system, TDPC or wave variable approaches are readily applicable, as suggested in Section 2.5.

So far, independent models but centralized task control as well as a combined model and combined control are considered. Another possibility would be to utilize external wrench estimation and independent control of the flying robot to realize impedance control with inertia shaping, compliance in one desired direction, or force amplification, as proposed in [112]. Furthermore, closed-loop rate control could be considered instead of pose control of the flying robot. An admittance interface [177] to the task space controller can be used to provide desired rates instead of forces and torques generated by the impedance controller.

The increased computing power available on modern flying robots enables the application of computationally more intensive control approaches, such as nonlinear model predictive control [207]. In particular, disturbance rejection and actuator saturation is handled more naturally by model predictive controllers.

Table 7.2: Evaluation of the research questions stated in Section 1.3.

Research question	Addressed in chapter(s)	Sufficiently addressed
Q1 How can the dynamics of the robotic support system composed of manipulator on a moving base and flying robot be modeled?	2, 3, 4	○ ○ 🟢
Q2 What are suitable concepts for coordinated control of flying robot and robot manipulator on a moving base?	4, 5	○ ○ 🟢
Q3 How can both flying robot and robot manipulator contribute to a successful realization of the assistance task?	4, 5	○ ○ 🟢
Q4 How can the relative distance between manipulator and flying robot be measured accurately in order to realize robust take-off and landing?	6	○ ○ 🟢
Q5 How do the coordinated control approaches perform in simulations and real world experiments with different VTOL UAVs?	4, 5, 6	○ 🟡 ○

Recent developments in object tracking based on deep neural networks [192] suggest that artificial markers might not be necessary any more and arbitrary objects can be tracked at high frame rates. It is also imaginable, that the visual inertial state estimates of both flying robot and robot manipulator could be fused directly, for example using a particle filter [196] instead of the Kalman filter presented in this work.

Finally, the novel insights gained from studying cooperative control of the two considered heterogeneous robotic systems may be transferred to other domains. They are applicable to other systems and other applications as well, such as collaborative manipulation, aerial manipulation, or cooperative load transportation.

Appendices

A1 Most relevant terms regarding control architectures

To increase clarity, the most relevant terms used in the literature to describe control system architectures are defined below in the context of robot-assisted take-off and landing of flying robots.

independent

Two systems are *independent*, if they are not physically interacting with each other and do not depend on state information of the other system. Flying robot and robot manipulator are *independent* only during approach and departure phase.

combined

The dynamics of flying robot and robot manipulator can be *combined* to one model, which considers the bilateral coupling terms.

coupled / decoupled

The dynamics of flying robot and robot manipulator are bilaterally *coupled* in the combined system model. They may be *decoupled* by control, e.g. via feedback of the computed coupling terms.

decomposed (or separated)

The combined model is *decomposed* into submodels, which can be computed separately and, hence, used for distributed control. However, they are not independent.

distributed

Distributed control refers to the control of the combined system, where flying robot and robot manipulator use their own on-board control computer. Distributed control deals with control of two or more systems with independent control computers connected in a communication network.

coordinated

Flying robot and robot manipulator need to be *coordinated* in order to fulfill the task at hand. One of the two systems or a higher level instance may be responsible for the coordination.

overactuated (or redundant)

There is a difference between overactuation and redundancy. A system is *overactuated*, if the number of actuators is greater than the number of degrees of freedom. It is referred to as *redundant*, if it has more actuated degrees of freedom than are actually required for the task. If a system is overactuated, it is also redundant, but not vice versa.[1]

fully actuated

A robotic system is *fully actuated*, if the number of its degrees of freedom is equal to the number of its actuated degrees of freedom.

underactuated

A robotic system is *underactuated*, if the number of its degrees of freedom is smaller then the number of its actuated degrees of freedom. For example, a flying robot with multiple rotors but parallel thrust vectors in the body frame is underactuated with respect to its 6D pose, because it cannot apply forces perpendicular to the thrust direction.

cooperative control

Cooperative control is a distinct field in control theory. It considers multiple agents and the communication network between them.

centralized / decentralized control

One can distinguish if decisions are made centralized, i.e. if the states of all agents are known at one point, or decentralized, i.e. if decisions are made locally by an agent depending on its own state and maybe the states of its nearest neighbours.

shared control

Shared control means human operator control plus automation. This is not addressed in this work.

A2 Forward kinematics of DLR Light Weight Robot

In general, the forward transformation from coordinate frame k to frame $k-1$ can be carried out using a homogeneous transformation matrix $\boldsymbol{T}_{k-1,k} \in \mathrm{SE}(3)$ defined as

$$\boldsymbol{T}_{k-1,k} = \begin{bmatrix} c(\theta_k) & -s(\theta_k) & 0 & a_k \\ s(\theta_k)c(\alpha_{k-1}) & c(\theta_k)c(\alpha_{k-1}) & -s(\alpha_{k-1}) & -s(\alpha_{k-1})d_k \\ s(\theta_k)s(\alpha_{k-1}) & c(\theta_k)s(\alpha_{k-1}) & c(\alpha_{k-1}) & c(\alpha_{k-1})d_k \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A2.1}$$

[1] For example, a flying robot with six or eight rotors with parallel thrust vectors is underactuated, but redundant, i.e. it can lose a single rotor and still maintain stable flight. If the thrust vectors are arranged appropriately, the vehicle may become fully actuated or even overactuated.

where $s(\cdot) = \sin(\cdot)$, $c(\cdot) = \cos(\cdot)$, and a_{k-1}, α_{k-1}, d_k, and θ_k are the Denavit-Hartenberg (DH) parameters ([14], p. 67). The DH parameters of the DLR LWR 4+ used in this work are listed in Table A2.1.

Table A2.1: DH parameters of DLR LWR 4+.

k	a_{k-1}	α_{k-1} [deg]	d_k	θ_k
1	0	0	l_1	ϕ_1
2	0	90	0	ϕ_2
3	0	-90	l_2	ϕ_3
4	0	-90	0	ϕ_4
5	0	90	l_3	ϕ_5
6	0	90	0	ϕ_6
7	0	-90	l_4	ϕ_7

The lengths l_1 - l_3 are given in Figure 2.3 in Section 2.2, whereas l_4 is the length of the used end-effector. Finally, the forward kinematics transformation is obtained as

$$\boldsymbol{T}_{b,7} = \boldsymbol{T}_{b,1}\boldsymbol{T}_{1,2}\boldsymbol{T}_{2,3}\boldsymbol{T}_{3,4}\boldsymbol{T}_{4,5}\boldsymbol{T}_{5,6}\boldsymbol{T}_{6,7}, \tag{A2.2}$$

where b denotes the base of the manipulator.

A3 Derivation of the generalized pseudoinverse

Consider a vector of task space velocities $\dot{\boldsymbol{x}}$ and a vector of velocities in configuration space $\dot{\boldsymbol{\phi}}$, which are connected via the Jacobian matrix $\boldsymbol{J}$, such that

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{\phi}}. \tag{A3.1}$$

In order to derive a generalized pseudoinverse of $\boldsymbol{J}$, consider the optimization problem

$$\begin{aligned} &\min_{\dot{\boldsymbol{\phi}}} \frac{1}{2}\dot{\boldsymbol{\phi}}^T\boldsymbol{Q}\dot{\boldsymbol{\phi}} \\ \text{s. t.}\quad &\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{\phi}} = \boldsymbol{0}, \end{aligned}$$

which includes the metric $\boldsymbol{Q}$. Define

$$\begin{aligned} \Gamma &= \frac{1}{2}\dot{\boldsymbol{\phi}}^T\boldsymbol{Q}\dot{\boldsymbol{\phi}} + \left(\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{\phi}}\right)^T\boldsymbol{\lambda} && \text{(A3.2)} \\ &= \frac{1}{2}\dot{\boldsymbol{\phi}}^T\boldsymbol{Q}\dot{\boldsymbol{\phi}} + \boldsymbol{\lambda}^T\dot{\boldsymbol{x}} - \boldsymbol{\lambda}^T\boldsymbol{J}\dot{\boldsymbol{\phi}}, && \text{(A3.3)} \end{aligned}$$

where $\boldsymbol{\lambda}$ are the so-called Lagrange multipliers. The partial derivative of Γ w.r.t. $\dot{\boldsymbol{\phi}}$ is set to zero

$$\frac{\partial\Gamma}{\partial\dot{\boldsymbol{\phi}}} = \dot{\boldsymbol{\phi}}^T\boldsymbol{Q} - \boldsymbol{\lambda}^T\boldsymbol{J} = \boldsymbol{0}, \tag{A3.4}$$

which leads to

$$\dot{\boldsymbol{\phi}} = \boldsymbol{Q}^{-T}\boldsymbol{J}^T\boldsymbol{\lambda}. \tag{A3.5}$$

Setting the partial derivative of Γ w.r.t. the Lagrange multipliers to zero

$$\frac{\partial \Gamma}{\partial \boldsymbol{\lambda}} = \left(\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{\phi}}\right)^T = \boldsymbol{0} \tag{A3.6}$$

yields the forward kinematics mapping (A3.1). Multiplying $\boldsymbol{J}$ from the left in (A3.5) gives

$$\boldsymbol{J}\dot{\boldsymbol{\phi}} = \boldsymbol{J}\boldsymbol{Q}^{-T}\boldsymbol{J}^T\boldsymbol{\lambda}, \tag{A3.7}$$

which can be rearranged to

$$\boldsymbol{\lambda} = \left(\boldsymbol{J}\boldsymbol{Q}^{-T}\boldsymbol{J}^T\right)^{-1}\dot{\boldsymbol{x}} \tag{A3.8}$$

for any non-singular matrix $\boldsymbol{Q}$ and full row-rank Jacobian matrix $\boldsymbol{J}$. Inserting (A3.8) in (A3.5) eventually yields

$$\dot{\boldsymbol{\phi}} = \boldsymbol{Q}^{-T}\boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{Q}^{-T}\boldsymbol{J}^T\right)^{-1}\dot{\boldsymbol{x}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}}, \tag{A3.9}$$

where $\boldsymbol{J}^{\#}$ is the generalized pseudoinverse, which solves the minimization problem stated above. It can be directly seen, that $\boldsymbol{Q} = \boldsymbol{I}$ results in the Moore-Penrose pseudoinverse, which minimizes the Euclidean norm $\dot{\boldsymbol{\phi}}^T\dot{\boldsymbol{\phi}}$. With $\boldsymbol{Q} = \boldsymbol{M}$, where $\boldsymbol{M}$ is the inertia matrix, the dynamically consistent pseudoinverse [40], which minimizes the kinetic energy $\frac{1}{2}\dot{\boldsymbol{\phi}}^T\boldsymbol{M}\dot{\boldsymbol{\phi}}$, is obtained.

A4 Reference attitude from desired thrust vector

Given a desired thrust vector in the inertial frame $\boldsymbol{f}_d \in \mathbb{R}^3$, the desired orientation of a flying robot w.r.t. the inertial frame can be determined. The actual computation depends on the utilized attitude representation. Three different possibilities are summarized in the table below. The formula for the quaternion is derived in Appendix A5. Conversion between the attitude representations is possible, however, extracting Euler angles $(\varphi_d \quad \theta_d \quad \psi_d)$ or the quaternion $\boldsymbol{q}_d$ from the rotation matrix $\boldsymbol{R}_d$ is tedious [30].

Table A4.1: Computation of the reference attitude given a desired thrust vector $\boldsymbol{f}_d$.

	Attitude representation [Reference]	
Euler angles [126]	Rotation matrix [96, 99]	Quaternion [147]
$\theta_d = \arcsin\left(\frac{\boldsymbol{f}_d^T\boldsymbol{e}_1}{\lvert\lvert\boldsymbol{f}_d\rvert\rvert\rvert}\right)$	$\boldsymbol{z}_b = \frac{\boldsymbol{f}_d}{\lvert\lvert\boldsymbol{f}_d\rvert\rvert}$, $\boldsymbol{y}_b = \frac{\boldsymbol{z}_b \times \boldsymbol{x}_c}{\lvert\lvert\boldsymbol{z}_b \times \boldsymbol{x}_c\rvert\rvert}$,	$\boldsymbol{q}_{rp} = \frac{1}{\sqrt{2(1+\boldsymbol{f}_b^T\boldsymbol{f}_d)}}\begin{pmatrix}1 + \boldsymbol{f}_b^T\boldsymbol{f}_d \\ \boldsymbol{f}_b \times \boldsymbol{f}_d\end{pmatrix}$
$\varphi_d = -\arcsin\left(\frac{\boldsymbol{f}_d^T\boldsymbol{e}_2}{\lvert\lvert\boldsymbol{f}_d\rvert\rvert\rvert\cos(\theta_d)}\right)$	$\boldsymbol{x}_b = \boldsymbol{y}_b \times \boldsymbol{z}_b$,	$\boldsymbol{f}_b = \pm\boldsymbol{e}_3$, s.t. $\boldsymbol{f}_b^T\boldsymbol{f}_d \geq 0$
Set ψ_d as desired	with $\boldsymbol{x}_c = \begin{pmatrix}\cos(\psi_d) \\ \sin(\psi_d) \\ 0\end{pmatrix}$	$\boldsymbol{q}_c = \begin{pmatrix}\cos(\frac{\psi_d}{2}) \\ 0 \\ 0 \\ \sin(\frac{\psi_d}{2})\end{pmatrix}$
	$\boldsymbol{R}_d = \begin{bmatrix}\boldsymbol{x}_b & \boldsymbol{y}_b & \boldsymbol{z}_b\end{bmatrix}$	$\boldsymbol{q}_d = \boldsymbol{q}_{rp} \otimes \boldsymbol{q}_c$

A5 Quaternion from two vectors

This problem always arises in the control of underactuated flying robots. The desired thrust vector in the inertial frame $\boldsymbol{f}_i$ is commanded by the position controller. From $\boldsymbol{f}_i$ and the nominal thrust vector $\boldsymbol{f}_b$, which is usually equal to $\boldsymbol{e}_3 = (0 \quad 0 \quad 1)^T$, the quaternion reference $\boldsymbol{q}_d$ for the attitude controller needs to be found.

Consider the unit vectors $\boldsymbol{f}_b$, $\boldsymbol{f}_i$, and $\boldsymbol{n}$ and recall the definition of the unit quaternion (2.14)

$$\boldsymbol{q}_d = \begin{pmatrix} \cos(\frac{\varphi}{2}) \\ \sin(\frac{\varphi}{2})\boldsymbol{n} \end{pmatrix} \tag{A5.1}$$

where φ is the rotation angle and $\boldsymbol{n}$ is the rotation axis. The angle φ between $\boldsymbol{f}_b$ and $\boldsymbol{f}_i$ is given by

$$\cos(\varphi) = \boldsymbol{f}_b^T \boldsymbol{f}_i. \tag{A5.2}$$

Furthermore, it holds that

$$\sin(\varphi)\boldsymbol{n} = \boldsymbol{f}_b \times \boldsymbol{f}_i. \tag{A5.3}$$

Using the trigonometric relations

$$\cos(\frac{\varphi}{2}) = \pm\sqrt{\frac{1+\cos(\varphi)}{2}} \tag{A5.4}$$

and

$$\sin(\frac{\varphi}{2}) = \frac{\sin(\varphi)}{2\cos(\frac{\varphi}{2})}, \tag{A5.5}$$

it follows that

$$\cos(\frac{\varphi}{2}) = \pm\sqrt{\frac{1+\boldsymbol{f}_b^T\boldsymbol{f}_i}{2}} = \pm\frac{1+\boldsymbol{f}_b^T\boldsymbol{f}_i}{\sqrt{2(1+\boldsymbol{f}_b^T\boldsymbol{f}_i)}}, \tag{A5.6}$$

$$\sin(\frac{\varphi}{2})\boldsymbol{n} = \pm\frac{\sin(\varphi)\boldsymbol{n}}{\sqrt{2(1+\cos(\varphi))}} = \pm\frac{\boldsymbol{f}_b\times\boldsymbol{f}_i}{\sqrt{2(1+\boldsymbol{f}_b^T\boldsymbol{f}_i)}}, \tag{A5.7}$$

and, hence,

$$\boldsymbol{q}_d = \pm\frac{1}{\sqrt{2(1+\boldsymbol{f}_b^T\boldsymbol{f}_i)}}\begin{pmatrix} 1+\boldsymbol{f}_b^T\boldsymbol{f}_i \\ \boldsymbol{f}_b\times\boldsymbol{f}_i \end{pmatrix}. \tag{A5.8}$$

Interestingly, $\boldsymbol{q}_d$ is in fact a unit quaternion, due to $\left\|\begin{pmatrix} 1+\boldsymbol{f}_b^T\boldsymbol{f}_i \\ \boldsymbol{f}_b\times\boldsymbol{f}_i \end{pmatrix}\right\| = \sqrt{2(1+\boldsymbol{f}_b^T\boldsymbol{f}_i)}$.

A6 Efficient flip trajectory generation

Here, a computationally efficient method for computing trajectories for bidirectional thrust vehicles, including a flip to inverted flight, are presented. Due to their simplicity and defined derivatives, it is convenient to use polynomials for trajectory generation. A minimum jerk solution using fifth order polynomials is presented in [162]. This method is adopted and additional constraints for jerk consistency at the flip trajectory segment intersection are derived.

Only x-, y-, z and yaw angle ψ are treated, due to the flatness property w.r.t. x, y, z, ψ and their derivatives. For completeness, recall the fifth order polynomial formulation from [162]:

$$\begin{pmatrix} p(t) \\ v(t) \\ a(t) \\ j(t) \\ s(t) \end{pmatrix} = \begin{pmatrix} \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + \frac{a_0}{2}t^2 + v_0 t + p_0 \\ \frac{\alpha}{24}t^4 + \frac{\beta}{6}t^3 + \frac{\gamma}{2}t^2 + a_0 t + v_0 \\ \frac{\alpha}{6}t^3 + \frac{\beta}{2}t^2 + \gamma t + a_0 \\ \frac{\alpha}{2}t^2 + \beta t + \gamma \\ \alpha t + \beta \end{pmatrix}, \tag{A6.1}$$

wherein p, v, a, j, and s denote position, velocity, acceleration, jerk, and snap, respectively. The parameters α, β, and γ are obtained from

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \frac{1}{t_d^5} \begin{bmatrix} 720 & -360t_d & 60t_d^2 \\ -360t_d & 168t_d^2 & -24t_d^3 \\ 60t_d^2 & -24t_d^3 & 3t_d^4 \end{bmatrix} \begin{pmatrix} \Delta p \\ \Delta v \\ \Delta a \end{pmatrix}, \tag{A6.2}$$

where t_d is the duration of the trajectory segment and the initial (index 0) and final states (index f) are summarized in

$$\begin{pmatrix} \Delta p \\ \Delta v \\ \Delta a \end{pmatrix} = \begin{pmatrix} p_f - p_0 - v_0 t_d - \frac{1}{2}a_0 t_d^2 \\ v_f - v_0 - a_0 t_d \\ a_f - a_0 \end{pmatrix}. \tag{A6.3}$$

For a 180 deg ($\phi_f = \pm\pi$) flip it must hold that $\phi_h = \pm\frac{\pi}{2}$ at height $z_h = h$. It then follows from

$$\cos(\phi_h) = \frac{\boldsymbol{f}^T \boldsymbol{e}_3}{||\boldsymbol{f}^T \boldsymbol{e}_3||} = \frac{\ddot{z}_h + g}{||\ddot{z}_h + g||} = 0 \tag{A6.4}$$

where the force $\boldsymbol{f}$ in the inertial frame is given by the definition of the position controller, that the acceleration at z_h needs to be $\ddot{z}_h = -g$ and that the accelerations $\ddot{x}_h$ and $\ddot{y}_h$ are free. Since the position can be translated to any arbitrary initial position, without loss of generality, the start position is assumed to be at the origin. Furthermore, for a smooth trajectory jerk consistency at the intersection of two segments is required with $t_{d,1} = t_d, 2 = t_d$, i.e. $j_1(t_{d,1}) = j_2(0)$. This yields for the z-direction

$$\frac{\alpha_{z1}}{2}t_d^2 + \beta_{z1}t_d + \gamma_{z1} = \gamma_{z2} \tag{A6.5}$$

and with zero velocity at start 0, via-point h, and end f, as well as zero acceleration at start 0 and end f

$$\Delta z_1 = h, \qquad \Delta \dot{z}_1 = 0, \qquad \Delta \ddot{z}_1 = -g,$$

$$\Delta z_2 = -h + \tfrac{1}{2}g t_d^2, \quad \Delta \dot{z}_2 = g t_d, \quad \Delta \ddot{z}_2 = g,$$

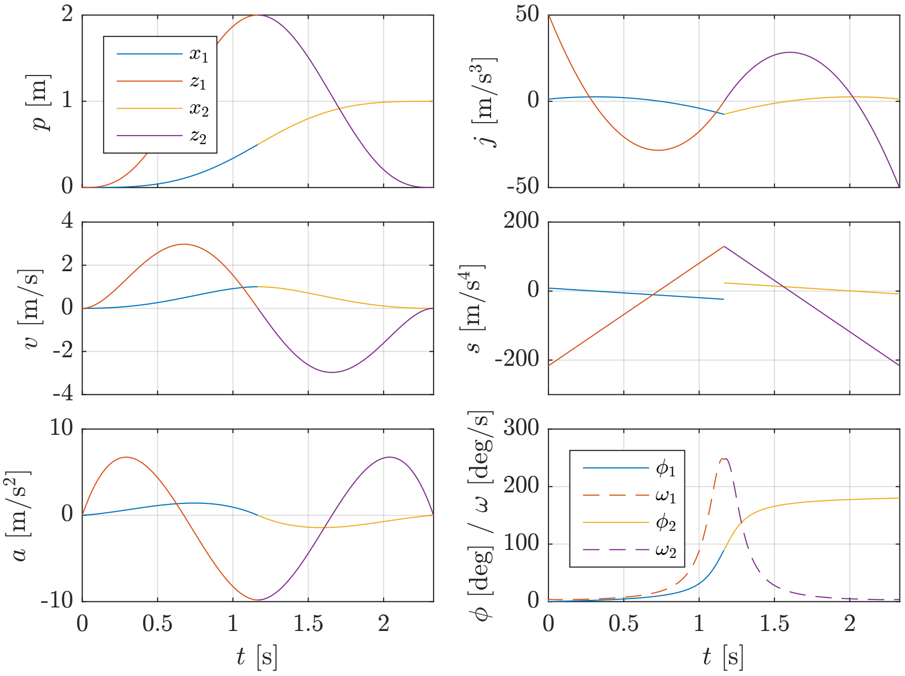


Figure A6.1: Exemplary generated flip trajectory.

and from (A6.2)

$$\begin{aligned}\alpha_{z1} &= \frac{1}{t_d^5}(720h - 60gt_d^2),\\ \beta_{z1} &= \frac{1}{t_d^5}(-360ht_d + 24gt_d^3),\\ \gamma_{z1} &= \frac{1}{t_d^5}(60ht_d^2 - 3gt_d^4),\\ \gamma_{z2} &= \frac{1}{t_d^5}(-60ht_d^2 + 9gt_d^4).\end{aligned} \tag{A6.6}$$

Inserting (A6.6) in (A6.5) yields the relationship between the height h of the flip and the duration t_d of one segment:

$$t_d = \sqrt{\frac{20h}{3g}}. \tag{A6.7}$$

This leads to zero jerk $\dddot{z}_h = 0$ at z_h. For the x-direction the velocity $\dot{x}_h$ at $x_h = \frac{x_f}{2}$ and the end position x_f is left open. One obtains

$$\begin{aligned}\Delta x_1 &= \tfrac{x_f}{2}, & \Delta\dot{x}_1 &= \dot{x}_h, & \Delta\ddot{x}_1 &= 0,\\ \Delta x_2 &= \tfrac{x_f}{2} - \dot{x}_h t_d, & \Delta\dot{x}_2 &= -\dot{x}_h, & \Delta\ddot{x}_2 &= 0.\end{aligned}$$

Due to zero acceleration $\ddot{x}_h = 0$, the jerk at x_h is always consistent. With the above results and (A6.7), the trajecory can be computed using (A6.1) and (A6.2). Thrust and angular rate feasibility, i.e.

$$f_{\min} \leq f(t) \leq f_{\max} \tag{A6.8}$$

and

$$\frac{1}{f(t)^2}||j(t)||^2 \leq \omega_{\max}^2, \tag{A6.9}$$

with

$$f(t) = m \left|\left| \begin{pmatrix} \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} \right|\right|, \tag{A6.10}$$

can be achieved by iterating over h, x_f, and $\dot{x}_h$ until a feasible trajectory is found.

A7 Flip quaternion and axis

A quaternion $\boldsymbol{q} = (q_w \quad q_x \quad q_y \quad q_z)^T = (\eta \quad \boldsymbol{\epsilon}^T)^T$ can be converted to a rotation matrix using the Euler-Rodrigues formula [28]

$$\begin{aligned} \boldsymbol{R}(\boldsymbol{q}) &= \left(\eta^2 - \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right)\boldsymbol{I}_{3\times 3} + 2\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T + 2\eta\boldsymbol{S}(\boldsymbol{\epsilon}) && \text{(A7.1)} \\ &= \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2q_xq_y-2q_wq_z & 2q_wq_y+2q_xq_z \\ 2q_wq_z+2q_xq_y & 1-2q_x^2-2q_z^2 & 2q_yq_z-2q_wq_x \\ 2q_xq_z-2q_wq_y & 2q_wq_x+2q_yq_z & 1-2q_x^2-2q_y^2 \end{bmatrix}. && \text{(A7.2)} \end{aligned}$$

It follows for a $\phi = 180\,\mathrm{deg}$ rotation, i.e. $q_w = \cos(\frac{\phi}{2}) = 0$

$$\overset{q_w=0}{=} \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2q_xq_y & 2q_xq_z \\ 2q_xq_y & 1-2q_x^2-2q_z^2 & 2q_yq_z \\ 2q_xq_z & 2q_yq_z & 1-2q_x^2-2q_y^2 \end{bmatrix}$$

and for a rotation axis in the xy-plane, i.e. $q_z = 0$

$$\overset{q_z=0}{=} \begin{bmatrix} 1-2q_y^2 & 2q_xq_y & 0 \\ 2q_xq_y & 1-2q_x^2 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

The rotation axis $\boldsymbol{\epsilon} = (q_x \quad q_y \quad 0)^T$ with $||\boldsymbol{\epsilon}|| = 1$ for a 180 deg flip can be obtained from initial and final position as

$$\boldsymbol{\epsilon} = \frac{\Delta\boldsymbol{p} \times \boldsymbol{e}_3}{||\Delta\boldsymbol{p} \times \boldsymbol{e}_3||} \tag{A7.3}$$

where $\Delta\boldsymbol{p} = \boldsymbol{p}_f - \boldsymbol{p}_0$ and $\boldsymbol{e}_3 = (0 \quad 0 \quad 1)^T$.

A8 Criterion for maximum end-effector acceleration

Here, a stability criterion for the maximum end-effector acceleration of the robot manipulator is derived considering the PD attitude controller of the flying robot. Recall the linear models (4.26) and (4.27) with (4.16) derived in Section 4.2.1 and Section 4.2.2, which may be written as

$$\dot{\varphi} = \omega_x \tag{A8.1}$$

$$\dot{\omega}_x = -k_{\omega_x}\omega_x - k_\varphi \varphi + m_u g l \varphi + \frac{m_u l}{I_{xx}} \ddot{r}_{EE,y} \tag{A8.2}$$

and

$$\dot{\theta} = \omega_y \tag{A8.3}$$

$$\dot{\omega}_y = -k_{\omega_y}\omega_y - k_\theta \theta + m_u g l \theta - \frac{m_u l}{I_{yy}} \ddot{r}_{EE,x}, \tag{A8.4}$$

respectively. First, consider the Lyapunov function candidate

$$V_x = \frac{1}{2}\omega_x^2 + \frac{1}{2}(k_\varphi - m_u g l)\varphi^2 \geq 0, \tag{A8.5}$$

with $k_\varphi > m_u g l$ (cf. condition (4.18)). Its first derivative w.r.t. time is

$$\begin{aligned}
\dot{V}_x &= \omega_x \dot{\omega}_x + k_\varphi \varphi \dot{\varphi} - m_u g l \varphi \dot{\varphi} && \text{(A8.6)}\\
&= -k_{\omega_x}\omega_x^2 - k_\varphi \varphi \omega_x + m_u g l \varphi \omega_x + \frac{m_u l}{I_{xx}} \ddot{r}_{EE,y}\omega_x + k_\varphi \varphi \dot{\varphi} - m_u g l \varphi \dot{\varphi} && \text{(A8.7)}\\
&= -k_{\omega_x}\omega_x^2 + \frac{m_u l}{I_{xx}} \ddot{r}_{EE,y}\omega_x, && \text{(A8.8)}
\end{aligned}$$

for which it has to hold that $\dot{V}_x \leq 0$ and, therefore,

$$\ddot{r}_{EE,y} \leq \frac{k_{\omega_x} I_{xx}}{m_u l}\omega_x. \tag{A8.9}$$

Second, consider the Lyapunov function candidate

$$V_y = \frac{1}{2}\omega_y^2 + \frac{1}{2}(k_\theta - m_u g l)\theta^2 \geq 0, \tag{A8.10}$$

with $k_\theta > m_u g l$ (cf. condition (4.18)). Its first derivative w.r.t. time is

$$\begin{aligned}
\dot{V}_y &= \omega_y \dot{\omega}_y + k_\theta \theta \dot{\theta} - m_u g l \theta \dot{\theta} && \text{(A8.11)}\\
&= -k_{\omega_y}\omega_y^2 - k_\theta \theta \omega_y + m_u g l \theta \omega_y - \frac{m_u l}{I_{yy}} \ddot{r}_{EE,x}\omega_y + k_\theta \theta \dot{\theta} - m_u g l \theta \dot{\theta} && \text{(A8.12)}\\
&= -k_{\omega_y}\omega_y^2 - \frac{m_u l}{I_{yy}} \ddot{r}_{EE,x}\omega_y, && \text{(A8.13)}
\end{aligned}$$

for which it has to hold that $\dot{V}_y \leq 0$ and, therefore,

$$\ddot{r}_{EE,x} \geq -\frac{k_{\omega_y} I_{yy}}{m_u l}\omega_y. \tag{A8.14}$$

A9 Proof of stability independent of time delay

Here, it is shown that stability of the closed-loop control of the manipulator and the flying robot may be assured independent from the time delay of the communication channel.

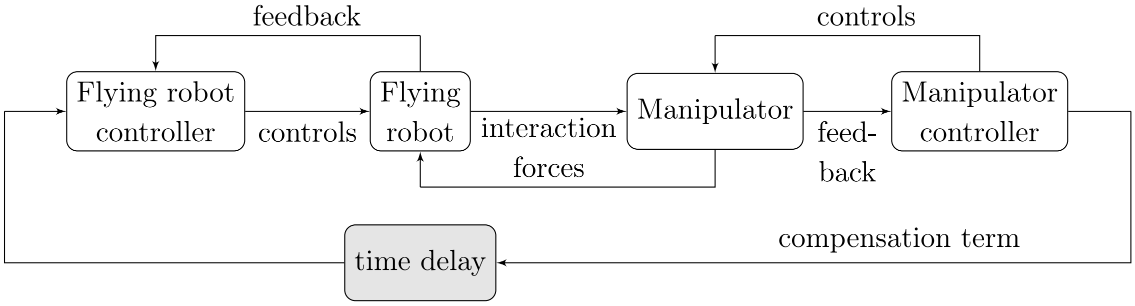


Figure A9.1: Control loop with time delay.

Figure A9.1 shows the considered interconnection of the flying robot and the manipulator. It is reasonable to neglect the internal time delays of the separate control loops of the flying robot and the manipulator, since they are not the main concern of the analysis presented here. However, a single time delay occurs depending on where the compensation term $\Delta\boldsymbol{\tau}_{att}$ (cf. (4.34) - (4.37) in Section 4.2.5 or (4.67) in Section 4.3.3) is computed, i.e. on the manipulator side using (delayed) measurements of the UAV orientation or on the UAV side using the (delayed) commanded force of the manipulator. Both cases boil down to a single time delay as depicted on Figure A9.1.

The following assumptions are made to allow a linear stability analysis:

- Only the motion in the x, z-plane and manipulator forces in the x-direction are considered, i.e. the applied force of the manipulator in the z-direction is assumed zero.
- The thrust force of the flying robot is always $T = mg$.
- A single time delay $d \geq 0$ is considered as pointed out above.

Recall the separate model (4.4) of the flying robot u derived in Section 4.1. Under the above assumptions, the model reduces to

$$\begin{pmatrix} \ddot{x}(t) \\ \dot{x}(t) \\ \ddot{\theta}(t) \\ \dot{\theta}(t) \end{pmatrix} = \begin{bmatrix} -\frac{d_x}{m} & -\frac{k_x}{m} & 0 & -gm_u \\ 1 & 0 & 0 & 0 \\ -\frac{d_x l}{Iyy} & -\frac{k_x l}{Iyy} & -\frac{d_\theta}{Iyy} & -\frac{k_\theta}{Iyy} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} \dot{x}(t) \\ x(t) \\ \dot{\theta}(t) \\ \theta(t) \end{pmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{d_x l}{Iyy} & \frac{k_x l}{Iyy} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \dot{x}(t-d) \\ x(t-d) \\ \dot{\theta}(t-d) \\ \theta(t-d) \end{pmatrix} \tag{A9.1}$$

where the second term on the right-hand side is the delayed compensation of the tilt torque, which is equal to $\boldsymbol{S}(\rho)\boldsymbol{R}_{ui}\boldsymbol{F}_{EE}$ in the nonlinear case. Equation (A9.1) is given in the general form ([65], p. 44)

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_0\boldsymbol{x}(t) + \boldsymbol{A}_1\boldsymbol{x}(t-d), \qquad d \geq 0. \tag{A9.2}$$

One can be distinguish between delay-independent and delay-dependent stability ([65], p. 31). Theorem 2.1 in [65] can be used to show delay-independent stability of system (A9.2).

Theorem A9.1. *The system (A9.2) is stable independent of the time delay if and only if*

(i) $\boldsymbol{A}_0$ *is stable,*

(ii) $\boldsymbol{A}_0 + \boldsymbol{A}_1$ *is stable,*

(iii) $r((j\omega \boldsymbol{E} - \boldsymbol{A}_0)^{-1}\boldsymbol{A}_1) < 1, \quad \forall \omega > 0$*, where* $r(\cdot)$ *denotes the spectral radius of a matrix.*

The proof of Theorem A9.1 can be found in [65] on pp. 44. If the system is not stable independent of the time delay, it is possible to determine the delay margin, i.e. the limit of d up to which the system is stable and at which it holds that ([65], p. 48)

$$\det(j\omega \boldsymbol{E} - \boldsymbol{A}_0 - \boldsymbol{A}_1 e^{-j\omega d}) = 0. \tag{A9.3}$$

A more general analysis of systems under time delay can be performed using the Lyapunov-Krasovskii stability theorem or the Razumikhin theorem [65]. However, this involves solving linear matrix inequalities (LMIs) and, as pointed out in [65], provides usually very conservative results.

In the following, the numerical values listed in Table A9.1 are used, which are taken from Chapter 4 and Chapter 5.

Table A9.1: Exemplary numerical values for stability analysis.

Symbol	m_u	I_{yy}	l	g	k_x	d_x	k_θ	d_θ
Value	0.48	0.007	0.06	9.81	1000	0.5	0.66	0.05
Unit	kg	kg m^2	m	m/s^2	N/m	Ns/m	N/m	Ns/m
Description	UAV mass	UAV inertia	lever arm of universal hinge	acceleration of gravity	stiffness of impedance controller	damping of impedance controller	stiffness of attitude controller	damping of attitude controller
Section	4.2.6	4.2.6	4.2.6	4.2.6	5.3.6	5.3.6	4.2.6	4.2.6

It is straightforward to verify that $\boldsymbol{A}_0$ and $\boldsymbol{A}_0 + \boldsymbol{A}_1$ of system (A9.1) are stable, since their eigenvalues

$$\begin{aligned} \boldsymbol{A}_0 : \quad & \lambda_1 = -0.56 + 45.86i, \quad \lambda_2 = -0.56 - 45.86i, \\ & \lambda_3 = -3.54 + 7.86i, \quad \lambda_4 = -3.54 - 7.86i \\ \boldsymbol{A}_0 + \boldsymbol{A}_1 : \quad & \lambda_1 = -0.52 + 45.64i, \quad \lambda_2 = -0.52 - 45.64i, \\ & \lambda_3 = -3.57 + 9.03i, \quad \lambda_4 = -3.57 - 9.03i \end{aligned} \tag{A9.4}$$

lie in the left complex half-plane. Empirically, it can be shown that the system can become unstable for an increased lever arm l and the constant controller gains in Table A9.1. On the contrary, appropriately increasing the attitude controller gains restores stability even with increased length l. Next, criterion (iii) in Theorem A9.1 is verified by computing the spectral radius, i.e. the maximum eigenvalue, of $(s\boldsymbol{E} - \boldsymbol{A}_0)^{-1}\boldsymbol{A}_1$. It is found analytically, that with the parameters in Table A9.1 the maximum eigenvalue is always below 1. Figure A9.2 depicts the numerical result for a defined range of s.

Hence, all three criteria of Theorem A9.1 are fulfilled and, therefore, system (A9.1) is stable independent of the time delay d. In summary, this illustrates that closed-loop controllers for robotic assistance of flying robots can be designed which result in stability independent of the time delay of the communication channel. On the other hand, if stability independent of the time delay can not be achieved, alternative methods, such as wave variables [62] or the time domain passivity approach [58], need to be implemented additionally.

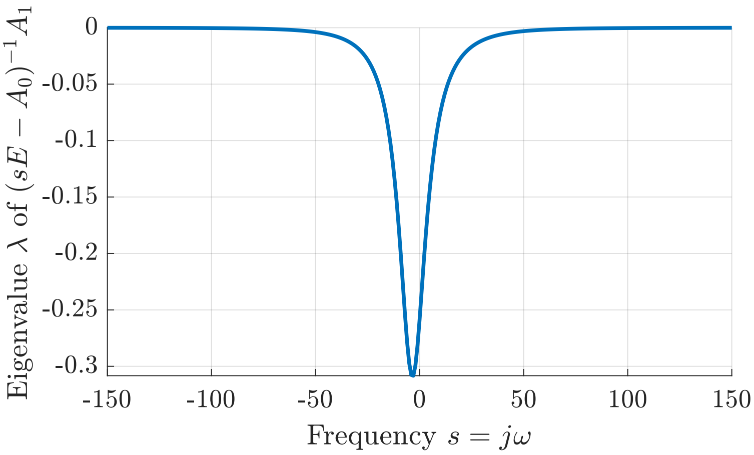


Figure A9.2: Maximum eigenvalue of system (A9.1) for different frequencies $j\omega$, i.e. time delays. It can be seen that the eigenvalue is < 1 for all ω and, hence, the system is stable independent of the time delay d.

A10 Adaptive control of combined system: Proof of passivity

Consider the dynamics of the combined system (5.41) with scalar multiplicative input uncertainty and the adaptive inverse dynamics controller (5.43) or the adaptive impedance controller (5.44), respectively.

Expanding (5.41) on both sides by $-(1+\varepsilon)\boldsymbol{\nu}$ leads for inverse dynamics control to

$$(1+\varepsilon)\left(\boldsymbol{\Lambda}(\ddot{\tilde{\boldsymbol{x}}} + \boldsymbol{P}\dot{\tilde{\boldsymbol{x}}} + \boldsymbol{K}\dot{\tilde{\boldsymbol{x}}} + \boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{x}}) + \boldsymbol{\mu} - \boldsymbol{\mu}_d\right) = \boldsymbol{F}_\tau + (1+\varepsilon)\boldsymbol{F}_{ext} - (1+\varepsilon)\boldsymbol{\nu} \qquad \text{(A10.1)}$$

and for impedance control to

$$(1+\varepsilon)\left(\boldsymbol{\Lambda}\ddot{\tilde{\boldsymbol{x}}} + \boldsymbol{\Lambda}\boldsymbol{P}\dot{\tilde{\boldsymbol{x}}} + \boldsymbol{K}\dot{\tilde{\boldsymbol{x}}} + \boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{x}} + \boldsymbol{\mu} - \boldsymbol{\mu}_d\right) = \boldsymbol{F}_\tau + (1+\varepsilon)\boldsymbol{F}_{ext} - (1+\varepsilon)\boldsymbol{\nu}. \qquad \text{(A10.2)}$$

Inserting (5.43) in (A10.1) or (5.44) in (A10.2), respectively, yields the dynamics of the combined tracking error $\boldsymbol{s} = \dot{\tilde{\boldsymbol{x}}} + \boldsymbol{P}\tilde{\boldsymbol{x}}$ as

$$\overline{\boldsymbol{\Lambda}}(\dot{\boldsymbol{s}} + \boldsymbol{K}\boldsymbol{s}) + \overline{\boldsymbol{\mu}} - \overline{\boldsymbol{\mu}}_d = \tilde{\varepsilon}\boldsymbol{\nu} + \overline{\boldsymbol{F}}_{ext}, \qquad \text{(inverse dynamics control)} \qquad \text{(A10.3)}$$

$$\overline{\boldsymbol{\Lambda}}\dot{\boldsymbol{s}} + \overline{\boldsymbol{K}}\boldsymbol{s} + \overline{\boldsymbol{\mu}} - \overline{\boldsymbol{\mu}}_d = \tilde{\varepsilon}\boldsymbol{\nu} + \overline{\boldsymbol{F}}_{ext}, \qquad \text{(impedance control)} \qquad \text{(A10.4)}$$

where $\tilde{\varepsilon} = \hat{\varepsilon} - \varepsilon$ and $\overline{\boldsymbol{\Lambda}} = (1+\varepsilon)\boldsymbol{\Lambda}$, $\overline{\boldsymbol{K}} = (1+\varepsilon)\boldsymbol{K}$, $\overline{\boldsymbol{\mu}} = (1+\varepsilon)\boldsymbol{\mu}$, $\overline{\boldsymbol{\mu}}_d = (1+\varepsilon)\boldsymbol{\mu}_d$, and $\overline{\boldsymbol{F}}_{ext} = (1+\varepsilon)\boldsymbol{F}_{ext}$ for brevity. Now, consider the positive definite Lyapunov function candidate

$$V = \frac{1}{2}\left(\boldsymbol{s}^T\overline{\boldsymbol{\Lambda}}\boldsymbol{s} + \tilde{\boldsymbol{x}}^T\boldsymbol{Q}\tilde{\boldsymbol{x}} + \frac{1}{\gamma}\tilde{\varepsilon}^2\right), \qquad \text{(A10.5)}$$

with constant gain $\gamma > 0$. An appropriate structure of the positive definite matrix $\boldsymbol{Q}$ is found in the following. The first derivative of (A10.5) w.r.t. time along the solution of the differential equations (A10.3) or (A10.4), respectively, is

$$\dot{V} = \boldsymbol{s}^T\overline{\boldsymbol{\Lambda}}\dot{\boldsymbol{s}} + \frac{1}{2}\boldsymbol{s}^T\dot{\overline{\boldsymbol{\Lambda}}}\boldsymbol{s} + \tilde{\boldsymbol{x}}^T\boldsymbol{Q}\dot{\tilde{\boldsymbol{x}}} + \frac{1}{2}\tilde{\boldsymbol{x}}^T\dot{\boldsymbol{Q}}\tilde{\boldsymbol{x}} + \frac{1}{\gamma}\tilde{\varepsilon}\dot{\hat{\varepsilon}}. \qquad \text{(A10.6)}$$

Note that ε is assumed to be constant. Inserting (A10.3), the definition of the tracking error $\boldsymbol{s} = \dot{\tilde{\boldsymbol{x}}} + \boldsymbol{P}\tilde{\boldsymbol{x}}$, and the adaptation law $\dot{\hat{\varepsilon}} = -\gamma \boldsymbol{s}^T\boldsymbol{\nu}$ gives

$$\begin{aligned}\dot{V} = &- \dot{\tilde{\boldsymbol{x}}}^T\overline{\boldsymbol{\Lambda}}\boldsymbol{K}\dot{\tilde{\boldsymbol{x}}} - \dot{\tilde{\boldsymbol{x}}}^T\overline{\boldsymbol{\Lambda}}\boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}^T\boldsymbol{P}\overline{\boldsymbol{\Lambda}}\boldsymbol{K}\dot{\tilde{\boldsymbol{x}}} - \tilde{\boldsymbol{x}}^T\boldsymbol{P}\overline{\boldsymbol{\Lambda}}\boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{x}} \\ &+ \frac{1}{2}\boldsymbol{s}^T\dot{\overline{\boldsymbol{\Lambda}}}\boldsymbol{s} + \boldsymbol{s}^T(\overline{\boldsymbol{\mu}}_d - \overline{\boldsymbol{\mu}}) + \tilde{\boldsymbol{x}}^T\boldsymbol{Q}\dot{\tilde{\boldsymbol{x}}} + \frac{1}{2}\tilde{\boldsymbol{x}}^T\dot{\boldsymbol{Q}}\tilde{\boldsymbol{x}} + \boldsymbol{s}^T\overline{\boldsymbol{F}}_{ext}.\end{aligned} \tag{A10.7}$$

For $\boldsymbol{Q} = 2\boldsymbol{P}\overline{\boldsymbol{\Lambda}}\boldsymbol{K}$, the term $-\dot{\tilde{\boldsymbol{x}}}^T\overline{\boldsymbol{\Lambda}}\boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}^T\boldsymbol{P}\overline{\boldsymbol{\Lambda}}\boldsymbol{K}\dot{\tilde{\boldsymbol{x}}}$ vanishes due to the symmetry of $\overline{\boldsymbol{\Lambda}}$, $\boldsymbol{K}$, and $\boldsymbol{P}$. In addition, choosing $\boldsymbol{\mu}_d = \boldsymbol{\mu} - \frac{1}{2}\dot{\boldsymbol{\Lambda}}\boldsymbol{s}$ cancels the derivative $\dot{\overline{\boldsymbol{\Lambda}}}$ of the kinetic energy matrix. Finally, inserting $\dot{\boldsymbol{Q}} = 2\boldsymbol{P}\dot{\overline{\boldsymbol{\Lambda}}}\boldsymbol{K}$, yields

$$\dot{V} = -\dot{\tilde{\boldsymbol{x}}}^T\overline{\boldsymbol{\Lambda}}\boldsymbol{K}\dot{\tilde{\boldsymbol{x}}} - \tilde{\boldsymbol{x}}^T\boldsymbol{P}\overline{\boldsymbol{\Lambda}}\boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{x}}^T\boldsymbol{P}\dot{\overline{\boldsymbol{\Lambda}}}\boldsymbol{K}\tilde{\boldsymbol{x}} + \boldsymbol{s}^T\overline{\boldsymbol{F}}_{ext}. \tag{A10.8}$$

Therefore, $\dot{V} \leq \boldsymbol{s}^T\overline{\boldsymbol{F}}_{ext}$ and, hence, passivity w.r.t. the power port $\boldsymbol{s}^T\overline{\boldsymbol{F}}_{ext}$ can only be shown as long as $\tilde{\boldsymbol{x}}^T\boldsymbol{P}\overline{\boldsymbol{\Lambda}}\boldsymbol{K}\boldsymbol{P}\tilde{\boldsymbol{x}} \geq \tilde{\boldsymbol{x}}^T\boldsymbol{P}\dot{\overline{\boldsymbol{\Lambda}}}\boldsymbol{K}\tilde{\boldsymbol{x}}$ and if $\overline{\boldsymbol{\Lambda}}\boldsymbol{K}$ and $\boldsymbol{P}\overline{\boldsymbol{\Lambda}}\boldsymbol{K}$, i.e. $\boldsymbol{Q}$, are positive semidefinite. These requirements are clearly a disadvantage of inverse dynamics control compared to impedance control.

On the contrary, inserting (A10.4), $\boldsymbol{s} = \dot{\tilde{\boldsymbol{x}}} + \boldsymbol{P}\tilde{\boldsymbol{x}}$, and $\dot{\hat{\varepsilon}} = -\gamma\boldsymbol{s}^T\boldsymbol{\nu}$ in (A10.6) gives

$$\dot{V} = - \dot{\tilde{\boldsymbol{x}}}^T\overline{\boldsymbol{K}}\dot{\tilde{\boldsymbol{x}}} - \dot{\tilde{\boldsymbol{x}}}^T\overline{\boldsymbol{K}}\boldsymbol{P}\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}^T\boldsymbol{P}\overline{\boldsymbol{K}}\dot{\tilde{\boldsymbol{x}}} - \tilde{\boldsymbol{x}}^T\boldsymbol{P}\overline{\boldsymbol{K}}\boldsymbol{P}\tilde{\boldsymbol{x}} \tag{A10.9}$$

$$+ \frac{1}{2}\boldsymbol{s}^T\dot{\overline{\boldsymbol{\Lambda}}}\boldsymbol{s} + \boldsymbol{s}^T(\overline{\boldsymbol{\mu}}_d - \overline{\boldsymbol{\mu}}) + \tilde{\boldsymbol{x}}^T\boldsymbol{Q}\dot{\tilde{\boldsymbol{x}}} + \frac{1}{2}\tilde{\boldsymbol{x}}^T\dot{\boldsymbol{Q}}\tilde{\boldsymbol{x}} + \boldsymbol{s}^T\overline{\boldsymbol{F}}_{ext}. \tag{A10.10}$$

For $\boldsymbol{Q} = 2\boldsymbol{P}\overline{\boldsymbol{K}}$, the term $-\dot{\tilde{\boldsymbol{x}}}^T\overline{\boldsymbol{K}}\boldsymbol{P}\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}^T\boldsymbol{P}\overline{\boldsymbol{K}}\dot{\tilde{\boldsymbol{x}}}$ vanishes due to the symmetry of $\overline{\boldsymbol{K}}$ and $\boldsymbol{P}$. Again, choosing $\boldsymbol{\mu}_d = \boldsymbol{\mu} - \frac{1}{2}\dot{\boldsymbol{\Lambda}}\boldsymbol{s}$ cancels the derivative $\dot{\overline{\boldsymbol{\Lambda}}}$ of the kinetic energy matrix. Since $\dot{\boldsymbol{Q}} = \boldsymbol{0}$, it follows that

$$\dot{V} = -\dot{\tilde{\boldsymbol{x}}}^T\overline{\boldsymbol{K}}\dot{\tilde{\boldsymbol{x}}} - \tilde{\boldsymbol{x}}^T\boldsymbol{P}\overline{\boldsymbol{K}}\boldsymbol{P}\tilde{\boldsymbol{x}} + \boldsymbol{s}^T\overline{\boldsymbol{F}}_{ext} \leq \boldsymbol{s}^T\overline{\boldsymbol{F}}_{ext}. \tag{A10.11}$$

This shows passivity w.r.t. the power port $\boldsymbol{s}^T\overline{\boldsymbol{F}}_{ext}$ of the closed-loop dynamics (A10.4) under adaptive impedance control. The only requirement, in addition to positive definite gain matrices $\boldsymbol{P}$ and $\boldsymbol{K}$, is that $\boldsymbol{Q}$ is positive semidefinite.

List of Figures

List of Tables

Bibliography

[1] Kimon P. Valavanis and George J. Vachtsevanos, ed. *Handbook of Unmanned Aerial Vehicles - 5 Volume Set.* New York, NY, USA: Springer, 2014. ISBN: 9048197066.

[2] Tin Muskardin, Georg Balmer, Linnea Persson, Sven Wlach, Maximilian Laiacker, Aníbal Ollero, and Konstantin Kondak. “A Novel Landing System to Increase Payload Capacity and Operational Availability of High Altitude Long Endurance UAVs”. In: *Journal of Intelligent and Robotic Systems* 88.2-4 (2017), pp. 597–618.

[3] SwissDrones Operating AG. *SwissDrones SDO 50 V2.* online. [Online; accessed June 6, 2018]. URL: `http://www.swissdrones.com/sdo-50/`.

[4] Quantum Systems GmbH. *Quantum Tron.* online. [Online; accessed June 6, 2018]. URL: `https://www.quantum-systems.com/tron/`.

[5] F. Ruggiero, V. Lippiello, and A. Ollero. “Aerial Manipulation: A Literature Review”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1957–1964. ISSN: 2377-3766. DOI: `10.1109/LRA.2018.2808541`.

[6] Yamaha Motor Corporation. *Yamaha RMAX helicopter.* online. [Online; accessed June 6, 2018]. URL: `https://www.yamahamotorsports.com/motorsports/pages/precision-agriculture-rmax`.

[7] SZ DJI Technology Co. Ltd. *DJI Inspire 1.* online. [Online; accessed June 6, 2018]. URL: `https://www.dji.com/de/inspire-1`.

[8] Intel Corporation. *Intel Falcon 8.* online. [Online; accessed June 6, 2018]. URL: `https://www.intel.de/content/www/de/de/products/drones/falcon-8.html`.

[9] A. Campos, J. Quintero, R. Saltaren, M. Ferre, and R. Aracil. “An Active helideck testbed for floating structures based on a Stewart-Gough platform”. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2008, pp. 3705–3710. DOI: `10.1109/IROS.2008.4650750`.

[10] R. Godzdanker, M. J. Rutherford, and K. P. Valavanis. “ISLANDS: A Self-Leveling landing platform for autonomous miniature UAVs”. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM).* 2011, pp. 170–175. DOI: `10.1109/AIM.2011.6027085`.

[11] Stephen A. Conyers, Nikolaos I. Vitzilaios, Matthew J. Rutherford, and Kimon P. Valavanis. “A Mobile Self-Leveling Landing Platform for VTOL UAVs”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2015, pp. 815–822. DOI: `10.1109/ICRA.2015.7139272`.

[12] L.A. Sandino, D. Santamaria, M. Bejar, A. Viguria, K. Kondak, and A. Ollero. “Tether-guided landing of unmanned helicopters without GPS sensors”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2014, pp. 3096–3101. DOI: `10.1109/ICRA.2014.6907304`.

[13] S.-R. Oh, K. Pathak, S.K. Agrawal, H.R. Pota, and M. Garratt. "Approaches for a tether-guided landing of an autonomous helicopter". In: *IEEE Transactions on Robotics* 22.3 (2006), pp. 536–544. DOI: 10.1109/TRO.2006.870657.

[14] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Prentice Hall, 1986. ISBN: 0201103265.

[15] H. Hirschmüller. "Accurate and efficient stereo processing by semi-global matching and mutual information". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. 2005, 807–814 vol. 2. DOI: 10.1109/CVPR.2005.56.

[16] T. Tomić, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka. "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue". In: *IEEE Robotics and Automation Magazine* 19.3 (2012), pp. 46–56. ISSN: 1070-9932. DOI: 10.1109/MRA.2012.2206473.

[17] K. Schmid, T. Tomic, F. Ruess, H. Hirschmüller, and M. Suppa. "Stereo vision based indoor/outdoor navigation for flying robots". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013, pp. 3955–3962. DOI: 10.1109/IROS.2013.6696922.

[18] S. G. Brunner, F. Steinmetz, R. Belder, and A. Dömel. "RAFCON: A graphical tool for engineering complex, robotic tasks". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 3283–3290. DOI: 10.1109/IROS.2016.7759506.

[19] Teodor Tomić. "Model-based control of flying robots for robust interaction under wind influence". Diss. Hannover: Gottfried Wilhelm Leibniz University, 2018, p. 144. DOI: 10.15488/3987.

[20] G. Hirzinger, A. Albu-Schaffer, M. Hahnle, I. Schaefer, and N. Sporer. "On a new generation of torque controlled light-weight robots". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. 2001, 3356–3363 vol.4. DOI: 10.1109/ROBOT.2001.933136.

[21] G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, and M. Schedl. "DLR's torque-controlled light weight robot III-are we reaching the technological limits now?" In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 2002, 1710–1716 vol.2. DOI: 10.1109/ROBOT.2002.1014788.

[22] A. Albu-Schäffer, C. Ott, and G. Hirzinger. "A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots". In: *International Journal of Robotics Research* 26.1 (2007), pp. 23–39. DOI: 10.1177/0278364907073776. URL: 10.1177/0278364907073776.

[23] A. Albu-Schäffer, C. Ott, U. Frese, and G. Hirzinger. "Cartesian impedance control of redundant robots: recent results with the DLR-light-weight-arms". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3. 2003, 3704–3709 vol.3. DOI: 10.1109/ROBOT.2003.1242165.

[24] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control.* Springer Science & Business Media, 2010. ISBN: 9781846286414.

[25] Franziska Zacharias. *Knowledge Representations for Planning Manipulation Tasks.* Berlin Heidelberg: Springer Science & Business Media, 2012. ISBN: 978-3-642-25182-5.

[26] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch. "Rigid-Body Attitude Control". In: *IEEE Control Systems* 31.3 (2011), pp. 30–51. ISSN: 1066-033X. DOI: 10.1109/MCS.2011.940459.

[27] M. D. Shuster. "A Survey of Attitude Representations". In: *Journal of the Astronautical Sciences* 41 (1993), pp. 439–517.

[28] Jack B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality.* Princeton University Press, 2002. ISBN: 9780691102986.

[29] T. Lee, M. Leoky, and N. H. McClamroch. "Geometric tracking control of a quadrotor UAV on SE(3)". In: *IEEE Conference on Decision and Control (CDC).* 2010, pp. 5420–5425. DOI: 10.1109/CDC.2010.5717652.

[30] James Diebel. "Representing attitude: Euler angles, unit quaternions, and rotation vectors". In: *Matrix* 58.15-16 (2006), pp. 1–35.

[31] Hartmut Bremer. *Elastic Multibody Dynamics: A Direct Ritz Approach.* New York, NY, USA: Springer, 2008. ISBN: 1402086792.

[32] G. Garofalo, C. Ott, and A. Albu-Schäffer. "On the closed form computation of the dynamic matrices and their differentiations". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2013, pp. 2364–2359. DOI: 10.1109/IROS.2013.6696688.

[33] William M. Silver. "On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators". In: *The International Journal of Robotics Research* 1.2 (1982), pp. 60–70. DOI: 10.1177/027836498200100204.

[34] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation.* Boca Raton, Florida: CRC Press, 1994. ISBN: 9780849379819.

[35] Hubert Gattringer. *Starr-elastische Robotersysteme - Theorie und Anwendungen.* 2011. Aufl. Berlin Heidelberg New York: Springer-Verlag, 2011. ISBN: 978-3-642-22828-5.

[36] L. J. Love, J. F. Jansen, and F. G. Pin. "On the modeling of robots operating on ships". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* Vol. 3. 2004, 2436–2443 Vol.3. DOI: 10.1109/ROBOT.2004.1307426.

[37] C. Ott. *Cartesian Impedance Control of Redundant and Flexible-Joint Robots.* New York, NY, USA: Springer, 2008. ISBN: 9783540692539.

[38] O. Khatib. "A unified approach for motion and force control of robot manipulators: The operational space formulation". In: *IEEE Journal of Robotics and Automation* 3.1 (1987), pp. 43–53. DOI: 10.1109/JRA.1987.1087068.

[39] Bojan Nemec and Leon Zlajpah. “Null space velocity control with dynamically consistent pseudo-inverse”. In: *Robotica* 18.5 (2000), pp. 513–518.

[40] Oussama Khatib. “Inertial Properties in Robotic Manipulation: An Object-Level Framework”. In: *The International Journal of Robotics Research* 14.1 (1995), pp. 19–36. DOI: 10.1177/027836499501400103.

[41] H. Henderson and S. Searle. “On Deriving the Inverse of a Sum of Matrices”. In: *SIAM Review* 23.1 (1981), pp. 53–60. DOI: 10.1137/1023004.

[42] A. Ollero, G. Heredia, A. Franchi, G. Antonelli, K. Kondak, A. Sanfeliu, A. Viguria, J. R. Martinez-de Dios, F. Pierri, J. Cortes, A. Santamaria-Navarro, M. A. Trujillo Soto, R. Balachandran, J. Andrade-Cetto, and A. Rodriguez. “The AEROARMS Project: Aerial Robots with Advanced Manipulation Capabilities for Inspection and Maintenance”. In: *IEEE Robotics and Automation Magazine* 25.4 (2018), pp. 12–23. ISSN: 1558-223X. DOI: 10.1109/MRA.2018.2852789.

[43] G. Muscio, F. Pierri, M. A. Trujillo, E. Cataldi, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero. “Coordinated Control of Aerial Robotic Manipulators: Theory and Experiments”. In: *IEEE Transactions on Control Systems Technology* 26.4 (2018), pp. 1406–1413. ISSN: 2374-0159. DOI: 10.1109/TCST.2017.2716905.

[44] M. Orsag, C. Korpela, S. Bogdan, and P. Oh. “Dexterous Aerial Robots-Mobile Manipulation Using Unmanned Aerial Systems”. In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1453–1466. ISSN: 1941-0468. DOI: 10.1109/TRO.2017.2750693.

[45] Markus Ryll, Giuseppe Muscio, Francesco Pierri, Elisabetta Cataldi, Gianluca Antonelli, Fabrizio Caccavale, Davide Bicego, and Antonio Franchi. “6D interaction control with aerial robots: The flying end-effector paradigm”. In: *The International Journal of Robotics Research* 38.9 (2019), pp. 1045–1062. DOI: 10.1177/0278364919856694.

[46] M. J. Kim, K. Kondak, and C. Ott. “A Stabilizing Controller for Regulation of UAV With Manipulator”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1719–1726. ISSN: 2377-3774. DOI: 10.1109/LRA.2018.2803205.

[47] G. Garofalo, F. Beck, and C. Ott. “Task-space Tracking Control for Underactuated Aerial Manipulators”. In: *European Control Conference (ECC).* 2018, pp. 628–634. DOI: 10.23919/ECC.2018.8550248.

[48] S. Kim, S. Choi, and H. J. Kim. “Aerial manipulation using a quadrotor with a two DOF robotic arm”. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2013, pp. 4990–4995. DOI: 10.1109/IROS.2013.6697077.

[49] H. Yang and D. Lee. “Dynamics and control of quadrotor with robotic manipulator”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2014, pp. 5544–5549. DOI: 10.1109/ICRA.2014.6907674.

[50] E. Yilmaz, H. Zaki, and M. Unel. “Nonlinear Adaptive Control of an Aerial Manipulation System”. In: *European Control Conference (ECC).* 2019, pp. 3916–3921. DOI: 10.23919/ECC.2019.8795709.

[51] V. Lippiello, R. Mebarki, and F. Ruggiero. “Visual Coordinated Landing of a UAV on a Mobile Robot Manipulator”. In: *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2013, pp. 1–7. DOI: 10.1109/SSRR.2013.6719338.

[52] E. Narváez, A. A. Ravankar, A. Ravankar, Y. Kobayashi, and T. Emaru. “Vision based autonomous docking of VTOL UAV using a mobile robot manipulator”. In: *IEEE/SICE International Symposium on System Integration (SII)*. 2017, pp. 157–163. DOI: 10.1109/SII.2017.8279205.

[53] D. Lee, T. Ryan, and H. J. Kim. “Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing”. In: *ICRA*. 2012, pp. 971–976. DOI: 10.1109/ICRA.2012.6224828.

[54] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza. “Vision-based autonomous quadrotor landing on a moving platform”. In: *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, pp. 200–207. DOI: 10.1109/SSRR.2017.8088164.

[55] P. J. From, V. Duindam, J. T. Gravdahl, and S. Sastry. “Modeling and motion planning for mechanisms on a non-inertial base”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 3320–3326. DOI: 10.1109/ROBOT.2009.5152666.

[56] P. J. From, J. T. Gravdahl, and P. Abbeel. “On the influence of ship motion prediction accuracy on motion planning and control of robotic manipulators on seaborne platforms”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 5281–5288. DOI: 10.1109/ROBOT.2010.5509813.

[57] A. Marino. “Distributed Adaptive Control of Networked Cooperative Mobile Manipulators”. In: *IEEE Transactions on Control Systems Technology* 26.5 (2018), pp. 1646–1660. ISSN: 2374-0159. DOI: 10.1109/TCST.2017.2720673.

[58] B. Hannaford and Jee-Hwan Ryu. “Time domain passivity control of haptic interfaces”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 2001, 1863–1869 vol.2. DOI: 10.1109/ROBOT.2001.932880.

[59] K. Hertkorn, T. Hulin, P. Kremer, C. Preusche, and G. Hirzinger. “Time Domain Passivity Control for multi-degree of freedom haptic devices with time delay”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 1313–1319. DOI: 10.1109/ROBOT.2010.5509148.

[60] Jee-Hwan Ryu, Jordi Artigas, and Carsten Preusche. “A passive bilateral control scheme for a teleoperator with time-varying communication delay”. In: *Mechatronics* 20.7 (2010). Special Issue on Design and Control Methodologies in Telerobotics, pp. 812 –823. ISSN: 0957-4158. DOI: 10.1016/j.mechatronics.2010.07.006.

[61] B. Hannaford and Jee-Hwan Ryu. “Time-domain passivity control of haptic interfaces”. In: *IEEE Transactions on Robotics and Automation* 18.1 (2002), pp. 1–10. ISSN: 2374-958X. DOI: 10.1109/70.988969.

[62] G. Niemeyer and J.-J.E. Slotine. “Using wave variables for system analysis and robot control”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 1997, 1619–1625 vol.2. DOI: 10.1109/ROBOT.1997.614372.

[63] C. Ott and Yoshihiko Nakamura. "Employing wave variables for coordinated control of robots with distributed control architecture". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2008, pp. 575–582. DOI: 10.1109/ROBOT.2008.4543268.

[64] Günter Niemeyer and Jean-Jacques E. Slotine. "Telemanipulation with Time Delays". In: *The International Journal of Robotics Research* 23.9 (2004), pp. 873–890. DOI: 10.1177/0278364904045563.

[65] Keqin Gu, Jie Chen, and Vladimir L. Kharitonov. *Stability of Time-Delay Systems.* Berlin Heidelberg: Springer Science & Business Media, 2003. ISBN: 978-0-817-64212-9.

[66] Min Wu, Yong He, and Jin-Hua She. *Stability Analysis and Robust Control of Time-Delay Systems.* 2010. Aufl. Berlin Heidelberg: Springer Science & Business Media, 2010. ISBN: 978-3-642-03037-6.

[67] J. Birk. "State estimation for processes with discrete-time and time-delayed measurements". In: *European Control Conference (ECC).* 1999, pp. 4809–4812. DOI: 10.23919/ECC.1999.7100097.

[68] T. D. Larsen, N. A. Andersen, O. Ravn, and N. K. Poulsen. "Incorporation of time delayed measurements in a discrete-time Kalman filter". In: *IEEE Conference on Decision and Control (CDC).* Vol. 4. 1998, 3972–3977 vol.4. DOI: 10.1109/CDC.1998.761918.

[69] Huanshui Zhang, Xiao Lu, and Daizhan Cheng. "Optimal estimation for continuous-time systems with delayed measurements". In: *IEEE Transactions on Automatic Control* 51.5 (2006), pp. 823–827. ISSN: 2334-3303. DOI: 10.1109/TAC.2006.874983.

[70] V. Lippiello and F. Ruggiero. "Exploiting redundancy in Cartesian impedance control of UAVs equipped with a robotic arm". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2012, pp. 3768–3773. DOI: 10.1109/IROS.2012.6386021.

[71] F. Huber, K. Kondak, K. Krieger, D. Sommer, M. Schwarzbach, M. Laiacker, I. Kossyk, S. Parusel, S. Haddadin, and A. Albu-Schäffer. "First analysis and experiments in aerial manipulation using fully actuated redundant robot arm". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2013, pp. 3452–3457. DOI: 10.1109/IROS.2013.6696848.

[72] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger. "Reactive Whole-Body Control: Dynamic Mobile Manipulation Using a Large Number of Actuated Degrees of Freedom". In: *IEEE Robotics and Automation Magazine* 19.2 (2012), pp. 20–33. ISSN: 1558-223X. DOI: 10.1109/MRA.2012.2191432.

[73] A. Dietrich, C. Ott, and A. Albu-Schäffer. "Multi-objective compliance control of redundant manipulators: Hierarchy, control, and stability". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2013, pp. 3043–3050. DOI: 10.1109/IROS.2013.6696787.

[74] A. Dietrich, X. Wu, K. Bussmann, C. Ott, A. Albu-Schäffer, and S. Stramigioli. "Passive Hierarchical Impedance Control Via Energy Tanks". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 522–529. ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2645504.

[75] G. Schreiber, C. Ott, and G. Hirzinger. "Interactive redundant robotics: control of the inverted pendulum with nullspace motion". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Vol. 1. 2001, 158–164 vol.1. DOI: 10.1109/IROS.2001.973352.

[76] Bojan Nemec, Leon Žlajpah, and Damir Omrčen. "Comparison of null-space and minimal null-space control algorithms". In: *Robotica* 25.5 (2007), pp. 511–520. DOI: 10.1017/S0263574707003402.

[77] J. Hollerbach and Ki Suh. "Redundancy resolution of manipulators through torque optimization". In: *IEEE Journal of Robotics and Automation* 3.4 (1987), pp. 308–316. ISSN: 0882-4967. DOI: 10.1109/JRA.1987.1087111.

[78] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics.* 2008. Aufl. Berlin Heidelberg: Springer Science & Business Media, 2008. ISBN: 978-3-540-23957-4.

[79] N. Hogan. "Impedance Control: An Approach to Manipulation". In: *Proc. American Control Conference (ACC).* 1984, pp. 304–313.

[80] T. Tomić and S. Haddadin. "A unified framework for external wrench estimation, interaction control and collision reflexes for flying robots". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2014, pp. 4197–4204. DOI: 10.1109/IROS.2014.6943154.

[81] A. de Luca and R. Mattone. "Sensorless Robot Collision Detection and Hybrid Force/Motion Control". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2005, pp. 999–1004. DOI: 10.1109/ROBOT.2005.1570247.

[82] S. Haddadin, A. De Luca, and A. Albu-Schäffer. "Robot Collisions: A Survey on Detection, Isolation, and Identification". In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1292–1312. ISSN: 1552-3098. DOI: 10.1109/TRO.2017.2723903.

[83] Burak Yüksel, Cristian Secchi, Heinrich H. Bülthoff, and Antonio Franchi. "Aerial Physical Interaction via IDA-PBC". In: *International Journal of Robotics Research* (2019).

[84] F. Augugliaro and R. D'Andrea. "Admittance control for physical human-quadrocopter interaction". In: *European Control Conference (ECC).* 2013, pp. 1805–1810. DOI: 10.23919/ECC.2013.6669643.

[85] C. D. McKinnon and A. P. Schoellig. "Unscented external force and torque estimation for quadrotors". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2016, pp. 5651–5657. DOI: 10.1109/IROS.2016.7759831.

[86] Jean-Jacques Slotine and Weiping Li. *Applied Nonlinear Control.* New York: Prentice-Hall, 1991. ISBN: 9780130408907.

[87] H.K. Khalil. *Nonlinear Systems.* New York: Prentice Hall, 2000. ISBN: 9780131227408.

[88] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control.* 1st ed. New York: New York, NY, USA: Wiley, 2005. ISBN: 978-0-471-64990-8.

[89] P.N. Paraskevopoulos, A.S. Tsirikos, and X. Koutsoukos. "Nonlinear Decoupling Control for a Robot Manipulator". In: *IFAC Proceedings Volumes* 31.20 (1998), pp. 653 –658. ISSN: 1474-6670.

[90] A.R. Meenakshi and C. Rajian. "On a product of positive semidefinite matrices". In: *Linear Algebra and its Applications* 295.1 (1999), pp. 3 –6. ISSN: 0024-3795. DOI: 10.1016/S0024-3795(99)00014-2.

[91] W. Durham, K.A. Bordignon, and R. Beck. *Aircraft Control Allocation.* Aerospace Series. Hoboken, NJ, USA: Wiley, 2016. ISBN: 9781118827772.

[92] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* Hoboken, NJ, USA: Wiley, 2011. ISBN: 1119991498.

[93] Farhad A. Goodarzi, Daewon Lee, and Taeyoung Lee. "Geometric nonlinear PID control of a quadrotor UAV on SE(3)". In: *European Control Conference (ECC).* 2013, pp. 3845–3850.

[94] S. Bouabdallah and R. Siegwart. "Full control of a quadrotor". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2007, pp. 153–158.

[95] D. Lee, H. Jin Kim, and S. Sastry. "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter". In: *International Journal of Control, Automation and Systems* 7 (2009), pp. 419–428. DOI: 10.1007/s12555-009-0311-8.

[96] D. Mellinger and V. Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.

[97] M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart. "Inversion based direct position control and trajectory following for micro aerial vehicles". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2013, pp. 2933–2939. DOI: 10.1109/IROS.2013.6696772.

[98] T. Tomić. "Evaluation of acceleration-based disturbance observation for multicopter control". In: *European Control Conference (ECC).* 2014, pp. 2937–2944. DOI: 10.1109/ECC.2014.6862237.

[99] M. Faessler, A. Franchi, and D. Scaramuzza. "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 620–626. DOI: 10.1109/LRA.2017.2776353.

[100] R. Mahony, V. Kumar, and P. Corke. "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor". In: *IEEE Robotics and Automation Magazine* 19.3 (2012), pp. 20–32. ISSN: 1070-9932. DOI: 10.1109/MRA.2012.2206474.

[101] T. Fernando, J. Chandiramani, T. Lee, and H. Gutierrez. "Robust adaptive geometric tracking controls on SO(3) with an application to the attitude dynamics of a quadrotor UAV". In: *IEEE Conference on Decision and Control (CDC).* 2011, pp. 7380–7385. DOI: 10.1109/CDC.2011.6161306.

[102] H. Demircioglu and H. I. Bastürk. “Adaptive attitude and altitude control of a quadrotor despite unknown wind disturbances”. In: *IEEE Conference on Decision and Control (CDC).* 2017, pp. 274–279. DOI: 10.1109/CDC.2017.8263678.

[103] M. Schreier. “Modeling and adaptive control of a quadrotor”. In: *IEEE International Conference on Mechatronics and Automation.* 2012, pp. 383–390. DOI: 10.1109/ICMA.2012.6282874.

[104] Ewoud J. J. Smeur, Guido C. H. E. de Croon, and Qiping Chu. “Cascaded Incremental Nonlinear Dynamic Inversion Control for MAV Disturbance Rejection”. In: *CoRR* abs/1701.07254 (2017). arXiv: 1701.07254. URL: http://arxiv.org/abs/1701.07254.

[105] E. Tal and S. Karaman. “Accurate Tracking of Aggressive Quadrotor Trajectories Using Incremental Nonlinear Dynamic Inversion and Differential Flatness”. In: *IEEE Conference on Decision and Control (CDC).* 2018, pp. 4282–4288. DOI: 10.1109/CDC.2018.8619621.

[106] Moses Bangura and Robert Mahony. “Real-time Model Predictive Control for Quadrotors”. In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 11773 –11780. ISSN: 1474-6670. DOI: 10.3182/20140824-6-ZA-1003.00203.

[107] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza. “PAMPC: Perception-Aware Model Predictive Control for Quadrotors”. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2018, pp. 1–8. DOI: 10.1109/IROS.2018.8593739.

[108] Steven L. Waslander and Carlos Wang. “Wind Disturbance Estimation and Rejection for Quadrotor Position Control”. In: *AIAA Infotech Aerospace Conference.* 2009, pp. 1–14. DOI: http://dx.doi.org/10.2514/6.2009-1983.

[109] T. Tomić, K. Schmid, P. Lutz, A. Mathers, and S. Haddadin. “The flying anemometer: Unified estimation of wind velocity from aerodynamic power and wrenches”. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2016, pp. 1637–1644. DOI: 10.1109/IROS.2016.7759264.

[110] S. Bellens, J. De Schutter, and H. Bruyninckx. “A hybrid pose / wrench control framework for quadrotor helicopters”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2012, pp. 2269–2274.

[111] F. Ruggiero, J. Cacace, H. Sadeghian, and V. Lippiello. “Impedance control of VToL UAVs with a momentum-based external generalized forces estimator”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2014, pp. 2093–2099.

[112] T. Tomić, C. Ott, and S. Haddadin. “External Wrench Estimation, Collision Detection, and Reflex Reaction for Flying Robots”. In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1467–1482. ISSN: 1552-3098. DOI: 10.1109/TRO.2017.2750703.

[113] T. Tomić and S. Haddadin. “Simultaneous estimation of aerodynamic and contact forces in flying robots: Applications to metric wind estimation and collision detection”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2015, pp. 5290–5296. DOI: 10.1109/ICRA.2015.7139937.

[114] J. Gordon Leishman. *Principles of Helicopter Aerodynamics*. 2nd ed. New York, NY, USA: Cambridge University Press, 2006. ISBN: 1107013356.

[115] Tor A. Johansen and Thor I. Fossen. "Control allocation - A survey". In: *Automatica* 49.5 (2013), pp. 1087 –1103. ISSN: 0005-1098. DOI: doi.org/10.1016/j.automatica.2013.01.035.

[116] Thomas Raffler, Jian Wang, and Florian Holzapfel. "Path Generation and Control for Unmanned Multirotor Vehicles Using Nonlinear Dynamic Inversion and Pseudo Control Hedging". In: *IFAC Proceedings Volumes* 46.19 (2013). 19th IFAC Symposium on Automatic Control in Aerospace, pp. 194 –199. ISSN: 1474-6670. DOI: 10.3182/20130902-5-DE-2040.00132.

[117] Eric N. Johnson and Anthony J. Calise. *Pseudo-Control Hedging: A New Method For Adaptive Control*. 2000.

[118] M. Faessler, D. Falanga, and D. Scaramuzza. "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 476–482. ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2640362.

[119] Ewoud Smeur, Daan Höppener, and Christophe De Wagter. "Prioritized Control Allocation for Quadrotors Subject to Saturation". In: *International Micro Air Vehicle Conference and Flight Competition*. Ed. by H. de Plinval J.-M. Moschetta G. Hattenberger. Toulouse, France, 2017, pp. 37–43.

[120] O. Härkegård. "Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation". In: *IEEE Conference on Decision and Control (CDC)*. Jan. 2003, 1295 –1300 vol.2. ISBN: 0-7803-7516-5. DOI: 10.1109/CDC.2002.1184694.

[121] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran. "Continuous-time trajectory optimization for online UAV replanning". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 5332–5339. DOI: 10.1109/IROS.2016.7759784.

[122] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers. "Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 215–222.

[123] A. Bry, A. Bachrach, and N. Roy. "State estimation for aggressive flight in GPS-denied environments using onboard sensing". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2012, pp. 1–8. DOI: 10.1109/ICRA.2012.6225295.

[124] M.W. Müller and R. D'Andrea. "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 45–52. DOI: 10.1109/ICRA.2014.6906588.

[125] Peter H. Zipfel. *Modeling and Simulation of Aerospace Vehicle Dynamics -*. Stanford University: American Institute of Aeronautics and Astronautics, 2000. ISBN: 978-1-563-47456-9.

[126] K. Kondak, M. Bernard, N. Meyer, and G. Hommel. "Autonomously Flying VTOL-Robots: Modeling and Control". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2007, pp. 736–741. DOI: 10.1109/ROBOT.2007.363074.

[127] M. Bangura and R. Mahony. "Thrust Control for Multirotor Aerial Vehicles". In: *IEEE Transactions on Robotics* 33.2 (2017), pp. 390–405. ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2633562.

[128] Dávid Rau, Jozef Rodina, Lukáš Palkovič, and Peter Hubinský. "Sensorless field oriented control of BLDC motors for MAVS". In: *Transactions on Electrical Engineering* 4 (2015), pp. 91–96.

[129] *UM2516 User manual - Electronic speed controller Discovery kit for drones*. Rev. 2. STMicroelectronics. 2020. URL: https://www.st.com/resource/en/user_manual/dm00564746-electronic-speed-controller-discovery-kit-for-drones-with-stm32g431cb-stmicroelectronics.pdf.

[130] *The AutoQuad ESC32 - advanced Electronic Speed Controller*. online. [Online; accessed October 14, 2019]. 2019. URL: http://autoquad.org/esc32/?lang=en.

[131] A. Franchi and A. Mallet. "Adaptive closed-loop speed control of BLDC motors with applications to multi-rotor aerial vehicles". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 5203–5208. DOI: 10.1109/ICRA.2017.7989610.

[132] M. Podhradský, J. Bone, C. Coopmans, and A. Jensen. "Battery model-based thrust controller for a small, low cost multirotor Unmanned Aerial Vehicles". In: *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2013, pp. 105–113. DOI: 10.1109/ICUAS.2013.6564679.

[133] Joel H. Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics*. 3rd. Berlin Heidelberg: Springer Science & Business Media, 2012. ISBN: 978-3-642-56026-2.

[134] Gareth D. Padfield. *Helicopter Flight Dynamics - The Theory and Application of Flying Qualities and Simulation Modelling*. New York: John Wiley & Sons, 2008. ISBN: 978-0-470-69116-8.

[135] R. Gill and R. D'Andrea. "Propeller thrust and drag in forward flight". In: *IEEE Conference on Control Technology and Applications (CCTA)*. 2017, pp. 73–79. DOI: 10.1109/CCTA.2017.8062443.

[136] M. Bangura, H. Lim, H. J. Kim, and R. Mahony. "Aerodynamic power control for multirotor aerial vehicles". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 529–536. DOI: 10.1109/ICRA.2014.6906906.

[137] Pedro Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero. "Characterization of the Aerodynamic Ground Effect and Its Influence in Multirotor Control". In: *International Journal of Aerospace Engineering* (2017), p. 17. DOI: 10.1155/2017/1823056.

[138] Davide Del Cont Bernard, Fabio Riccardi, Mattia Giurato, and Marco Lovera. "A dynamic analysis of ground effect for a quadrotor platform". In: vol. 50. 2017, p. 6. DOI: 10.1016/Journalifacol.2017.08.1500.

[139] E. Davis and P. E. I. Pounds. "Passive Position Control of a Quadrotor With Ground Effect Interaction". In: *IEEE Robotics and Automation Letters (RA-L)* 1.1 (2016), pp. 539–545. DOI: 10.1109/LRA.2016.2514351.

[140] John M. Wallace and Peter V. Hobbs. *Atmospheric Science - An Introductory Survey.* 2. ed. Amsterdam: Elsevier, 2006. ISBN: 978-0-080-49953-6.

[141] M. G. Lawrence. "The Relationship between Relative Humidity and the Dewpoint Temperature in Moist Air: A Simple Conversion and Applications." In: *Bulletin of the American Meteorological Society* 86 (Feb. 2005), pp. 225–233. DOI: 10.1175/BAMS-86-2-225.

[142] Guowei Cai, Ben M. Chen, and Tong Heng Lee. *Unmanned Rotorcraft Systems.* 2011 ed. Berlin Heidelberg: Springer Science & Business Media, 2011. ISBN: 978-0-857-29635-1.

[143] *BME280 - Data sheet.* 1.6. Bosch Sensortec. 2018. URL: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280-DS002.pdf.

[144] Ira H. A. Abbott. *Theory of Wing Sections: Including a Summary of Airfoil Data.* New York, NY, USA: Dover Publications, 1959.

[145] G. Jiang and R. Voyles. "Hexrotor UAV platform enabling dextrous interaction with structures-flight test". In: *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR).* 2013, pp. 1–6. DOI: 10.1109/SSRR.2013.6719377.

[146] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, and A. Franchi. "6D physical interaction with a fully actuated aerial robot". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2017, pp. 5190–5195. DOI: 10.1109/ICRA.2017.7989608.

[147] Mark Cutler and Jonathan P. How. "Analysis and Control of a Variable-Pitch Quadrotor for Agile Flight". In: *ASME Journal of Dynamic Systems, Measurement and Control* 137.10 (2015), p. 14. DOI: 10.1115/1.4030676.

[148] D. Brescianini and R. D'Andrea. "Design, modeling and control of an omni-directional aerial vehicle". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2016, pp. 3261–3266. DOI: 10.1109/ICRA.2016.7487497.

[149] M. Ryll, H. H. Bülthoff, and P. R. Giordano. "A Novel Overactuated Quadrotor Unmanned Aerial Vehicle: Modeling, Control, and Experimental Validation". In: *IEEE Transactions on Control Systems Technology* 23.2 (2015), pp. 540–556. ISSN: 1063-6536. DOI: 10.1109/TCST.2014.2330999.

[150] M. Kamel, S. Verling, O. Elkhatib, C. Sprecher, P. Wulkop, Z. Taylor, R. Siegwart, and I. Gilitschenski. "The Voliro Omniorientational Hexacopter: An Agile and Maneuverable Tiltable-Rotor Aerial Vehicle". In: *IEEE Robotics and Automation Magazine* 25.4 (2018), pp. 34–44. ISSN: 1070-9932. DOI: 10.1109/MRA.2018.2866758.

[151] N. Gupta, M. Kothari, and Abhishek. "Flight dynamics and nonlinear control design for variable-pitch quadrotors". In: *Proc. American Control Conference (ACC)*. 2016, pp. 3150–3155. DOI: 10.1109/ACC.2016.7525402.

[152] Namrata Gupta, Mangal Kothari, and Abhishek. "Modeling and Control of Inverted Flight of a Variable-Pitch Quadrotor". In: *CoRR* abs/1709.06407 (2017). arXiv: 1709.06407. URL: http://arxiv.org/abs/1709.06407.

[153] Bernard Mettler. *Identification Modeling and Characteristics of Miniature Rotorcraft.* New York, NY, USA: Springer, 2010. ISBN: 978-1-4419-5311-7.

[154] Ioannis A. Raptis and Kimon P. Valavanis. *Linear and Nonlinear Control of Small-Scale Unmanned Helicopters.* 2011. ed. Berlin Heidelberg: Springer Science & Business Media, 2010. ISBN: 978-9-400-70023-9.

[155] Jorge Nocedal and Stephen Wright. *Numerical Optimization.* 2nd ed. Berlin Heidelberg: Springer Science & Business Media, 2006. ISBN: 978-0-387-40065-5.

[156] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. "qpOASES: A parametric active-set algorithm for quadratic programming". In: *Mathematical Programming Computation* 6.4 (2014), pp. 327–363.

[157] P. Foehn and D. Scaramuzza. "Onboard State Dependent LQR for Agile Quadrotors". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 6566–6572. DOI: 10.1109/ICRA.2018.8460885.

[158] M. W. Spong, R. Ortega, and R. Kelly. "Comments on "Adaptive manipulator control: a case study" by J. Slotine and W. Li". In: *IEEE Transactions on Automatic Control* 35.6 (1990), pp. 761–762. ISSN: 2334-3303. DOI: 10.1109/9.53565.

[159] James B. Carrell. *Groups, Matrices, and Vector Spaces - A Group Theoretic Approach to Linear Algebra.* 1st ed. 2017. Berlin, Heidelberg: Springer, 2017. ISBN: 978-0-387-79428-0.

[160] R. Ortega and M. W. Spong. "Adaptive motion control of rigid robots: a tutorial". In: *IEEE Conference on Decision and Control (CDC)*. 1988, 1575–1584 vol.2. DOI: 10.1109/CDC.1988.194594.

[161] J. Dougherty, D. Lee, and T. Lee. "Laser-based guidance of a quadrotor uav for precise landing on an inclined surface". In: *Proc. American Control Conference (ACC)*. 2014, pp. 1210–1215. DOI: 10.1109/ACC.2014.6859391.

[162] M. W. Mueller, M. Hehn, and R. D'Andrea. "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation". In: *IEEE Transactions on Robotics* 31.6 (2015), pp. 1294–1310. ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2479878.

[163] A. V. Rao, D. A. Benson, C. L. Darby, M. A. Patterson, C. Francolin, I. Sanders, and G. T. Huntington. "Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method". In: *ACM Transactions on Mathematical Software* 37.2 (2010), p. 39.

[164] H.J. Ferreau, H.G. Bock, and M. Diehl. "An online active set strategy to overcome the limitations of explicit MPC". In: *International Journal of Robust and Nonlinear Control* 18.8 (2008), pp. 816–830.

[165] P.E.I. Pounds and A.M. Dollar. "Stability of Helicopters in Compliant Contact Under PD-PID Control". In: *IEEE Transactions on Robotics* 30.6 (2014), pp. 1472–1486. ISSN: 1552-3098. DOI: 10.1109/TRO.2014.2363371.

[166] Richard C. Dorf and Robert H. Bishop. *Modern Control Systems*. 12 ed. Harlow: Pearson Education Limited, 2011. ISBN: 978-0-13-602458-3.

[167] Otto Föllinger. *Regelungstechnik*. 11th ed. Berlin, Offenbach: Vde Verlag GmbH, 2013. ISBN: 978-3-8007-3231-9.

[168] G. Roppenecker. "On parametric state feedback design". In: *International Journal of Control* 43.3 (1986), pp. 793–804. DOI: 10.1080/00207178608933503.

[169] Frederick Walker Fairman. *Linear Control Theory: The State Space Approach*. Hoboken, NJ, USA: Wiley, 1998. ISBN: 9780471974895.

[170] Marcel J. Sidi. *Spacecraft Dynamics and Control: A Practical Engineering Approach*. New York, NY, USA: Cambridge University Press, 1997. ISBN: 9780521550727.

[171] Pierre-Jean Bristeau, François Callou, David Vissière, and Nicolas Petit. "The Navigation and Control Technology Inside the AR.Drone Micro UAV". In: *18th IFAC World Congress*. 2011, pp. 1477–1484.

[172] Moritz Maier, André Oeschger, and Konstantin Kondak. *Robot-assisted Landing of VTOL UAVs*. https://youtu.be/r6_uVIhORAg. [Online; accessed Nov. 12, 2015]. German Aerospace Center (DLR), Institute of Robotics and Mechatronics, 2015.

[173] P. De Monte and B. Lohmann. "Trajectory tracking control for a quadrotor helicopter based on backstepping using a decoupling quaternion parametrization". In: *Proc. Mediterranean Conference on Control and Automation (MED)*. 2013, pp. 507–512. DOI: 10.1109/MED.2013.6608769.

[174] S. Bhat and D.S. Bernstein. "A topological obstruction to global asymptotic stabilization of rotational motion and the unwinding phenomenon". In: *Proc. American Control Conference (ACC)*. Vol. 5. 1998, pp. 2785–2789. DOI: 10.1109/ACC.1998.688361.

[175] R. Kristiansen, P.J. Nicklasson, and J.T. Gravdahl. "Satellite Attitude Control by Quaternion-Based Backstepping". In: *IEEE Transactions on Control Systems Technology* 17.1 (2009), pp. 227–232. DOI: 10.1109/tcst.2008.924576.

[176] Bernard Mettler, Chris Dever, and Eric Feron. "Scaling effects and dynamic characteristics of miniature rotorcraft". In: *Journal of Guidance, Control, and Dynamics* 27.3 (2004), pp. 466–478.

[177] Alexander Dietrich, Kristin Bussmann, Florian Petit, Paul Kotyczka, Christian Ott, Boris Lohmann, and Alin Albu-Schäffer. "Whole-body impedance control of wheeled mobile manipulators". In: *Autonomous Robots* 40.3 (2016), pp. 505–517. ISSN: 1573-7527. DOI: 10.1007/s10514-015-9438-z. URL: 10.1007/s10514-015-9438-z.

[178] Francesco Bullo and Richard M. Murray. *Proportional Derivative (PD) Control on the Euclidean Group*. Tech. rep. California Institute of Technology, 1995.

[179] Gianluca Garofalo, Bernd Henze, Johannes Englsberger, and Christian Ott. “On the inertially decoupled structure of the floating base robot dynamics”. In: *IFAC-PapersOnLine* 48.1 (2015). 8th Vienna International Conferenceon Mathematical Modelling, pp. 322 –327. ISSN: 2405-8963. DOI: 10.1016/Journalifacol.2015.05.189. URL: http://www.sciencedirect.com/science/article/pii/S2405896315001901.

[180] F. Beck, G. Garofalo, and C. Ott. “Vibration Control for Manipulators on a Translationally Flexible Base”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2019, pp. 4451–4457. DOI: 10.1109/ICRA.2019.8793904.

[181] D. Lee. “Passive Decomposition and Control of Nonholonomic Mechanical Systems”. In: *IEEE Transactions on Robotics* 27.1 (2011), pp. 978–992. ISSN: 1941-0468. DOI: 10.1109/TRO.2011.2159425.

[182] C. W. Wampler. “Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 16.1 (1986), pp. 93–101. ISSN: 0018-9472. DOI: 10.1109/TSMC.1986.289285.

[183] C. Gaz, F. Flacco, and A. De Luca. “Identifying the dynamic model used by the KUKA LWR: A reverse engineering approach”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 2014, pp. 1386–1392. DOI: 10.1109/ICRA.2014.6907033.

[184] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* 1st ed. Nashua, U.S.A.: Athena Scientific, 1996. ISBN: 1886529043.

[185] I. A. Gravagne and I. D. Walker. “On the structure of minimum effort solutions with application to kinematic redundancy resolution”. In: *IEEE Transactions on Robotics and Automation* 16.6 (2000), pp. 855–863. ISSN: 2374-958X. DOI: 10.1109/70.897797.

[186] T. I. Fossen and S. I. Sagatun. “Adaptive control of nonlinear underwater robotic systems”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 1991, 1687–1694 vol.2. DOI: 10.1109/ROBOT.1991.131862.

[187] M. Triantafyllou, M. Bodson, and M. Athans. “Real time estimation of ship motions using Kalman filtering techniques”. In: *IEEE Journal of Oceanic Engineering* 8.1 (1983), pp. 9–20. ISSN: 2373-7786. DOI: 10.1109/JOE.1983.1145542.

[188] S. Küchler, J.K. Eberharter, K. Langer, K. Schneider, and O. Sawodny. “Heave Motion Estimation of a Vessel Using Acceleration Measurements”. In: *IFAC Proceedings Volumes* 44.1 (2011). 18th IFAC World Congress, pp. 14742 –14747. ISSN: 1474-6670. DOI: 10.3182/20110828-6-IT-1002.01935.

[189] B. Wheeler, A. Ng, B. Kilberg, F. Maksimovic, and K. S. J. Pister. “A Low-Power Optical Receiver for Contact-free Programming and 3D Localization of Autonomous Microsystems”. In: *Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON).* 2019, pp. 0371–0376. DOI: 10.1109/UEMCON47517.2019.8992964.

[190] F. Lazzari, A. Buffi, P. Nepa, and S. Lazzari. "Numerical Investigation of an UWB Localization Technique for Unmanned Aerial Vehicles in Outdoor Scenarios". In: *IEEE Sensors Journal* 17.9 (2017), pp. 2896–2903. ISSN: 2379-9153. DOI: 10.1109/JSEN.2017.2684817.

[191] Artur Sagitov, Ksenia Shabalina, Leysan Sabirova, Hongbing Li, and Evgeni Magid. "ARTag, AprilTag and CALTag Fiducial Marker Systems: Comparison in a Presence of Partial Marker Occlusion and Rotation". In: *International Conference on Informatics in Control, Automation and Robotics (ICINCO).* Jan. 2017, pp. 182–191. DOI: 10.5220/0006478901820191.

[192] David Held, Sebastian Thrun, and Silvio Savarese. "Learning to Track at 100 FPS with Deep Regression Networks". In: *European Conference on Computer Vision (ECCV).* Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 749–765. ISBN: 978-3-319-46448-0.

[193] Daniel Wagner and Dieter Schmalstieg. "Artoolkitplus for pose tracking on mobile devices". In: *Proceedings of 12th Computer Vision Winter Workshop (CVWW).* 2007, pp. 139–146.

[194] J. Wang and E. Olson. "AprilTag 2: Efficient and robust fiducial detection". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2016, pp. 4193–4198. DOI: 10.1109/IROS.2016.7759617.

[195] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[196] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* Cambridge, MA, USA: The MIT Press, 2005. ISBN: 0262201623.

[197] Switzerland Hubert Kirrmann Solutil and William Dickerson. "Precision Time Protocol profile for power utility automation Application". In: *Pacworld* (2016). URL: https://www.pacw.org/issue/september_2016_issue/iecieee.html.

[198] Kaiqiang Feng, Jie Li, Xiaoming Zhang, Chong Shen, Yu Bi, Tao Zheng, and Jun Liu. "A New Quaternion-Based Kalman Filter for Real-Time Attitude Estimation Using the Two-Step Geometrically-Intuitive Correction Algorithm". In: *Sensors* 17.9 (2017), pp. 1–21. DOI: 10.3390/s17092146.

[199] S. Hutchinson, G. D. Hager, and P. I. Corke. "A tutorial on visual servo control". In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 651–670. ISSN: 2374-958X. DOI: 10.1109/70.538972.

[200] F. Chaumette and S. Hutchinson. "Visual servo control. I. Basic approaches". In: *IEEE Robotics and Automation Magazine* 13.4 (2006), pp. 82–90. ISSN: 1558-223X. DOI: 10.1109/MRA.2006.250573.

[201] F. Chaumette and S. Hutchinson. "Visual servo control. II. Advanced approaches [Tutorial]". In: *IEEE Robotics and Automation Magazine* 14.1 (2007), pp. 109–118. ISSN: 1558-223X. DOI: 10.1109/MRA.2007.339609.

[202] Tsuneo Yoshikawa. “Manipulability of Robotic Mechanisms”. In: *The International Journal of Robotics Research* 4.2 (1985), pp. 3–9. DOI: 10.1177/027836498500400201.

[203] T. Yoshikawa. “Translational and Rotational Manipulability of Robotic Manipulators”. In: *Proc. American Control Conference (ACC)*. 1990, pp. 228–233. DOI: 10.23919/ACC.1990.4790733.

[204] Jihong Lee. “A study on the manipulability measures for robot manipulators”. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 3. 1997, 1458–1465 vol.3. DOI: 10.1109/IROS.1997.656551.

[205] Pasquale Chiacchio, Yann Bouffard-Vercelli, and François Pierrot. “Force polytope and force ellipsoid for redundant manipulators”. In: *Journal of Robotic Systems* 14.8 (1997), pp. 613–620. DOI: 10.1002/(SICI)1097-4563(199708)14:8<613::AID-ROB3>3.0.CO;2-P.

[206] T. Hulin, A. Albu-Schäffer, and G. Hirzinger. “Passivity and Stability Boundaries for Haptic Systems With Time Delay”. In: *IEEE Transactions on Control Systems Technology* 22.4 (2014), pp. 1297–1309. ISSN: 2374-0159. DOI: 10.1109/TCST.2013.2283372.

[207] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli. “Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1398–1404. DOI: 10.1109/ICRA.2016.7487274.

Publications

Journals

[208] Moritz Maier, Andre Oeschger, and Konstantin Kondak. "Robot-assisted Landing of VTOL UAVs: Design and Comparison of Coupled and Decoupling Linear State-space Control Approaches". In: *IEEE Robotics and Automation Letters (RA-L)* 1.1 (2016), pp. 114–121. DOI: `10.1109/LRA.2015.2502920`.

[209] Moritz Maier, Manuel Keppler, Christian Ott, and Alin Albu-Schäffer. "Adaptive Air Density Estimation for Precise Tracking Control and Accurate External Wrench Observation for Flying Robots". In: *IEEE Robotics and Automation Letters (RA-L)* 5.2 (2020), pp. 1445–1452. DOI: `10.1109/LRA.2020.2967333`.

[210] Philipp Lutz, Marcus G. Müller, Moritz Maier, Samantha Stoneman, Teodor Tomić, Ingo von Bargen, Martin Schuster, Florian Steidle, Armin Wedler, Wolfgang Stürzl, and Rudolph Triebel. "ARDEA - An MAV with skills for future planetary missions". In: *Journal of Field Robotics (JFR), Special Issue on Future Challenges and Opportunities in Vision-based Drone Navigation* (2019). DOI: `10.1002/rob.21949`.

[211] Martin J. Schuster, Marcus G. Müller, Sebastian G. Brunner, Hannah Lehner, Peter Lehner, Ryo Sakagami, Andreas Dömel, Lukas Meyer, Bernhard Vodermayer, Riccardo Giubilato, Mallikarjuna Vayugundla, Josef Reill, Florian Steidle, Ingo von Bargen, Kristin Bussmann, Rico Belder, Philipp Lutz, Wolfgang Stürzl, Michal Smíšek, Moritz Maier, Samantha Stoneman, Andre Fonseca Prince, Bernhard Rebele, Maximilian Durner, Emanuel Staudinger, Siwei Zhang, Robert Pohlmann, Esther Bischoff, Christian Braun, Susanne Schröder, Enrico Dietz, Sven Frohmann, Anko Börner, Heinz-Wilhelm Hübers, Bernard Foing, Rudolph Triebel, Alin O. Albu-Schäffer, and Armin Wedler. "The ARCHES Space-Analogue Demonstration Mission: Towards Heterogeneous Teams of Autonomous Robots for Collaborative Scientific Sampling in Planetary Exploration". In: *IEEE Robotics and Automation Letters (RA-L)* (2020). DOI: `10.1109/LRA.2020.3007468`.

Conferences

[212] Moritz Maier and Konstantin Kondak. "Robot assisted landing of VTOL UAVs on ships: A simulation case study of the touch-down phase". In: *IEEE Conference on Control Technology and Applications (CCTA)*. Kohala Coast, Hawai'i, 2017, pp. 2094–2101. DOI: `10.1109/CCTA.2017.8062762`.

[213] Moritz Maier and Konstantin Kondak. “Landing of VTOL UAVs using a Stationary Robot Manipulator: A new Approach for Coordinated Control”. In: *IEEE Conference on Decision and Control (CDC)*. Osaka, Japan, 2015, pp. 1497–1502. DOI: 10.1109/CDC.2015.7402422.

[214] Moritz Maier, Konstantin Kondak, and Christian Ott. “Enabling robot assisted landing of heavy UAV rotorcraft via combined control and workload sharing”. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, Canada, 2017, pp. 5128–5134. DOI: 10.1109/IROS.2017.8206399.

[215] Moritz Maier. “Bidirectional Thrust for Multirotor MAVs with Fixed-Pitch Propellers”. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, pp. 1–8. DOI: 10.1109/IROS.2018.8593836.

[216] Louis T. Tcheumadjeu, F. Andert, Q. Tang, A. Sohr, R. Kaul, J. Belz, P. Lutz, M. Maier, M. G. Müller, and W. Stürzl. “Integration of an Automated Valet Parking Service into an Internet of Things Platform”. In: *International Conference on Intelligent Transportation Systems (ITSC)*. Maui, Hawai’i, 2018, pp. 662–668. DOI: 10.1109/ITSC.2018.8569230.

[217] Marcus G. Müller, F. Steidle, M. J. Schuster, P. Lutz, M. Maier, S. Stoneman, T. Tomić, and W. Stürzl. “Robust Visual-Inertial State Estimation with Multiple Odometries and Efficient Mapping on an MAV with Ultra-Wide FOV Stereo Vision”. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, pp. 3701–3708. DOI: 10.1109/IROS.2018.8594117.

[218] Roman Weitschat, Jan Ehrensperger, Moritz Maier, and Harald Aschemann. “Safe and Efficient Human-Robot Collaboration Part I: Estimation of Human Arm Motions”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia, 2018, pp. 1993–1999. DOI: 10.1109/ICRA.2018.8461190.

[219] Teodor Tomić, Moritz Maier, and Sami Haddadin. “Learning quadrotor maneuvers from optimal control and generalizing in real-time”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, 2014, pp. 1747–1754. DOI: 10.1109/icra.2014.6907087.

Diploma thesis

[220] Moritz Maier. “Learning Quadrotor Maneuvers from Optimal Control”. Diploma thesis. Technical University of Munich, 2013. URL: https://elib.dlr.de/85798/.

Patents

[221] Moritz Maier. “Cleaning device and method for cleaning a surface”. English. German pat. DE102018115818B4. Mar. 2020.

[222] Moritz Maier. “Determination of an air density by an aircraft”. English. German pat. DE102019131398A1. Oct. 2020.