Imperial College London
Department of Computing

Safe and Accurate MAV Control, Navigation and Manipulation

Dimosthenis Tzoumanikas

29th November 2020

Supervised by Dr Stefan Leutenegger

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the Diploma of Imperial College London. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Abstract

This work focuses on the problem of precise, aggressive and safe Micro Aerial Vehicle (MAV) navigation as well as deployment in applications which require physical interaction with the environment.

To address these issues, we propose three different MAV model based control algorithms that rely on the concept of receding horizon control. As a starting point, we present a computationally cheap algorithm which utilises an approximate linear model of the system around hover and is thus maximally accurate for slow reference maneuvers. Aiming at overcoming the limitations of the linear model parameterisation, we present an extension to the first controller which relies on the true nonlinear dynamics of the system. This approach, even though computationally more intense, ensures that the control model is always valid and allows tracking of full state aggressive trajectories. The last controller addresses the topic of aerial manipulation in which the versatility of aerial vehicles is combined with the manipulation capabilities of robotic arms. The proposed method relies on the formulation of a hybrid nonlinear MAV-arm model which also takes into account the effects of contact with the environment. Finally, in order to enable safe operation despite the potential loss of an actuator, we propose a supervisory algorithm which estimates the health status of each motor. We further showcase how this can be used in conjunction with the nonlinear controllers described above for fault-tolerant MAV flight.

While all the developed algorithms are formulated and tested using our specific MAV platforms (consisting of underactuated hexacopters for the free flight experiments, hexacopter-delta arm system for the manipulation experiments), we further discuss how these can be applied to other underactuated/overactuated MAVs and robotic arm platforms. The same applies to the fault tolerant control where we discuss different stabilisation techniques depending on the capabilities of the available hardware.

Even though the primary focus of this work is on feedback control, we thoroughly describe the custom hardware platforms used for the experimental evaluation, the state estimation algorithms which provide the basis for control as well as the parameter identification required for the formulation of the various control models.

We showcase all the developed algorithms in experimental scenarios designed to highlight the corresponding strengths and weaknesses as well as show that the proposed methods can run in realtime on commercially available hardware.

Acknowledgements

A big part of my academic life has come to an end. I would like to thank those who helped me and inspired me throughout these years.

I would like to begin by thanking my supervisor Dr Stefan Leutenegger for providing me with the opportunity to pursue a PhD under his guidance. Stefan has been extremely supportive and generous with sharing his time and knowledge. He gave me the freedom to follow research topics I found interesting whilst being there to give his input when required. Special thanks also go to my co-supervisor Prof. Andrew Davison for his overall support, advice and encouragement during these years. Since day one, Stefan and Andy have supplied an easygoing and inspiring working environment while they both acted as academic role models for us younger researchers.

As a member of the Smart Robotics Lab it was an honor working with, sharing concerns and joys with the past and present members of the group: Wenbin Li, Binbin Xu, Nils Funk, Chris Choi, Sotiris Papatheodorou and Masha Popovic; all the MSc/UROP students who contributed to the experiments: Felix Graule, Marius Grimm and Qingyue Yan; as well as the many excellent researchers from the Dyson Robotics Lab including: Michael Bloesch, Sajaad Saeedi, Ronnald Clark, John McCormac, Andrea Nicastro, Jan Czarnowski and Tristan Laidlow.

I would also like to thank Iosifina Pournara and Amani El-Kholy for organising everything related to my PhD studies as well as the Engineering and Physical Sciences Research Council (EPSRC) and Imperial College London for funding this research.

Personally, I owe my deepest thanks to my parents, Petros, Panagiota, my sisters Athena and Georgia and my partner in life Naya who throughout these years have always stood by my side and supported me in every way possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Motivation

During the past years, MAVs have obtained significant popularity in the robotics community and have attracted the attention of both the academic and industrial world. The latest technological advancements in areas ranging from battery and computer systems design to the software-related engineering optimisation, computer vision and automatic control have enabled the cheap and successful MAV deployment in civil applications. Among those are: (i) search and rescue [Doherty and Rudol,

2007], (ii) agricultural monitoring [Zhang and Kovacs, 2012, Sa et al., 2018b], (iii) infrastructure inspection [Liu et al., 2014, Ozaslan et al., 2017] and (iv) parcel delivery [Rose, 2013, Dorling et al., 2017].

From a research perspective, development of aerial vehicles and their deployment in the unstructured real world, requires combination of multiple scientific topics, ranging from the actual hardware design to the software related algorithms that enable autonomous navigation. In this work, we discuss topics related to: (i) platform design, (ii) state estimation and (iii) control.

Particularly, a focus on MAV control will be given, with the emphasis on: (i) the design and implementation of algorithms which exploits the specific system's capabilities and constraints, (ii) the robust operation under actuator faults and (iii) the integration of physical interaction capabilities. Regarding the state estimation, we mainly rely on existing works appropriately re-purposed to fit our needs and sensor setups while the same applies to the platform design, where we use already proposed platforms to showcase our algorithms.

1.2 Background

In this Section we provide a brief overview of the research topics discussed in this work in order to set the scene for the contributions listed in the next Section. A more specific comparison of the contributions of this work with related research is provided in the individual Chapters.

1.2.1 Platform design

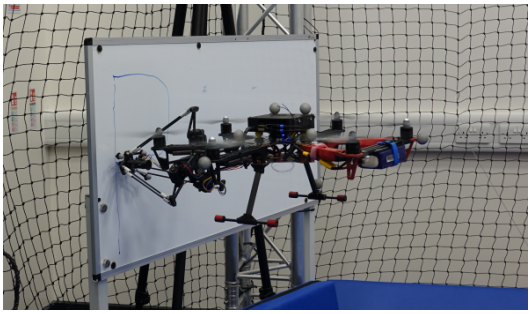
Depending on the target application numerous different platform designs have been proposed, ranging from simple quadrotors [Castillo et al., 2003], multirotors with non collinear motors [Rajappa et al., 2015, Brescianini and D'Andrea, 2016], more complex designs which actively control the per-motor thrust direction [Papachristos et al., 2016, Ryll et al., 2016, Kamel et al., 2018] or even platforms which can –on command– adapt their morphology [Riviere et al., 2018, Falanga et al., 2019, Bucki and Mueller, 2019].

Since the main focus of this work in on the software side and not the platform design by itself, we adopt the most common type of multirotors, those equipped with collinear motors and fixed pitch propellers. Control of these underactuated vehicles,

(a) Tracking and landing on a moving target. Two cameras were used for localisation and target detection. [Tzoumanikas et al., 2019]

(b) Fault detection and recovery despite the loss of an actuator. MAV maintains control of position and yaw. [Tzoumanikas et al., 2020b]

(c) Indoor exploration of an unknown area using a depth sensor for online mapping. [Dai et al., 2020]



(d) Joint MAV-aerial manipulator control for tasks including contact. [Tzoumanikas et al., 2020a]

Figure 1.1: Four snapshots of our MAVs performing autonomous missions.

is solely achieved by varying the speed of each motor which results in a change of the produced thrust and torque. A few illustrative examples of our MAVs executing autonomous missions are shown in Figure 1.1.

Regarding the required payload such as onboard sensors and computers which is also part of the platform design, we use commercially available products. When this is not possible e.g. the MAV mounted robotic arm for interaction tasks, we use already proposed in literature platforms modified to meet our needs.

1.2.2 Model based control

In early MAV control works such as [Castillo et al., 2003, Bouabdallah et al., 2004, Bouabdallah, 2007] simple Proportional Integral Derivative (PID) or Linear Quadratic Regulator (LQR) controllers based on linearised system models were proposed. These works relied on using Euler angles as a parameterisation for rotation. Aiming at overcoming the singularities introduced by the use of Euler angles and

using the nonlinear MAV model, [Lee et al., 2010] proposed a geometric tracking controller operating on $SE(3)$ which is nowadays one of the commonly used control approaches as it has been successfully deployed with appropriate modifications (e.g. including extra feedforward and integral terms) in various different vehicles and applications with examples in [Mellinger and Kumar, 2011, Rajappa et al., 2015, Richter et al., 2016, Invernizzi and Lovera, 2017, Bodie et al., 2018, Ryll et al., 2019].

While the aforementioned approaches rely on some MAV model they lack the capability of explicitly handling specific state, input or actuator constraints. An indirect way of ensuring minimum state constraint violation (e.g. maximum linear velocity) is by providing the controller a constraint-safe trajectory [Achtelik et al., 2013] obtained by a separate path planning module. Regarding infeasible input or actuator commands, this is usually handled by re-projecting them onto the boundary of the admissible input set [Schneider et al., 2012, Brescianini and D'Andrea, 2018].

A generic model based approach, also capable of handling constraints, adopted in this work is Model Predictive Controller (MPC) [Garcia et al., 1989, Morari and Lee, 1999, Findeisen and Allgöwer, 2002]. Briefly, the control input is computed by minimizing a cost function which is a metric of the system performance over a time horizon. The future system state is predicted by forward simulation using the latest system state $\mathbf{x}_0$ and the system dynamics $\mathbf{f}_d$ which are incorporated in the optimisation as appropriate equality constraints. The concept is illustrated bellow:

$$\mathbf{u}^* = \underset{\mathbf{u}_0,\dots,\mathbf{u}_{N-1}}{\operatorname{argmin}} \sum_{n=0}^{N} \Phi(\mathbf{x}_n, \mathbf{u}_n), \tag{1.1a}$$

$$\text{s.t. : } \mathbf{x}_{n+1} = \mathbf{f}_d(\mathbf{x}_n, \mathbf{u}_n), \tag{1.1b}$$

$$\mathbf{x}_0 \text{ known}, \tag{1.1c}$$

where $\mathbf{x}$, $\mathbf{u}$ denote the control state and input respectively and N the discrete time horizon. The reference for the state and input is included in the function Φ. Given a computed optimal input sequence $\mathbf{u}^* = [\mathbf{u}_0^\top, \mathbf{u}_1^\top, \dots, \mathbf{u}_{N-1}]^\top$, $\mathbf{u}_0$ is applied to the system and the whole optimisation is repeated using an updated system state $\mathbf{x}_0$ at a constant rate.

Main advantages of MPC compared to traditional techniques include:

- Straightforward formulation for multivariate control systems.
- Explicit use of the control model.
- Additional state and input constraints can be easily included in the optimisation problem.
- Ability of the controller to anticipate future events and react accordingly in advance. This is because both the predicted system states and the future references are used in the optimisation.

Another advantage, often neglected, is the capability of incorporating some high level system behaviour rather than precise tracking of some desired reference. Examples include the works in [Kamel et al., 2017] where an MPC enabled reference tracking while simultaneously avoiding obstacles as well the works in [Vlantis et al., 2015, Falanga et al., 2018] where the used MPC included a perception objective aiming at minimising the risk of state estimation failure.

It becomes clear that the precision of the control model used in the optimisation plays a crucial role in the closed loop performance. As a result, significant effort has been applied on the MAV system modelling as well as the experimental identification of specific model parameters [Bähnemann et al., 2017, Sa et al., 2018a, Burri et al., 2018]. In the simplest form, a linear control model as in [Bangura and Mahony, 2014, Baca et al., 2016, Beul et al., 2017, Tzoumanikas et al., 2019] results in a computationally cheap (usually a Quadratic Program (QP)) optimisation problem. Whereas, the use of a nonlinear model resulting in a Nonlinear Model Predictive Control (NMPC), e.g. in [Neunert et al., 2016, Kamel et al., 2017a, Falanga et al., 2018], offers a better approximation of the true system dynamics in the expense of a more complex optimisation problem usually solved using a Newton type optimisation algorithm. An alternative approach aiming at combining the best from both worlds is approximating the nonlinear dynamics as a sequence of linear systems (obtained through linearisation around different operating points) resulting in a switching model linear MPC [Alexis et al., 2011, Alexis et al., 2012, Papachristos et al., 2016].

Additional to the classification based on the type of control model (linear/nonlinear), another criterion is the separation or not of the MAV dynamics into individual subsystems controlled by different controllers. The most common approaches are that of separate position and attitude control [Darivianakis et al., 2014, Kamel

et al., 2017a, Tzoumanikas et al., 2019], separate position and rotational rate control [Falanga et al., 2018] as well as the less popular unified approaches [Neunert et al., 2016]. The dynamics separation is often introduced to simplify the control model as well as enable low rate control updates since the fast attitude/rate dynamics are independently controlled.

In this thesis, we present three different MPC controllers. The first one relies on a linear control model and is based on the position-attitude dynamics separations. This makes it applicable to any MAV platform equipped with a tuned attitude controller and suitable for simple, non aggressive maneuvers. The second controller is an NMPC, with the body moments and collective thrust as control inputs, that jointly controls the full MAV state. The use of a nonlinear model which also captures coupling between the different Degrees of Freedom (DoFs) enables flying aggressive maneuvers while fully exploiting the dynamic capabilities of the system. The last one is an NMPC jointly controlling the motion of an MAV-arm system targeted for aerial manipulation applications. As it will be shown later it is able of handling the effects of the arm movement and its interaction with the environment in the MAV dynamics.

1.2.3 Fault tolerant control

A desirable property for every MAV operating in the real world is the ability to return to a home position and perform a safety landing in the event of a motor failure. Here, we use the term motor failure for the case of either an actual motor malfunction or a propeller snap which results in thrust loss. While the problem of a partial loss (i.e. when the produced thrust is a fraction of the commanded one) has also been studied in literature [Ranjbaran and Khorasani, 2010, Izadi et al., 2011, Giribet et al., 2017, Nguyen and Hong, 2018], here we focus on the more challenging and more realistic event of complete thrust loss.

There exist two distinct approaches of handling this in practice. The first one is by using MAV platforms which can withstand a motor failure. Examples include, octacopters [Marks et al., 2012, Saied et al., 2015, Saied et al., 2017], hexacopters with unconventional motor layouts [Schneider et al., 2012, Michieletto et al., 2018] or tilted motors [Ryll et al., 2016, Michieletto et al., 2017] which with the appropriate software adaptations (i.e. distributing the control effort to the functional motors) can withstand a motor failure. The second approach is entirely software-based and relies

on the prioritisation of position control over other –possibly uncontrollable– DoFs such as yaw. Examples of the latter applied on quadcopters is given in [Freddi et al., 2011, Mueller and D'Andrea, 2014, de Crousaz et al., 2015, Wu et al., 2019, Sun et al., 2020] and hexacopters with symmetric motor layout in [Kamel et al., 2015].

Additionally, fault tolerant control requires the existence of a supervisory algorithm which monitors the health status of the actuators and accordingly signals a failure. In other words, a motor failure can only be handled if it can be detected. We categorise the different proposed motor failure observers based on the input data required for fault detection. We distinguish those which rely on direct actuator measurements (e.g. encoder, current measurements) such as [Saied et al., 2017], those which rely on the knowledge of the full MAV state [Nguyen et al., 2019, Nguyen and Hong, 2019] and finally those that only require inertial measurements [Lu and van Kampen, 2015, Saied et al., 2015].

In our work, we propose a software-based fault handling technique which can be applied on any hexacopter irrespective of the motor layout. It further enables full position and yaw controllability when this is physically possible while enabling prioritisation of certain DoFs when full state tracking is not feasible. As far as the fault detection is concerned, we adopt an actuator effectiveness metric similar to the ones in [Izadi et al., 2011, Lu and van Kampen, 2015] while the designed observer solely depends on inertial measurements and can thus be applied as an algorithmic update to any MAV.

1.2.4 Aerial manipulation

Aerial manipulation aims at combining the maneuverability of MAVs with the precision and the fast dynamics of robotic manipulators. This union is particularly useful for application requiring physical contact with the environment but comes at the cost of increased control complexity due to the coupling of the dynamics of the two systems.

The simplest approach, with examples in [Orsag et al., 2013, Jimenez-Cano et al., 2013, Chermprayong et al., 2019], is to separate the problem into independent MAV and arm control and totally ignore the existence of coupling. The effects of coupling can be minimised by adopting clever manipulator designs such as [Ruggiero et al., 2015, Ohnishi et al., 2017] which actively move the battery in order to counteract

the torques acting on the MAV as a result of the arm displacement. In alternative software-based approaches such as [Mellinger et al., 2011, Ruggiero et al., 2015, Fresk et al., 2017, Suarez et al., 2018] the control problem remains separated but some of the effects introduced in the MAV dynamics, such as payload mass or Centre of Mass (CoM) displacement and internal torques, are estimated and compensated online. Unifying the control problem, as for example in [Garimella and Kobilarov, 2015, Kamel et al., 2016b, Lunni et al., 2017] for free flight tasks, results in superior performance compared to decoupled approaches since the effects of the arm motion on the MAV dynamics are compensated prior to their propagation into tracking errors. Additionally, systems and methods such as [Nguyen and Lee, 2013, Cataldi et al., 2016, Ryll et al., 2019, Bodie et al., 2020] which were successfully used in experiments that include contact with a surface, also considered the effect of the contact forces in the control model. In these works the external forces and moments (as a result of contact) acting on the MAV body were either directly measured or estimated using appropriate observers.

Regarding the control of the system ability to exert desired forces in the environment, the different approaches vary depending on the type of the used MAV. For example, omnidirectional vehicles [Bodie et al., 2019, Tognon et al., 2019, Ryll et al., 2019] are capable of independently controlling the translational and angular acceleration and when in contact can exploit this property to apply a reference force at a desired position and direction without the absolute need of the additional degrees of freedom introduced by an attached arm. In contrast, approaches applied on underactuated MAVs [Lippiello and Ruggiero, 2012, Nguyen and Lee, 2013, Darivianakis et al., 2014], irrespectively of the use of an arm or not, rely on the dynamical/inertial coupling between the MAV and the end effector in order to counteract the vehicle inability to exert lateral forces.

As mentioned in Section 1.2.2, in this work we propose a NMPC that jointly controls our custom built MAV-arm system. Apart from the benefits of the coupled control approach listed above, our approach further enables reference force tracking as the contact force acting on the end effector is expressed as a function of the system states. We showcase our algorithm using our underactuated MAV and further discuss how it can be applied on omnidirectional vehicles which have superior capabilities for manipulation tasks.

1.2.5 Learning based frameworks

In recent years there is a growing popularity of machine learning based robotic frameworks compared to more traditional physics/geometry based ones. The application of the former has become possible due to the computing hardware advances, the accessibility and availability of training data as well the plethora of open-source tools (e.g. multirotor simulators, photorealistic rendering software). As a result numerous machine learning frameworks have been successfully applied on the topic of autonomous MAV navigation. We categorise these in methods which i) tackle the problem state estimation, ii) of control iii) or both as a unified end-to-end approach.

In terms of state estimation, there exist several visual odometry frameworks such as [Clark et al., 2017, Wang et al., 2018, Zhan et al., 2018] which can compute the system state given raw sensor measurements (e.g. input images, Inertia Measurement Unit (IMU) data), show competitive with state-of-the-art methods performance without adopting any of the modules of traditional visual odometry frameworks. More specific to MAVs, learning based frameworks have been particularly successful in application-oriented experiments such as forest trail navigation [Giusti et al., 2016, Smolyanskiy et al., 2017], drone racing [Jung et al., 2018, Kaufmann et al., 2018, Loquercio et al., 2020] and collision free navigation [Gandhi et al., 2017, Loquercio et al., 2018, Kouris and Bouganis, 2018]. In all the above, high level commands (e.g. reference velocities, position or orientation) are generated using the learned policies and are executed using traditional motion planners and controllers.

Control focused approaches such as [Hwangbo et al., 2017, Koch et al., 2019] aim at learning policies which resemble the role of a conventional controller (mapping state estimates to control inputs). In the works mentioned above, reinforcement learning has been used for stabilising the position and attitude of a quadrotor respectively. Unified end-to-end approaches such as [Zhang et al., 2016, Kahn et al., 2017, Müller et al., 2018a, Kaufmann et al., 2020] attempt to learn a policy that directly maps sensor readings (e.g. IMU and camera data) to control actions (e.g. motor commands, MAV body rates and thrust). These methods have the benefit that an explicit state estimation scheme might not be required during the testing phase and can be potentially more efficient than optimisation based methods such as MPC since only a forward network pass is required per control iteration. The main challenge however is to ensure that policies which were entirely trained in simulation

generalise well in real life in the presence of unmodeled disturbances and dynamics.

Our work follows the rather classic division between perception and control. Despite the great works in learning based navigation, neither the state estimation nor the control algorithms presented in this work rely on a learnt model. We believe however that our MAV navigation pipeline can be used to provide optimal responses which can be utilised during the training phase of a supervised learning framework.

1.3 Contributions

The main results presented in this work have been published in three different research papers. The full list of publications done in conjunction with this work as well as the video material that provides visualisation of the algorithms developed, are given in Section 1.4.

1.3.1 Paper I: Linear MPC with soft constraints.

Tzoumanikas, D., Li, W., Grimm, M., Zhang, K., Kovac, M., and Leutenegger, S. (2019). **Fully autonomous micro air vehicle flight and landing on a moving target using visual-inertial estimation and model-predictive control**. *Journal of Field Robotics*. [Tzoumanikas et al., 2019].

In this paper we present an MPC for controlling the MAV position that relies on an approximate linear model of the vehicle. Regarding the control model we follow a very similar formulation to the ones presented in [Alexis et al., 2013, Darivianakis et al., 2014, Kamel et al., 2017a] where it was assumed that the MAV position and attitude dynamics can be controlled independently. Using this assumption, the full state MAV response can be controlled by a cascade connection of a position and an attitude controller with the latter receiving orientation and collective thrust commands from the former. The desired orientation, parameterised using Euler angles, as well as the collective thrust is thus considered the control input of the linear MPC which is obtained by solving the corresponding optimisation problem as previously discussed. Compared to the aforementioned approaches we also incorporate soft state constraints in the optimisation problem. These are used to prevent operation close to gimbal lock as well as impose operation within a safe state envelope which further guarantees that the control model remains valid. Modelling the state constraints as soft, which can be violated if necessary, ensures that the underlying

optimisation solver will never face an infeasible problem. Additionally, we formulate the resulting optimisation problem as a QP, in a canonical form, that can be solved online using any generic QP solver without requiring pre-computation of an explicit solution.

In addition to the MPC, we present a vision-based framework for tracking and landing on a moving target as required from the MBZIRC[1] challenge in which we participated in March 2017. We showcase the performance of the MPC and the vision based tracking in experiments resembling the challenge were our method results in successful landings for target velocities up to 5.0 m/s.

We further evaluate the performance of the controller in the experiments conducted for the purposes of the AABM[2] project where the main objective was tracking precision. In particular, we show that the controller is capable of tracking slow reference trajectories ($u_{\max} \leq 1$ cm/s) with a maximum per-axis error of 1.5 cm.

The linear MPC and the corresponding experiments are presented in Chapter 3.

1.3.2 Paper II: Fault tolerant NMPC

Tzoumanikas, D., Yan, Q., and Leutenegger, S. (2020). **Nonlinear MPC with Motor Failure Identification and Recovery for Safe and Aggressive Multicopter Flight**. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Tzoumanikas et al., 2020b].

Drawing inspiration from approaches such as [Kamel et al., 2015, Falanga et al., 2018, de Crousaz et al., 2015, Neunert et al., 2016] that rely on the non-linear model of the system, we present a quaternion based NMPC designed in order to overcome the limitations (i.e. use of Euler angles for the orientation, model is accurate for operation around hover) of its linear counterpart presented in Chapter 3. Compared to our linear MPC [Tzoumanikas et al., 2019] as well as similar in concept approaches [Kamel et al., 2017a, Falanga et al., 2018, Faessler et al., 2017] our NMPC does not rely on the existence of an attitude or rate controller. The full MAV state is controlled by a single control block which enables full state tracking (given dynamically feasible trajectories) while allowing the same controller to be used as an attitude, rate or mixed mode (e.g. altitude and orientation) controller by penalising

[1] See https://www.mbzirc.com/challenge/2017. Accessed April 2020.
[2] See http://www.aerial-abm.com/. Accessed April 2020.

the appropriate optimisation terms. This approach further eliminates the need for identification of approximate closed loop dynamics for the attitude or rate response while it only requires the knowledge of physical quantities such as the MAV mass, inertia matrix as well as motor thrust and moment coefficients.

In order to ensure that our method can be easily transferred to any underactuated MAV we select the MAV body moments and collective thrust as the control input. This choice yields a constant computational complexity for the NMPC optimisation problem irrespectively of the number of motors. The control input is then transformed into motor commands using a separate algorithm which we refer to as control allocation. We propose an optimisation-based control allocation which, compared to traditional approaches [Achtelik et al., 2013, Lee et al., 2010] that employ the pseudoinverse, can prioritise the tracking of certain control inputs and most importantly produces feasible motor commands. We further show how our control allocation algorithm can be used when bidirectional motors are present and we explain how it can be adapted in case of a complete failure of a single motor. Following a controllability analysis of our MAV, we use these two properties to stabilise the position and yaw of our hexacopter MAV under a complete failure of a single motor. Compared to approaches such as [Schneider et al., 2012, Mazeh et al., 2018, Nguyen et al., 2019] which rely on asymmetric motor layouts and can only handle a failure of specific motors, our approach can also be applied on hexacopters with symmetric motor layouts ensuring that failure of any motor can be handled equally well.

Finally, in order to handle a motor failure which may occur in mid-flight, we developed an EKF observer which monitors the health status of all the motors and accordingly notifies the control allocation block in the event of a failure. The estimated health status corresponds to the percentage of motor thrust acting on the MAV body, thus providing an easy to tune threshold for triggering the failsafe. Our implementation relies on inertial measurements and can thus be applied as an algorithmic only update to any MAV.

These algorithms and the corresponding experimental results with and without failures are presented in Chapter 4.

1.3.3 Paper III: NMPC for aerial manipulation

Tzoumanikas, D., Graule, F., Yan, Q., Shah, D., Popovic, M., and Leutenegger, S. (2020). **Aerial Manipulation Using Hybrid Force and Position NMPC Applied to Aerial Writing**. *In Proceedings of Robotics: Science and Systems (RSS)*. [Tzoumanikas et al., 2020a].

In this paper, we present an extension of the NMPC, briefly described above, targeted for applications requiring interaction between the MAV and its environment such as inspection through contact. We consider the case where the MAV is equipped with a robotic arm with the main idea being the fast arm dynamics to compensate for errors occuring on the MAV base. In our approach we tackle the control problem in a model based way by using a NMPC that jointly controls the MAV-arm system. This is done by formulating a hybrid control model where the MAV is modeled as a single rigid body object and the quasi-static forces introduced by the arm and its interaction with the environment are taken into account. Compared to approaches such as [Darivianakis et al., 2014, Kamel et al., 2016b, Lunni et al., 2017] which either use two models (one for free flight and another for physical interaction) or only consider the free flight case, including the effect of contact forces into a single control model, eliminates the need of a switching mode controller, allows the NMPC to reason about contact before it actually happens and to natively handle the disturbance forces and moments (as a result of contact) acting on the MAV body. This approach further enables automatic compensation of the effects of the arm motion on the MAV dynamics such as the system's CoM displacement due to arm movement.

Similar to the NMPC presented in Chapter 4 the controller does not rely on the existence of a closed loop attitude/rate controller as we consider the MAV body moments, the collective thrust and the end-effector position as the control input. For the MAV related inputs we use the same control allocation algorithm also described in Chapter 4 while the end-effector position is transformed into actuator commands using the arm-specific forward kinematics which are separately described in Section 2.3.1. The specific choice of the control input enables easy adaptation to similar systems and omnidirectional vehicles [Bodie et al., 2019, Ryll et al., 2019, Brescianini and D'Andrea, 2018b] which have superior force exertion capabilities, since the control allocation and forward kinematics are the only platform-dependent components.

We evaluate the accuracy of the MAV-arm system using our custom built underactuated MAV and delta arm in *aerial writing* experiments that require multiple transitions between free flight and contact. Our system shows accuracy in the order of millimeters and repeatability in tracking trajectories of different type, size and with different velocity and acceleration profiles.

These algorithms and the corresponding experimental results are presented in Chapter 5.

1.4 Publications

The work described in this thesis resulted in the following publications:

- Tzoumanikas, D., Li, W., Grimm, M., Zhang, K., Kovac, M., and Leutenegger, S. (2019). **Fully autonomous micro air vehicle flight and landing on a moving target using visual-inertial estimation and model-predictive control**. *Journal of Field Robotics*. [Tzoumanikas et al., 2019].
- Tzoumanikas, D., Yan, Q., and Leutenegger, S. (2020). **Nonlinear MPC with Motor Failure Identification and Recovery for Safe and Aggressive Multicopter Flight**. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Tzoumanikas et al., 2020b].
- Tzoumanikas, D., Graule, F., Yan, Q., Shah, D., Popovic, M., and Leutenegger, S. (2020). **Aerial Manipulation Using Hybrid Force and Position NMPC Applied to Aerial Writing**. *In Proceedings of Robotics: Science and Systems (RSS)*. [Tzoumanikas et al., 2020a].

The following video material provides visualisation of some of the algorithms developed in this thesis:

- Landing on a moving target using visual-inertial estimation and model predictive control, `https://youtu.be/0RW68FGHVX8`.
- Nonlinear MPC with autonomous motor failure identification and recovery, `https://youtu.be/cAQeSZ3tIqY`.
- Nonlinear MPC for aerial manipulation, `https://youtu.be/iE--M00YF0o`.

While not described directly, the following publications were done in conjunction with this thesis:

- Dai, A., Papatheodorou, S., Funk, N., Tzoumanikas, D. and Leutenegger S. (2020). **Fast Frontier-based Information-driven Autonomous Exploration with an MAV**.*In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Dai et al., 2020].

- Xu, B., Li, W., Tzoumanikas, D., Bloesch, M., Davison, A. and Leutenegger S. (2019). **MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM**. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Xu et al., 2019].

- Zhang, K., Chermprayong, P., Tzoumanikas, D., Li, W., Grimm, M., Smentoch, M., Leutenegger, S. and Kovac, M. (2019). **Bioinspired design of a landing system with soft shock absorbers for autonomous aerial robots**. *Journal of Field Robotics*. [Zhang et al., 2019].

- Saeedi, S., Carvalho, E., Li, W., Tzoumanikas, D., Leutenegger, S., Kelly, P. and Davison, A. (2019). **Characterizing Visual Localization and Mapping Datasets**. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Saeedi et al., 2019].

- Li, W., Saeedi, S., McCormac, J., Clark, R., Tzoumanikas, D., Ye, Q., Huang, Y., Tang, R. and Leutenegger, S. (2018). **InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset**. *British Machine Vision Conference (BMVC)*. [Li et al., 2018].

1.5 Thesis structure

The remainder of this thesis is structured as follows:

Chapter 2 introduces basic notation, the multirotor platforms and their sensor setup as well as the different state estimation algorithms used in this work.

Chapter 3 describes a simple linear model predictive controller that relies on decoupling of the position and attitude dynamics. Its capabilities and limitations are

illustrated through experiments including tracking and landing on a fast moving target (as required in MBZIRC 2017 Challenge 1) and following trajectory commands for the purposes of the AABM project

Chapter 4 describes a non-linear model predictive controller developed to overcome the limitations of the one presented in Chapter 3. Additionally, it is shown how motor failures can be identified and handled by the developed algorithm. Its flight performance is evaluated on experiments with and without motor failures.

Chapter 5 extends the controller presented in Chapter 4 for aerial manipulation tasks. A hybrid model for the combined MAV-manipulator system is formulated that also takes into account the interaction forces from the environment. The combined system is jointly controlled by a single control block. The achieved accuracy and repeatability are shown in a series of "aerial-writing" tasks.

Chapter 6 concludes this thesis with a summary of the results presented and suggestions for future work.

Chapter 2

Preliminaries

Contents

In this Chapter, we present the software and hardware components that form the foundations for the algorithms presented ahead. In terms of layout, we start with the mathematical notation and continue with the presentation of the main MAV and payload hardware components. We conclude with the pipelines used for MAV state estimation.

2.1 Notation

In this work we use the following notation:

2.1.1 General notation

a A lower-case symbol denotes a scalar apart from common capital exceptions.

$\mathbf{a}$ A bold lower-case symbol denotes an m-dimensional column vector with a_i the i^{th} element as:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}, \mathbf{a}^\top = [a_1, a_2, \ldots, a_m]. \tag{2.1}$$

We use $\mathbf{a}_{i:j}$ to denote the vector consisting of the elements of $\mathbf{a}$ with indices in the $[i, j]$ range.

$\boldsymbol{a}$ A bold italic lower-case symbol represents an homogeneous vector defined as: $\boldsymbol{a} = [\mathbf{a}, 1]^\top$.

$\mathbf{A}$ A bold capital symbol denotes an $m \times n$ matrix with $a_{i,j}$ the element at the i^{th} row and j^{th} column:

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ldots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \tag{2.2}$$

$\mathbf{I}$ The identity matrix, optionally with dimensions as subscript.

$\mathbf{0}$ The zero matrix, optionally with dimensions as subscript.

$[\cdot]^\times$ The cross-product matrix that produces a skew symmetric matrix from a 3D vector such that $\mathbf{a} \times \mathbf{b} = [\mathbf{a}]^\times \mathbf{b}$. Given the vector $\mathbf{a} = [a_x, a_y, a_z]^\top$, $[\mathbf{a}]^\times$ can be computed by:

$$[\mathbf{a}]^\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}. \tag{2.3}$$

$\mathcal{A}$ A calligraphic capital symbol denotes a set.

2.1.2 Spaces and manifolds

$\mathbb{R}$ The set of real numbers.

$\mathbb{R}^+$ The set of positive real numbers.

$\mathbb{R}^m$ The vector space of real m-dimensional vectors.

$\mathbb{R}^{m \times n}$ The vector space of real $m \times n$-dimensional matrices.

$\mathbb{Z}$ The set of integers.

$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ The Normal distribussion with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

S^3 The 3-sphere group.

$SO(3)$ The group of 3D rotations.

$SE(3)$ The group of 3D rigid transformations.

$\boxplus$ The "box-plus" operator that applies a perturbation expressed in a tangent space to a manifold state.

$\boxminus$ The "box-minus" operator that expresses the difference of two manifold states in the tangent space.

2.1.3 Frames and transformations

$\underrightarrow{\mathcal{F}}_A$ A cartesian coordinate frame in $\mathbb{R}^3$.

${}_A\mathbf{v}$ A vector $\mathbf{v}$ expressed in the frame $\underrightarrow{\mathcal{F}}_A$.

${}_A\mathbf{r}_P$ The position vector from the origin of $\underrightarrow{\mathcal{F}}_A$ to the point P represented in the coordinate frame $\underrightarrow{\mathcal{F}}_A$.

${}_A\boldsymbol{r}_P$ The position vector represented in homogeneous coordinates: ${}_A\boldsymbol{r}_P = [{}_A\mathbf{r}_P^\top, 1]^\top$.

${}_A\mathbf{v}_{BC}$ The linear velocity of $\underrightarrow{\mathcal{F}}_C$ with respect to $\underrightarrow{\mathcal{F}}_B$ expressed in $\underrightarrow{\mathcal{F}}_A$.

${}_A\boldsymbol{\omega}_{BC}$ The rotational velocity of $\underrightarrow{\mathcal{F}}_C$ with respect to $\underrightarrow{\mathcal{F}}_B$ expressed in $\underrightarrow{\mathcal{F}}_A$.

ϕ The roll angle.

θ The pitch angle.

ψ The yaw angle.

$\mathbf{C}_{AB}$ The rotation matrix that transforms the vector ${}_{B}\mathbf{v}$ expressed in $\underrightarrow{\mathcal{F}}_B$ to one expressed in $\underrightarrow{\mathcal{F}}_A$ as: ${}_{A}\mathbf{v} = \mathbf{C}_{AB}\,{}_{B}\mathbf{v}$. The inverse rotation $\mathbf{C}_{BA}$ can be computed as: $\mathbf{C}_{BA} = \mathbf{C}_{AB}^{-1} = \mathbf{C}_{AB}^{\top}$.

$\mathbf{q}$ The quaternion following the Hamiltonian convention with vector part $\mathbf{q}_v$ and real part q_w give by: $\mathbf{q} = [\mathbf{q}_v^{\top}, q_w]^{\top}$. When used for representing orientation, the unit length $\mathbf{q}_{AB}$ represents the quaternion equivalent of the rotation matrix $\mathbf{C}_{AB}$.

$\mathbf{q}^*$ The conjugate of the quaternion $\mathbf{q}$ defined by: $\mathbf{q}^* = [-\mathbf{q}_v^{\top}, q_w]^{\top}$. For the rotation quaternion $\mathbf{q}_{AB}$, its conjugate represents the inverse rotation as: $\mathbf{q}_{AB}^* = \mathbf{q}_{AB}^{-1} = \mathbf{q}_{BA}$.

$\otimes$ The quaternion multiplication such that $\mathbf{q}_{AC} = \mathbf{q}_{AB} \otimes \mathbf{q}_{BC}$.

$[\cdot]^+$ The left-hand quaternion matrix such that $\mathbf{q}_{AC} = [\mathbf{q}_{AB}]^+ \mathbf{q}_{BC}$. In matrix form $[\mathbf{q}]^+$ is given by:

$$[\mathbf{q}]^+ = \begin{bmatrix} q_w & -\mathbf{q}_v^{\top} \\ \mathbf{q}_v & q_w\mathbf{I} + [\mathbf{q}_v]^{\times} \end{bmatrix}. \tag{2.4}$$

$[\cdot]^{\oplus}$ The right-hand quaternion matrix such that $\mathbf{q}_{AC} = [\mathbf{q}_{BC}]^{\oplus} \mathbf{q}_{AB}$. In matrix form $[\mathbf{q}]^{\oplus}$ is given by:

$$[\mathbf{q}]^{\oplus} = \begin{bmatrix} q_w & -\mathbf{q}_v^{\top} \\ \mathbf{q}_v & q_w\mathbf{I} - [\mathbf{q}_v]^{\times} \end{bmatrix}. \tag{2.5}$$

$\boldsymbol{T}_{AB}$ The transformation matrix that transforms homogeneous vectors from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_A$ as ${}_{A}\boldsymbol{r}_P = \boldsymbol{T}_{AB}\,{}_{B}\boldsymbol{r}_P$. The relationship between $\boldsymbol{T}_{AB}$, the rotation matrix $\mathbf{C}_{AB}$ and the position vector ${}_{A}\mathbf{r}_B$ is given by:

$$\boldsymbol{T}_{AB} = \begin{bmatrix} \mathbf{C}_{AB} & {}_{A}\mathbf{r}_B \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{2.6}$$

while the inverse transformation $\boldsymbol{T}_{BA}$ can be computed as:

$$\boldsymbol{T}_{BA} = \boldsymbol{T}_{AB}^{-1} = \begin{bmatrix} \mathbf{C}_{AB}^{\top} & -\mathbf{C}_{AB}^{\top}\,{}_{A}\mathbf{r}_B \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.7}$$

2.2 Multirotor platforms

In this section, we present the main hardware components of the MAV platforms used in the experiments presented in this work. As our main focus is not the platform design by itself, we briefly describe the essential hardware components while we put special emphasis on the identification of physical quantities (e.g motor coefficients) that are crucial for our system's performance when operating with our model based controllers.

Two different vehicles were built, based on the same frame shown in Figure 2.1, using off the shelf components and 3D printed parts. They were designed according to the following general specifications:

- The MAV including the extra payload (e.g. cameras, manipulator) has to be small enough ($< 1.0\,\text{m}$ in diagonal) and thus suitable for indoor operation.
- Its thrust to weight ratio should ideally exceed $2:1$ in order to enable sufficiently fast closed loop response and allow for additional payload.
- It should be equipped with sufficient onboard computing resources in order to be able to run the state estimation and control algorithms.
- It should support additional sensors required for completely autonomous navigation e.g. cameras, Global Positioning System (GPS).

Hexacopters (i.e. vehicles with six motors) were preferred due to their payload capacity and compact size compared to heavy lift quadcopters. Additionally, as discussed in Chapter 4 when equipped with bidirectional motors and the appropriate software, they can maintain full position and yaw controllability despite the loss of a single motor.

2.2.1 Onboard computer

All the developed algorithms run on an Intel NUC-7567U onboard computer running Ubuntu Server 16.04 shown in Figure 2.2. Its Intel Core i7 Central Processing Unit (CPU) coupled with 32GB of RAM provide enough computational capabilities for the estimation and control algorithms developed, while providing additional overhead if required.

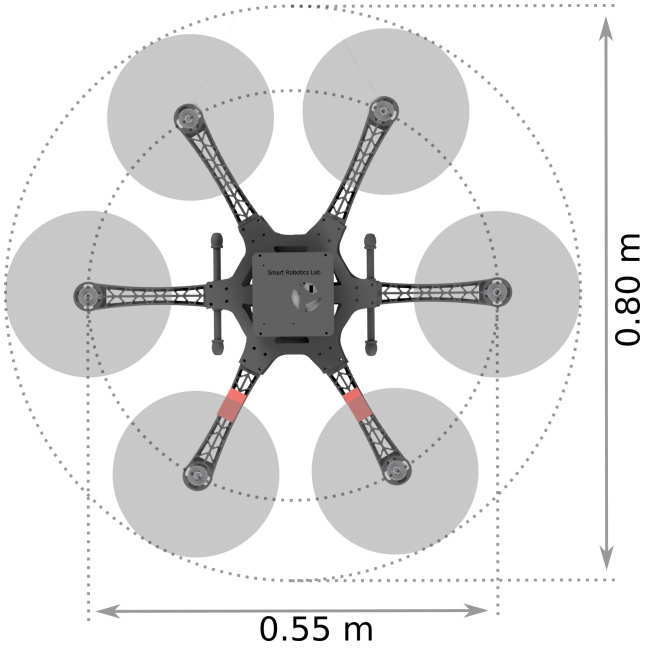


Figure 2.1: The custom-built platforms use a 0.55 m wide frame while the overall diameter (which depends on the propeller choice) is less than 0.8 m.

In order to maximise performance the CPU governor is permanently set to the performance mode. Additionally, we have modified the Operating System (OS) kernel to enable faster communication with the peripherals connected through serial. Finally, all the OS related power management settings (e.g. the WiFi adapter power saving mode) have been disabled in order to minimise delays. All these modifications come at the cost of increased power consumption. However, even at full load this remains bellow 40 W (not including the power consumption of connected peripherals e.g. cameras) which is a fraction of the power consumed by the motors (up to 6×170 W).

All our algorithms are implemented using C++, where when required we use multithreading, vectorisation and appropriate compiler optimisation flags in order to boost performance.

2.2.2 Flight controller

With the term *flight controller*, we refer to the board which contains the absolute necessary sensors for stable flight (such as the IMU) and which also interfaces with the motors. It is essentially the main hardware component of every commercially available MAV that can be remotely controlled with a joystick. It mainly consists

Figure 2.2: The MAV onboard computer.

of the following elements:

- Sensors required for MAV state estimation such as IMUs, magnetometers, barometer, GPS.
- Necessary hardware for interfacing with the MAV motors such as: PWM or I^2C drivers.
- Communication ports (e.g. UART, USB) required for data exchange with external devices (e.g additional sensors, onboard computers, RC receivers).
- An onboard CPU handling the interface with the sensors and motors as well as running the necessary estimation and control algorithms.

In all our platforms, the commercially available mRo PixRacer[1] flight controller, shown in Figure 2.3, was preferred for two main reasons: (i) its compact size (36 mm×36 mm) and low weight ($<$ 11 g) without compromising functionalities of a complete autopilot system (i.e. 168 MHz Cortex M4 CPU, 5×UART, 6×PWM outputs, 2×IMU for redundancy) (ii) its capability to be flashed with widely used in research open-source software such as PX4[2] or ArduPilot[3].

[1]See https://docs.px4.io/v1.9.0/en/flight_controller/pixracer.html. Accessed April 2020.

[2]See https://github.com/PX4/Firmware. Accessed April 2020.

[3]See https://github.com/ArduPilot/ardupilot. Accessed April 2020.

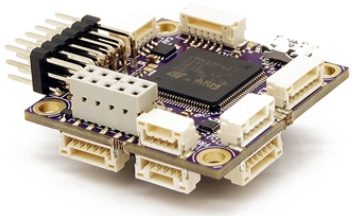


Figure 2.3: The mRo PixRacer flight controller used in both MAVs.

Special attention was paid on how the flight controller was mounted on the MAV. In order to reduce the high frequency vibrations (originating from the motor rotation) that affect the natively noisy IMU measurements, the device was rigidly attached on the onboard computer which was soft-mounted on the MAV frame. Another possible solution adopted in the MAV shown in Figure 2.11 (right) is to soft-mount the motors on the frame and thus preventing the transfer of vibrations on the frame.

On the software side, we flashed the flight controller with a modified PX4[2] firmware. Briefly, we kept the main software components such as the EKF-based state estimator, the attitude-rate controllers and device drivers while our modifications included:

- Reducing the CPU load and minimising communication delays.
- Modifying the IMU drivers in order to accessing the raw measurements at the highest possible rate.
- Adding support for bidirectional capable motors.

The PX4 firmware comes with a wide variety of pre-implemented estimation and control algorithms and can be deployed for autonomous missions without the need of additional software. However, in our work we mainly used the flight controller as a middleware used for reading of the IMU measurements and interfacing with the motors. Apart from the algorithms presented in Chapter 3 where the pre-implemented attitude controller was used, the algorithms in Chapters 4 and 5 do

not use any of the estimators or controllers of PX4. In these experiments, the flight controller which runs independent estimation and control algorithms (than the ones implemented on the onboard computer), acted as a safety backup in the event of a malfunction of the primary software/hardware stack.

2.2.3 Propulsion system

Two different propulsion systems were employed depending on the desired application. The individual components of each one, consisting of the Electronic Speed Controller (ESC), motor and propeller are shown in Figure 2.4.

(a) Propulsion system #1 is sold as a single kit and consists of the DJI 420 ESC, the DJI 2312E 960Kv motors and the self tightening DJI 9450 propellers.

(b) Propulsion system #2 consists of the bidirectional capable DYS 35A Aria ESC, the DJI 2312E 960Kv motors and the carbon reinforced Aeronaut CAMcarbon Light 9545 propellers. In order to cope with the large angular acceleration/deceleration due to direction change, the propellers were secured with elastic stop nuts.

Figure 2.4: The two different propulsion systems used in our MAVs.

Main difference between the two is that propulsion system #2 can reverse the direction of motor rotation in mid-flight. Based on that the motors can either rotate in a normal or inverted direction and thus generate thrust and moment in two directions. This property was used for stabilising the yaw in the event of a motor failure (experiments presented in Chapter 4). Unlike, conventional system designs that use symmetrical propellers (which have identical characteristics irrespectively of direction of rotation), we preferred propellers that are optimised for rotation in one direction.

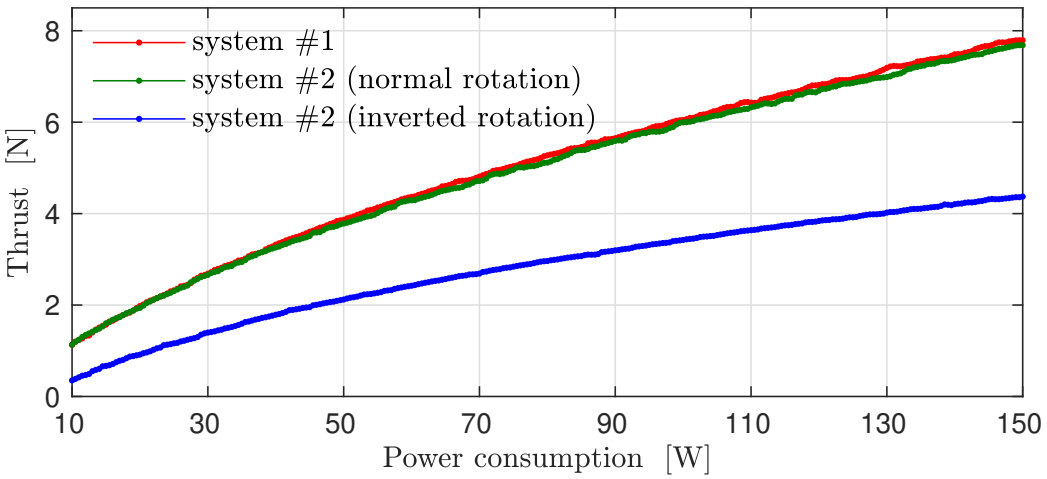


Figure 2.5: Per motor absolute thrust versus power consumption of the two propulsion systems. System #1 and system #2 (normal rotation) are very similar in terms of power consumption with system #1 being slightly more efficient. As expected system #2 is the least efficient when the motors are commanded to rotate in the opposite direction. This is because the propellers are aerodynamically optimised for rotation in one direction.

Based on the efficiency curves obtained from experimental data, this choice results on very similar efficiency curves for propulsion system #1 and system #2 (normal rotation). Propulsion system #1 is marginally more efficient and was thus preferred for task requiring significant payload (experiments presented in Chapters 3 and 5). Not surprisingly, the efficiency of propulsion system #2 (inverted rotation) is significantly lower than others. However, in this work inverted rotation was only used in the event of a motor failure and was completely disabled during failure-free operation.

The total power consumption P_{total} plotted in Figure 2.5 is the sum of the mechanical P_{mech} and the electrical power loss P_{el}: $P_{\text{total}} = P_{\text{mech}} + P_{\text{el}}$. Based on the analysis given in [Brescianini and D'Andrea, 2018a] the mechanical power dominates the power losses and is characterized by a third order polynomial in angular velocity resulting in: $P_{\text{total}} \propto |f|^{\frac{3}{2}}$, with f the motor generated thrust.

Regarding the relationship between the generated thrust f, moment M and the motor rotational velocity ω, we use the following model:

$$f = k_T \operatorname{sgn}(\omega)\omega^2, \tag{2.8a}$$

$$M = -k_M \operatorname{sgn}(\omega) f, \tag{2.8b}$$

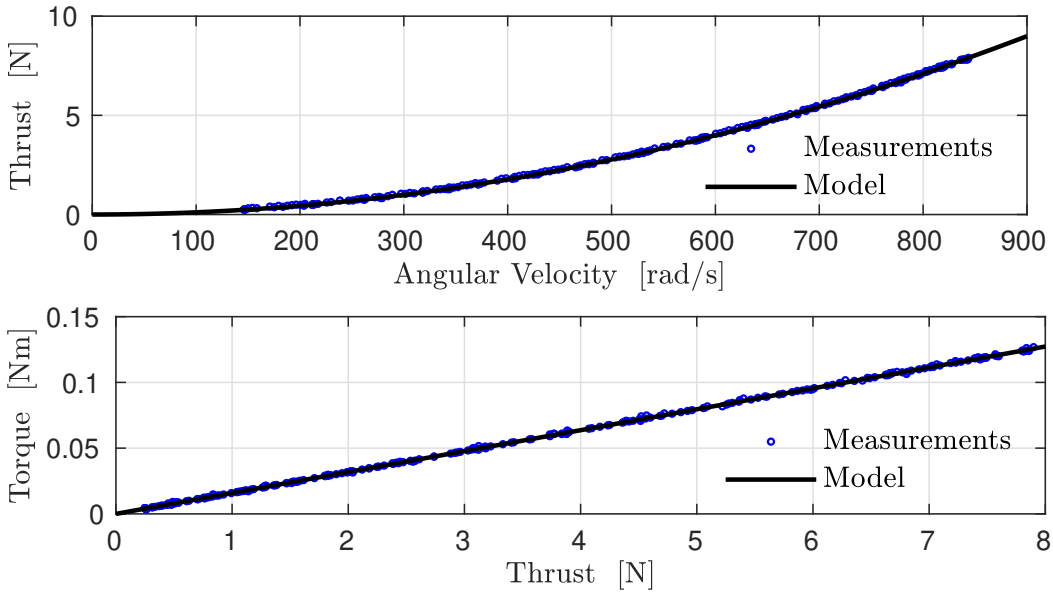


Figure 2.6: Identification results of the thrust and moment coefficients of propulsion system #1. Each dot corresponds to an average of 100 measurements and the solid line to the fitted model as defined in (2.8). The least squares fit error for the thrust and moment model is 4.8×10^{-2} N and 5.3×10^{-4} N m respectively.

with k_T, k_M thrust and moment constants. These were estimated using a load cell and an optical encoder for accurate rotational measurements. Figures 2.6 and 2.7 show the results of the experimental identification for system #1 and #2 respectively. The numeric values of the identified constants are given in Table 2.1. Note that in propulsion system #2 (which can switch direction of rotation on demand) we use non symmetrical propellers which results in different coefficients depending on the direction of rotation. We thus identified two sets of parameters, with k_T^+, k_M^+ and k_T^-, k_M^- corresponding to normal and inverted rotation respectively.

Unfortunately, conventional ESCs do not directly control the angular velocity of the motor. They are responsible for swtiching on and off the transistors that supply the motor coils with a fixed proportion (based on the PWM command) of the battery voltage. Consequently, the achieved angular velocity depends on the PWM command and the battery voltage which gradually decreases during flight. In our application we are interested in controlling the motor angular velocity and consequently the thrust and moment as accurately as possible. To achieve this with our sensorless ESC-motor setup, we experimentally determined the relationship between a PWM command and the achieved thrust and moment. Our model

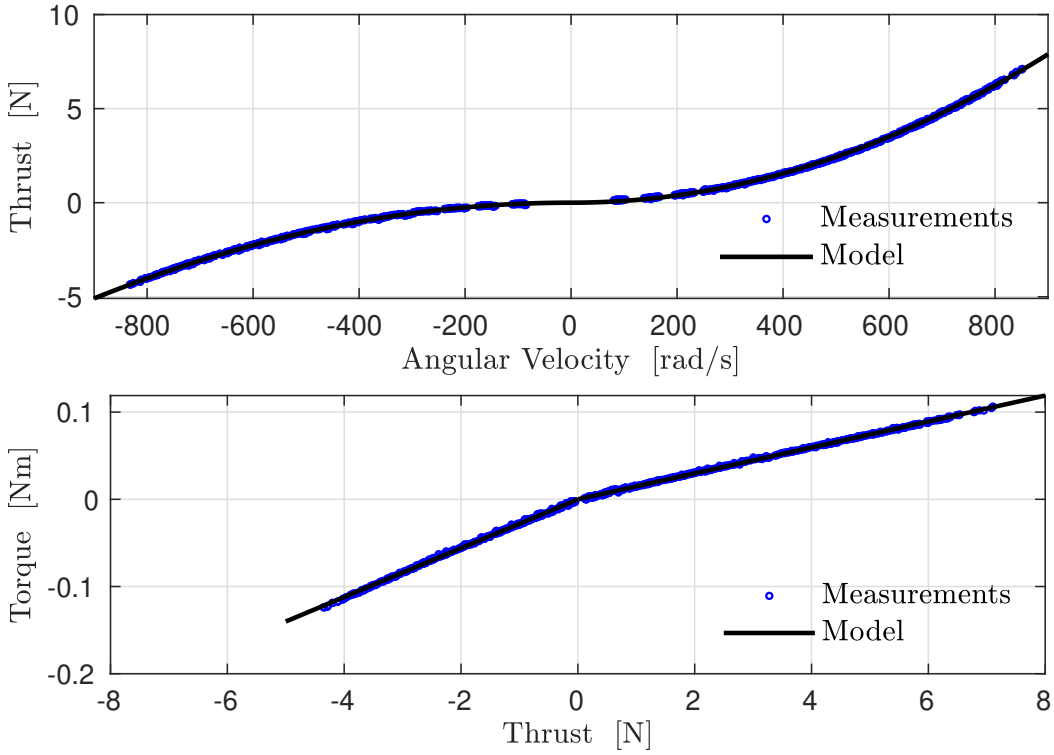


Figure 2.7: Identification results of the thrust and moment coefficients of propulsion system #2. Each dot corresponds to an average of 100 measurements and the solid line to the fitted model as defined in (2.8). Since non symmetrical propellers were used two sets of coefficients corresponding to normal (k_T^+, k_M^+) and inverted rotation (k_T^-, k_M^-) were identified. The least-squares fit error for the thrust and moment model for normal motor rotation is 5×10^{-2} N and 8.5×10^{-4} Nm respectively. The same quantities for inverted rotation are 4.4×10^{-2} N and 1.3×10^{-3} Nm.

Description	**Symbol**	**Value**	**Unit**
Propulsion system #1			
Thrust coefficient	k_T	1.1090×10^{-5}	$\mathrm{N/(rad/s)^2}$
Moment coefficient	k_M	1.5864×10^{-2}	Nm/N
Propulsion system #2			
Normal rotation thrust coefficient	k_T^+	9.7408×10^{-6}	$\mathrm{N/(rad/s)^2}$
Normal rotation moment coefficient	k_M^+	1.4873×10^{-2}	Nm/N
Inverted rotation thrust coefficient	k_T^-	6.3309×10^{-6}	$\mathrm{N/(rad/s)^2}$
Inverted rotation moment coefficient	k_M^-	2.7928×10^{-2}	Nm/N

Table 2.1: Thrust and moment coefficients of the two propulsion systems.

also takes into account the effect of the varying battery voltage. Regarding, the relationship between the rotational velocity ω and the PWM command c this was approximated using the following second order polynomial:

$$\omega = a_{V,2}c^2 + a_{V,1}c + a_{V,0}, \tag{2.9}$$

where the coefficients $a_{V,i}$ were identified for different battery voltage levels V, stored in lookup tables and used online depending on the measured battery voltage. The identification results of the model defined in (2.9) for the two propulsion systems are shown in Figure 2.8 where for visualisation purposes we only show the identified curves for 12 different voltage levels.

To summarise, given a desired motor force f, the desired angular velocity ω is computed using the model defined in (2.8) and the identified coefficients in Table 2.1. The desired angular velocity ω is then converted to a PWM command c using the model in (2.9) with the coefficients $a_{V,i}$ that correspond to the measured voltage V. We would like to highlight that our approach is open loop and thus natively inferior to closed loop approaches such as [Papachristos et al., 2012] or [Brescianini and D'Andrea, 2018b] where the velocity loop was closed using encoder feedback. However, it is superior than open loop approaches that do not include voltage compensation. An easier way of tackling the same problem would be to use commercially available hardware able to perform closed loop angular control. Unfortunately, these motors[4] are mainly designed for small MAVs and not ones like ours that carry significant payload.

2.2.4 Cameras

In this work we used cameras for two different purposes. The first being running SLAM for navigation not requiring external sensors (e.g motion capture system) and the second for generic environment perception (e.g. detecting, following and landing on a moving target as shown in Chapter 3). The cameras used for navigation, shown in Figure 2.9, are also equipped with an onboard IMU as the used SLAM algorithms also require inertial measurements. For simpler tasks, we used global shutter cameras as the ones shown in Figure 2.10.

Regarding the relationship between a 3D point ${}_C\mathbf{r}_P := [r_x, r_y, r_z]^\top$ in space and its corresponding projection $\mathbf{u} := [u, v]^\top$ onto the camera plane, we use a standard

[4] See https://iq-control.com/. Accessed April 2020.

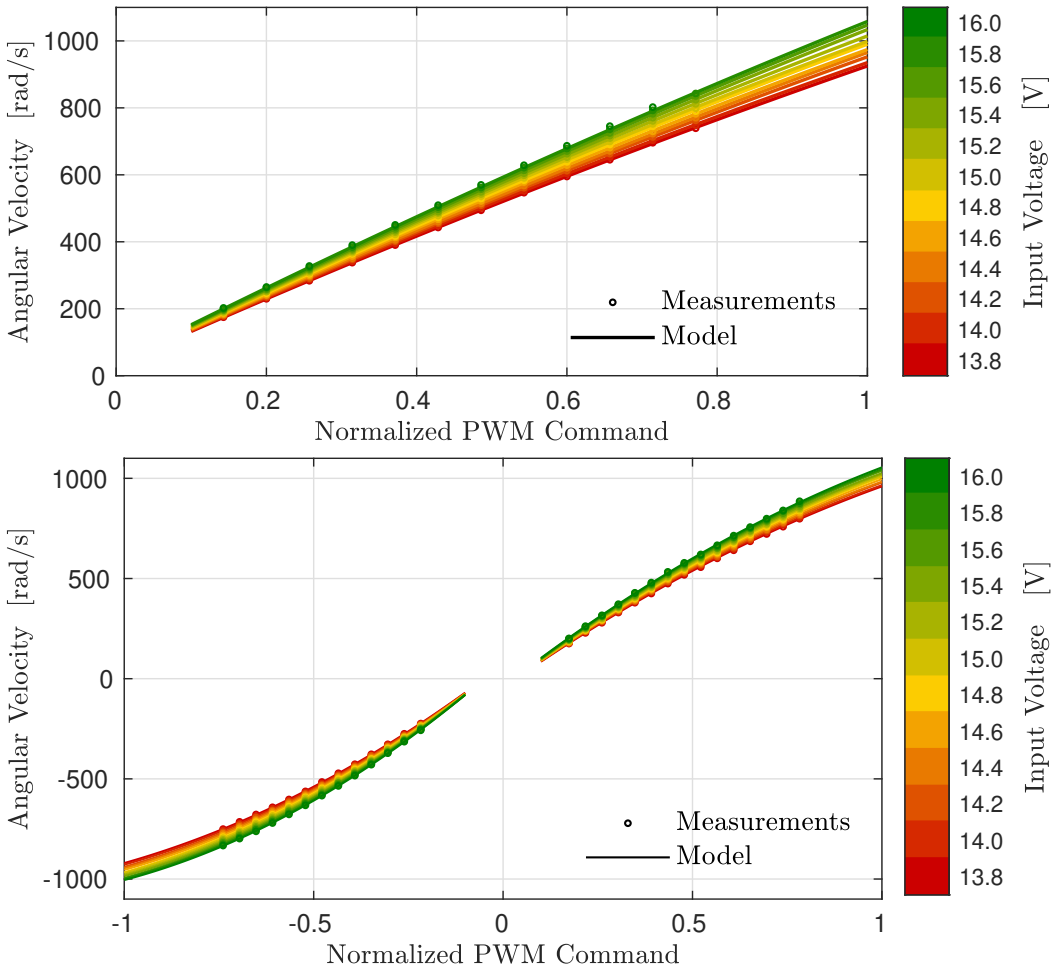
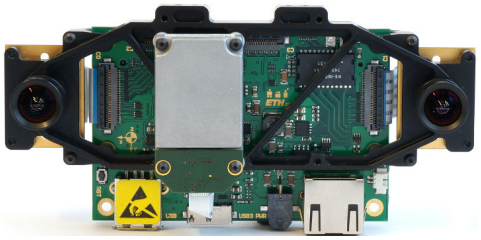


Figure 2.8: Experimental identification of the relationship between the PWM command and the achieved angular velocity for system #1 (top) and system #2 (bottom). For visualisation purposes we only show 12 different curves that correspond to different voltage levels.

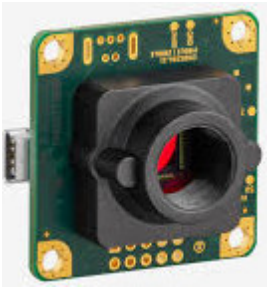


(a) The Skybotix Visual-Inertial (VI)-sensor with modified drivers that enable the use of one of its pins as an external trigger for data synchronisation with external devices.

(b) The Intel Realsense ZR300 Red Green Blue Depth (RGBD)-inertial sensor.

Figure 2.9: Camera sensors used for SLAM.

(a) FLIR Chameleon3 supporting resolutions up to 1288 × 964 @ 30 FPS.



(b) UI-1221LE supporting image resolution up to 752 × 480 @ 87 FPS.

Figure 2.10: Cameras used for the target tracking experiments presented in Chapter 3.

pinhole camera model. This is done in the following steps:

1. Projection onto the unit plane: $\mathbf{r}'_P = \frac{1}{r_z}\begin{bmatrix} r_x \\ r_y \end{bmatrix}$.
2. Apply the distortion model $\mathbf{d}$: $\mathbf{r}''_P = \mathbf{d}(\mathbf{r}'_P)$.
3. Scale and translate: $\mathbf{u} = \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix}\mathbf{r}''_P + \begin{bmatrix} c_x \\ c_y \end{bmatrix}$.

The projection model is not invertible and the point ${}_C\mathbf{r}_P$ can only be recovered when the depth r_z is known. In that case, ${}_C\mathbf{r}_P$ can be recovered by:

1. Convert the pixel coordinates to unit plane coordinates: $\mathbf{r}''_P = \begin{bmatrix} \frac{1}{f_x} & 0 \\ 0 & \frac{1}{f_y} \end{bmatrix}\left(\mathbf{u} - \begin{bmatrix} c_x & c_y \end{bmatrix}\right)$.
2. Correct for lens distortion: $\mathbf{r}'_P = \mathbf{d}^{-1}(\mathbf{r}''_P)$.
3. Convert to $\underrightarrow{\mathcal{F}}_C$ coordinates: ${}_C\mathbf{r}_P = r_z \begin{bmatrix} r'_x & r'_y & 1 \end{bmatrix}^\top$.

The camera intrinsics (f_x, f_y, c_x, c_y), the distortion model parameters as well as the transformation between the camera frame $\underrightarrow{\mathcal{F}}_C$ and the IMU frame $\underrightarrow{\mathcal{F}}_S$ were obtained using Kalibr[5] an open source calibration tool based on [Furgale et al., 2013].

[5] See https://github.com/ethz-asl/kalibr. Accessed April 2020.

2.2.5 Additional platforms and simulation environment

Even though not used for experiments presented in this work, the developed algorithms have been also implemented and thoroughly tested on two additional platforms. A commercially available AscTec Firefly hexacopter and a custom-built racing quadcopter shown in Figure 2.11.

(a) Vision based flight of an AscTec Firefly equipped with an Intel i7-3612QE.

(b) A racing quadcopter equipped with an Nvidia Jetson Tx1 computer.

Figure 2.11: Two additional platforms used during this research.

Prior to deployment on real systems, all our algorithms were thoroughly tested in simulation. We used the open source MAV simulator RotorS[6] described in [Furrer et al., 2016] which apart from multirotor models, provides simulated measurements of commonly used sensors (such as IMU, cameras, pose sensors). Conveniently, it is equipped with a ROS interface plugin allowing seamless transition between the simulation environment and the real system. A snapshot of our system executing the same task in real life and in simulation is given in Figure 2.12.

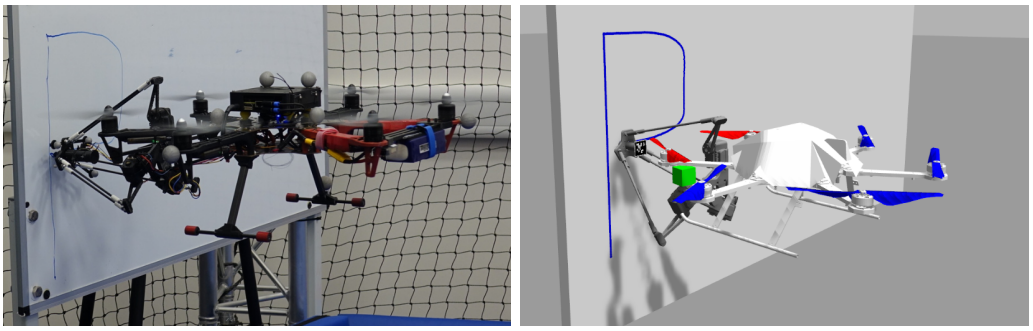


Figure 2.12: A snapshot of our MAV-arm system performing the same "aerial-writing" task in reality (left) and in the simulation environment (right).

[6] See https://github.com/ethz-asl/rotors_simulator. Accessed April 2020.

2.3 Aerial manipulator

For the experiments requiring physical interaction with the environment, we use a custom built 3DoF delta arm robot [Pierrot et al., 1990]. As shown in Figure 2.13 it consists of the arm base, the end effector and three identical and symmetrically placed arms. By construction, the end effector is always parallel to the arm base.

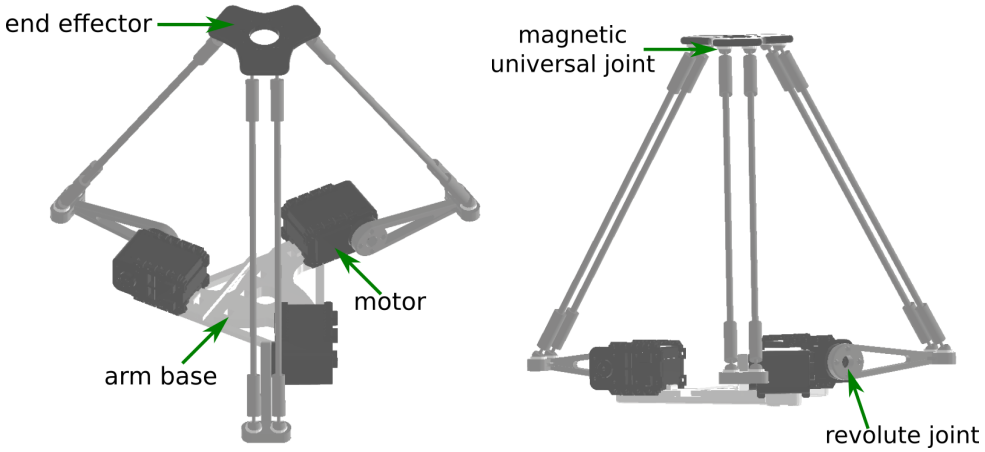


Figure 2.13: The delta arm used in the aerial manipulation experiments.

It is equipped with 3× Dynamixel AX-18A servo actuators which at the most precise control setting achieve closed loop position control with an accuracy –assuming no load– of ±0.29°. Conveniently, they are equipped with position encoders and can provide position feedback in realtime. Communication is achieved through a half duplex serial channel which supports connection of multiple servos with unique IDs. Due to servo working principle, the actuators are less accurate (especially under load) than stepper motors which are widely used in applications that require precision (e.g 3D printers). However, given the limited payload capacity of our MAVs, they were preferred due to their compact size and low weight (55 g).

Another unique feature of our design is the use of magnetic universal joints. They consist of two parts, a 10 mm in diameter neodymium countersunk magnet which provides a pulling force of ≈ 2 kg and an 8 mm in diameter steel ball. Compared to standard universal joints, they provide significant wider range of motion (resulting in an increased workspace) while backlash and friction are minimised. They further enable tool free and fast assembly/disassembly of the different links of the arm. Most importantly, they provide a mechanical point of failure allowing the arm links to disassemble instead of breaking in the event of a crash. Main disadvantage of this

design is that the maximum pulling force acting on the end effector should be less than the combined pulling force of the magnetic joints.

The reachable workspace of the end-effector with a total volume of $1.51 \times 10^{-2}\,\mathrm{m}^3$ is visualised in Figure 2.14. For its computation we consider the mechanical limitations of all the joints (revolute and universal) and not just the maximum and minimum servo motor angles.

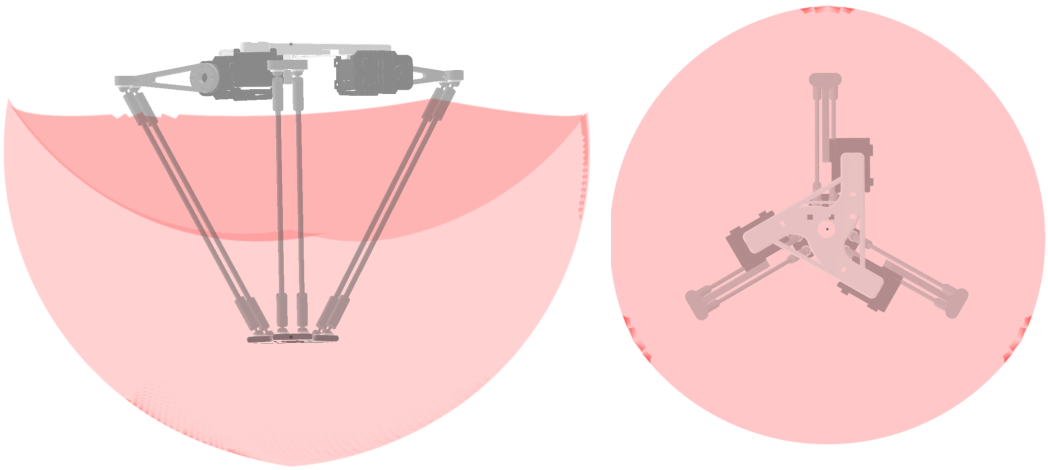


Figure 2.14: The reachable workspace of the end-effector from two different views. We ensure that the workspace volume ($1.51 \times 10^{-2}\,\mathrm{m}^3$ in this setup) is maximised by considering the mechanical limitations of every single joint (revolute and universal) and not just limiting the motion based on the servo motor angle bounds.

2.3.1 Forward kinematics

The forward kinematics problem (i.e. determining the position of the end effector ${}_A\mathbf{r}_E$ given the joint angles $\theta_1, \theta_2, \theta_3$) can be solved by computing the intersection points of three spheres 2.15 of radius l with the following centers:

$$ {}_A\mathbf{r}_{J_1} = \big(R - r + L\cos(\theta_1)\big)\,{}_A\mathbf{e}_x - \sin(\theta_1)\,{}_A\mathbf{e}_z, \tag{2.10a} $$

$$ {}_A\mathbf{r}_{J_2} = \mathbf{C}_z(120^o)\Big(\big(R - r + L\cos(\theta_2)\big)\,{}_A\mathbf{e}_x - \sin(\theta_2)\,{}_A\mathbf{e}_z\Big), \tag{2.10b} $$

$$ {}_A\mathbf{r}_{J_3} = \mathbf{C}_z(240^o)\Big(\big(R - r + L\cos(\theta_3)\big)\,{}_A\mathbf{e}_x - \sin(\theta_3)\,{}_A\mathbf{e}_z\Big), \tag{2.10c} $$

where R, r, L correspond to the arm physical parameters shown in Figure 2.15 with their numeric values given in Table 2.2, ${}_A\mathbf{e}_x = [1,0,0]^\top$, ${}_A\mathbf{e}_z = [0,0,1]^\top$ and $\mathbf{C}_z(120^o), \mathbf{C}_z(240^o)$ rotation matrices of 120^o and 240^o degrees around ${}_A\mathbf{e}_z$. The

solutions can be then obtained by solving the following system of equations:

$$(x - x_{J_1})^2 + (y - y_{J_1})^2 + (z - z_{J_1})^2 = l^2, \tag{2.11a}$$
$$(x - x_{J_2})^2 + (y - y_{J_2})^2 + (z - z_{J_2})^2 = l^2, \tag{2.11b}$$
$$(x - x_{J_3})^2 + (y - y_{J_3})^2 + (z - z_{J_3})^2 = l^2, \tag{2.11c}$$

with ${}_A x_{J_i}, {}_A y_{J_i}, {}_A z_{J_i} \forall i = \{1, 2, 3\}$ the corresponding components of ${}_A\mathbf{r}_{J_i}$ defined in (2.10). As an implementation trick, we can simplify the solution of (2.11) by expressing the centers ${}_A\mathbf{r}_{J_i}$ of the spheres in a coordinate frame $\underrightarrow{\mathcal{F}}_T$ with its origin located at ${}_A\mathbf{r}_{J_1}$ and its orientation relative to the frame $\underrightarrow{\mathcal{F}}_A$ given by:

$$\mathbf{C}_{AT} = \begin{bmatrix} \mathbf{x}_T & \mathbf{y}_T & \mathbf{z}_T \end{bmatrix} = \begin{bmatrix} \frac{{}_A\mathbf{r}_{J_2} - {}_A\mathbf{r}_{J_1}}{\|{}_A\mathbf{r}_{J_2} - {}_A\mathbf{r}_{J_1}\|} & \mathbf{z}_T \times \mathbf{x}_T & \frac{({}_A\mathbf{r}_{J_2} - {}_A\mathbf{r}_{J_1}) \times ({}_A\mathbf{r}_{J_3} - {}_A\mathbf{r}_{J_1})}{\|({}_A\mathbf{r}_{J_2} - {}_A\mathbf{r}_{J_1}) \times ({}_A\mathbf{r}_{J_3} - {}_A\mathbf{r}_{J_1})\|} \end{bmatrix}. \tag{2.12}$$

The sphere centers expressed in the frame $\underrightarrow{\mathcal{F}}_T$ are computed using:

$${}_T\mathbf{r}_{J_i} = \mathbf{C}_{AT}^\top {}_A\mathbf{r}_{J_i} - \mathbf{C}_{AT}^\top {}_A\mathbf{r}_{J_1}, \forall i = \{1, 2, 3\}, \tag{2.13}$$

and the system of equations (2.11) is transformed to:

$$x^2 + y^2 + z^2 = l^2, \tag{2.14a}$$
$$(x - x_{J_2'})^2 + y^2 + z^2 = l^2, \tag{2.14b}$$
$$(x - x_{J_3'})^2 + (y - y_{J_3'})^2 + z^2 = l^2, \tag{2.14c}$$

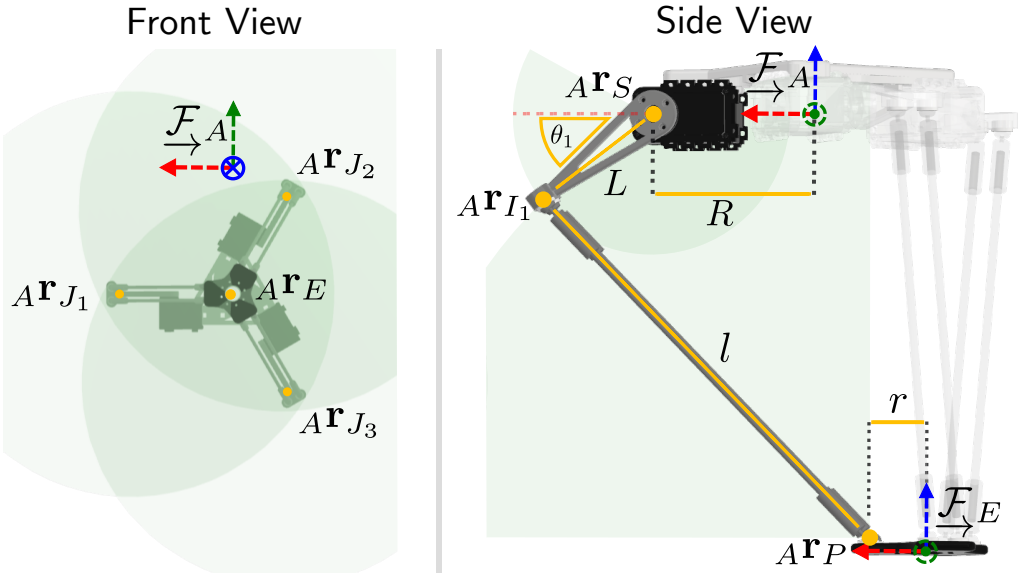


Figure 2.15: Two different views of a 3D model of the delta arm used. The green areas show the virtual spheres and disks used for the solution of the forward and inverse kinematics.

with $y_{J_i'}, z_{J_i'} \forall i = \{2,3\}$ the corresponding components of ${}_T\mathbf{r}_{J_i}$. The end effector position ${}_A\mathbf{r}_E$ can be then computed as:

$${}_T\mathbf{r}_E = \begin{bmatrix} x_E & y_E & z_E \end{bmatrix}^\top = \begin{bmatrix} \frac{x_{J_2'}}{2} & \frac{x_{J_3'}^2 + y_{J_3'}^2 - x_{J_3'} x_{J_2'}}{2y_{J_3'}} & \pm\sqrt{l^2 - x_E^2 - y_E^2} \end{bmatrix}^\top, \tag{2.15a}$$

$${}_A\mathbf{r}_E = \mathbf{C}_{AT}\, {}_T\mathbf{r}_E + {}_A\mathbf{r}_{J_1} \tag{2.15b}$$

Equation 2.15 has up to two solutions (depending on the value of $l^2 - x_E^2 - y_E^2$) corresponding to end effector positions above the arm base or bellow. In our system, configurations with the end effector above the arm base are mechanically impossible and thus eliminated by our kinematics solver.

Description	Symbol	Value	Unit
Arm base radius	R	7.2	cm
End effector radius	r	2.5	cm
Servo link length	L	6.5	cm
Rod length	l	20.2	cm

Table 2.2: Dimensions of the delta arm components.

2.3.2 Inverse kinematics

For the inverse kinematics the intersection between a sphere with radius l and a circular disk with radius L has to be computed for every joint angle. For the first joint as shown in Figure 2.15 the centre of the sphere is ${}_A\mathbf{r}_{P_1} = {}_A\mathbf{r}_E + r\,{}_A\mathbf{e}_x$ with ${}_A\mathbf{e}_x = [1,0,0]^\top$ while the center of the circular disk is ${}_A\mathbf{r}_{S_1} = R\,{}_A\mathbf{e}_x$. Their intersection ${}_A\mathbf{r}_{I_1}$ is computed using the following steps:

1. The 3D sphere is projected onto the plane defined by the circular disk resulting another circular disk with center ${}_A\mathbf{r}'_{P_1}$ and radius l'.
2. The intersections between the two circular disks that lie on the same plane, are computed.

This process results in up to two intersection points. Given an intersection point ${}_A\mathbf{r}_{I_1}$, the joint angle can be recovered as:

$$\theta_1 = \arcsin(z_{I_1}/L), \tag{2.16}$$

with z_{I_1} the z component of ${}_A\mathbf{r}_{I_1}$. Similarly to the forward kinematics case, solutions that produce mechanically impossible configurations are eliminated.

The joint angles θ_2 and θ_3 can be computed by performing the same procedure for the spheres with centers ${}_A\mathbf{r}_{P_2}, {}_A\mathbf{r}_{P_3}$, same radius l and the unit disks centered at ${}_A\mathbf{r}_{S_2}$ and ${}_A\mathbf{r}_{S_3}$ with radius L. The points ${}_A\mathbf{r}_{P_i}, {}_A\mathbf{r}_{S_i}$ $\forall i = 2, 3$ can be easily computed as follows:

$$ {}_A\mathbf{r}_{P_2} = {}_A\mathbf{r}_E + r\ \mathbf{C}_z(120^o)\,{}_A\mathbf{e}_x, \tag{2.17a} $$

$$ {}_A\mathbf{r}_{P_3} = {}_A\mathbf{r}_E + r\ \mathbf{C}_z(240^o)\,{}_A\mathbf{e}_x, \tag{2.17b} $$

$$ {}_A\mathbf{r}_{S_2} = \mathbf{C}_z(120^o)\,{}_A\mathbf{r}_{S_1}, \tag{2.17c} $$

$$ {}_A\mathbf{r}_{S_3} = \mathbf{C}_z(240^o)\,{}_A\mathbf{r}_{S_1}. \tag{2.17d} $$

2.4 State estimation

Precise MAV control relies on accurate state estimation that has to run at a rate equal or greater than the control rate. In this work, depending on the application, we use two different approaches for MAV state estimation. The first is an EKF able to fuse data from the onboard IMU and a generic pose sensor, mainly used in indoor flights with a motion capture system providing the pose measurements. While the second is visual-inertial SLAM used in outdoor operation or when a motion capture system or GPS signal is not available.

2.4.1 EKF for IMU and pose sensor fusion

For fusion of inertial measurements and pose estimates, we use a rather standard error-state Kalman filter formulation similar to the ones presented in [Lynen et al., 2013, Leutenegger et al., 2014]. In short, we aim to estimate the true system state given by:

$$ \mathbf{x}_{\mathrm{R}} := [{}_W\mathbf{r}_S^\top, {}_W\mathbf{v}_S^\top, \mathbf{q}_{WS}^\top, \mathbf{b}_\omega^\top, \mathbf{b}_a^\top]^\top \in \mathbb{R}^6 \times S^3 \times \mathbb{R}^6, \tag{2.18} $$

which consists of the position, linear velocity and orientation of the IMU with frame $\underrightarrow{\mathcal{F}}_S$ with respect to the World fixed frame $\underrightarrow{\mathcal{F}}_W$ as well as the sensor biases. The true system state $\mathbf{x}_{\mathrm{R}}$ is the composition of a nominal state and an error-state $\delta\mathbf{x}_{\mathrm{R}}$. The high frequency IMU measurements can be integrated into the nominal system state that does not take into account the model uncertainty. The update step of the EKF is performed at the arrival of new pose data and yields a Gaussian estimate for the error-state. After this, the error-state mean is incorporated into the nominal state and the whole process is repeated.

IMU model

In our EKF we use the following model for the true IMU dynamics:

$$
\begin{aligned}
{}_W\dot{\mathbf{r}}_S &= {}_W\mathbf{v}_S\,, && \text{(2.19a)}\\
{}_W\dot{\mathbf{v}}_S &= \mathbf{C}_{WS}\,(\tilde{\mathbf{a}} - \mathbf{b}_a + \boldsymbol{w}_a) + {}_W\mathbf{g}, && \text{(2.19b)}\\
\dot{\mathbf{q}}_{WS} &= \frac{1}{2}\boldsymbol{\Omega}(\tilde{\boldsymbol{\omega}} - \mathbf{b}_\omega + \boldsymbol{w}_\omega)\mathbf{q}_{WS}\,, && \text{(2.19c)}\\
\dot{\mathbf{b}}_\omega &= \boldsymbol{w}_{b_\omega}, && \text{(2.19d)}\\
\dot{\mathbf{b}}_a &= -\frac{1}{\tau}\mathbf{b}_a + \boldsymbol{w}_{b_a}, && \text{(2.19e)}
\end{aligned}
$$

where ${}_W\mathbf{g} := [0, 0, -9.81]^\top$ represents the gravitational acceleration. We assume that the IMU accelerometer $\tilde{\mathbf{a}}$ and gyro $\tilde{\boldsymbol{\omega}}$ measurements are biased and disturbed by white Gaussian noise $\boldsymbol{w}_a$ and $\boldsymbol{w}_\omega$ with densities σ_a and σ_ω respectively. We model the gyro bias $\mathbf{b}_\omega$ as random walk and the accelerometer bias $\mathbf{b}_a$ as bounded random walk with time constant $\tau > 0$. The nominal state dynamics correspond to the equations in (2.19) but without the noise terms.

An expression for the error state dynamics can be found using its definition as the difference between the true and the nominal state $\delta\mathbf{x}_\mathrm{R} := \mathbf{x}_\mathrm{R} \boxminus \overline{\mathbf{x}}_R$ and solving for $\delta\dot{\mathbf{x}}_R$. In order to obtain a minimal representation for the error state, we express the orientation error as the quaternion $\delta\mathbf{q} = \overline{\mathbf{q}}^* \otimes \mathbf{q}$ which for small errors can be approximated using the angle vector $\delta\boldsymbol{\theta}$ as $\delta\mathbf{q} \approx [\frac{1}{2}\delta\boldsymbol{\theta}^\top, 1]^\top$. We obtain a minimal error-state representation $\delta\mathbf{x}_\mathrm{R} := [\delta\mathbf{r}^\top, \delta\mathbf{v}^\top, \delta\boldsymbol{\theta}^\top, \delta\mathbf{b}_\omega^\top, \delta\mathbf{b}_a^\top]^\top \in \mathbb{R}^{15}$ around the state $\overline{\mathbf{x}}_R$ with the following non linear dynamics:

$$
\begin{aligned}
\delta\dot{\mathbf{r}} &= \delta\mathbf{v}\,, && \text{(2.20a)}\\
\delta\dot{\mathbf{v}} &= -\overline{\mathbf{C}}_{WB}\,[\tilde{\mathbf{a}} - \mathbf{b}_a]^\times \delta\boldsymbol{\theta} - \overline{\mathbf{C}}_{WB}\,\delta\mathbf{b}_a + \overline{\mathbf{C}}_{WB}\,\boldsymbol{w}_a, && \text{(2.20b)}\\
\delta\dot{\boldsymbol{\theta}} &= -[\tilde{\boldsymbol{\omega}} - \mathbf{b}_\omega]^\times \delta\boldsymbol{\theta} + \delta\mathbf{b}_\omega + \boldsymbol{w}_\omega, && \text{(2.20c)}\\
\delta\dot{\mathbf{b}}_\omega &= \boldsymbol{w}_{b_\omega}, && \text{(2.20d)}\\
\delta\dot{\mathbf{b}}_a &= -\frac{1}{\tau}\delta\mathbf{b}_a + \boldsymbol{w}_{b_a}. && \text{(2.20e)}
\end{aligned}
$$

Since the zero mean accelerometer noise $\boldsymbol{w}_a$ is uniform in all directions, its covariance ellipsoid is a 3D sphere centered at the origin and thus invariant to rotation. Based on that, substituting $\overline{\mathbf{C}}_{WB}\,\boldsymbol{w}_a$ with $\boldsymbol{w}_a$ in (2.20b) results an equivalent expression.

Linearising the expressions in (2.20) with respect to $\delta\mathbf{x}_\mathrm{R}$ around $\overline{\mathbf{x}}_R$ yields:

$$
\delta\dot{\mathbf{x}}_R = \mathbf{F}_c\delta\mathbf{x}_\mathrm{R} + \mathbf{L}_c\boldsymbol{w}, \tag{2.21}
$$

with $\boldsymbol{w} := [\boldsymbol{w}_\omega^\top, \boldsymbol{w}_a^\top, \boldsymbol{w}_{b_\omega}^\top, \boldsymbol{w}_{b_a}^\top]^\top$ the noise vector with covariance $\mathbf{Q}_c := \text{diag}(\sigma_a^2, \sigma_\omega^2, \sigma_{b_\omega}^2, \sigma_{b_a}^2)$ and the Jacobians evaluated at $\bar{\mathbf{x}}_R$ given by:

$$\mathbf{F}_c = \frac{\partial \delta \dot{\mathbf{x}}_R}{\partial \delta \mathbf{x}_\text{R}}, \quad \mathbf{L}_c = \frac{\partial \delta \dot{\mathbf{x}}_R}{\partial \boldsymbol{w}}. \tag{2.22}$$

Observation model

We perform the EKF updates using 6DoF pose estimates originating from a motion capture system. As the motion capture system tracks the pose of an object with an attached frame $\underrightarrow{\mathcal{F}}_V$ which may or may not be perfectly aligned with the IMU frame $\underrightarrow{\mathcal{F}}_S$, we introduce additional calibration states ${}_S\mathbf{r}_V$, $\mathbf{q}_{SV}$ that have no dynamics for the propagation phase. The augmented state and error state are now given by:

$$\mathbf{x}_\text{R} := [{}_W\mathbf{r}_S^\top, {}_W\mathbf{v}_S^\top, \mathbf{q}_{WS}^\top, \mathbf{b}_\omega^\top, \mathbf{b}_a^\top, {}_S\mathbf{r}_V^\top, \mathbf{q}_{SV}^\top]^\top \in \mathbb{R}^6 \times S^3 \times \mathbb{R}^9 \times S^3, \tag{2.23a}$$

$$\delta\mathbf{x}_\text{R} := [\delta\mathbf{r}^\top, \delta\mathbf{v}^\top, \delta\boldsymbol{\theta}_\text{WS}^\top, \delta\mathbf{b}_\omega^\top, \delta\mathbf{b}_a^\top, \delta\mathbf{r}^\top, \delta\boldsymbol{\theta}_\text{SV}^\top]^\top \in \mathbb{R}^{21}. \tag{2.23b}$$

The observation model for the position and orientation is given by:

$$\mathbf{z}_r = {}_W\mathbf{r}_V + \boldsymbol{v}_{\tilde{r}} = \mathbf{C}_{WS}\,{}_S\mathbf{r}_V + {}_W\mathbf{r}_S + \boldsymbol{v}_{\tilde{r}}, \tag{2.24a}$$

$$\mathbf{z}_q = \mathbf{q}_{WV} \otimes \delta\tilde{\mathbf{q}} = \mathbf{q}_{WS} \otimes \mathbf{q}_{SV} \otimes \delta\tilde{\mathbf{q}}\,, \tag{2.24b}$$

with $\boldsymbol{v}_{\tilde{r}} \sim \mathcal{N}(0, \sigma_{\tilde{r}}^2\mathbf{I}), \delta\tilde{\mathbf{q}} = [\frac{1}{2}\delta\boldsymbol{\theta}_{\tilde{q}}, 1], \delta\boldsymbol{\theta}_{\tilde{q}} \sim \mathcal{N}(0, \sigma_{\tilde{q}}^2\mathbf{I})$ modelling the position and orientation measurement noise respectively. We define the position and orientation residuals $\mathbf{y} = [\mathbf{y}_r^\top, \mathbf{y}_q^\top]^\top$ as:

$$\mathbf{y}_r = \mathbf{z}_r - {}_W\bar{\mathbf{r}}_V \tag{2.25a}$$

$$\mathbf{y}_q = 2[\bar{\mathbf{q}}_{WV}^* \otimes \mathbf{z}_q]_{1:3} \tag{2.25b}$$

which given the pose measurements ${}_W\tilde{\mathbf{r}}_V$, $\tilde{\mathbf{q}}_{WV}$ is computed as:

$$\tilde{\mathbf{y}}_r = {}_W\tilde{\mathbf{r}}_V - {}_W\bar{\mathbf{r}}_V = {}_W\tilde{\mathbf{r}}_V - (\bar{\mathbf{C}}_{WS}\,{}_S\bar{\mathbf{r}}_V + {}_W\bar{\mathbf{r}}_S), \tag{2.26a}$$

$$\tilde{\mathbf{y}}_q = 2[\bar{\mathbf{q}}_{WV}^* \otimes \tilde{\mathbf{q}}_{WV}]_{1:3} = 2[(\bar{\mathbf{q}}_{WS} \otimes \bar{\mathbf{q}}_{SV})^* \otimes \tilde{\mathbf{q}}_{WV}]_{1:3} \tag{2.26b}$$

where overbar indicates quantities evaluated using the filter's current estimate.

The Jacobians $\mathbf{H}$ and $\mathbf{V}$ with respect to $\delta\mathbf{x}_\text{R}$ and $\boldsymbol{v} := [\boldsymbol{v}_{\tilde{r}}^\top, \boldsymbol{v}_{\tilde{\mathbf{q}}}^\top]^\top$ used in the filter update are computed by:

$$\mathbf{H} = \frac{\partial \mathbf{y}}{\partial \delta \mathbf{x}_\text{R}}, \quad \mathbf{V} = \frac{\partial \mathbf{y}}{\partial \boldsymbol{v}}. \tag{2.27}$$

The EKF prediction and update steps are outlined bellow, where we use subscripts of the form $m|n$ to represent estimated quantities at time m given observations up to time n with $n \leq m$:

Prediction

1. Given the IMU measurements $\tilde{\mathbf{a}}$, $\tilde{\boldsymbol{\omega}}$ propagate the state $\mathbf{x}$ using (2.19).
2. Calculate the discrete equivalent $\mathbf{F}_k$, $\mathbf{L}_k$ of the Jacobians defined in (2.22).
3. Compute the propagated covariance using: $\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{L}_k \mathbf{Q}_k \mathbf{L}_k^\top$ with $\mathbf{Q}_k$ the discrete equivalent of the covariance matrix $\mathbf{Q}_c$.

Update

1. Given the pose measurements ${}_W\tilde{\mathbf{r}}_V$, $\tilde{\mathbf{q}}_{WV}$ and the current estimate $\overline{\mathbf{x}}$, compute the residual $\tilde{\mathbf{y}}_k$ defined in (2.26).
2. Compute the innovation $\mathbf{S}_{k|k} = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^\top$.
3. Compute the Kalman gain $\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_{k|k}^{-1}$.
4. Update the state estimate $\overline{\mathbf{x}}_{k|k} = \overline{\mathbf{x}}_{k|k-1} \boxplus \mathbf{K}_k \tilde{\mathbf{y}}_k$.
5. Update the state covariance $\mathbf{P}_{k|k} = \big(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\big) \mathbf{P}_{k|k-1}$.

Even though our implementation relies on pose measurements originating from a motion capture system, it can be accordingly modified to fuse information from different sensors (e.g. GPS, pressure sensor, magnetometer). In this case the correct measurement model (depending on the type of sensor) has to be used and if necessary the filter state has to be augmented with sensor-intrinsic and/or calibration states. An example includes the work presented in [Weiss et al., 2012] where a visual SLAM was used as a 6D pose sensor resulting in a loosely coupled visual SLAM capable of providing estimates at IMU rate.

2.4.2 Simultaneous Localisation and Mapping (SLAM)

For operation in environments where a motion capture system or GPS signal is not available, we use Open Keyframe-based Visual Inertial SLAM (OKVIS)[7] as a basis for our MAV state estimation. We made, however, several extensions in order for it to be usable with a low cost RGBD camera.

[7] Available open-source at http://ethz-asl.github.io/okvis/.

OKVIS jointly estimates the robot state $\mathbf{x}_\mathrm{R}$ defined in (2.18) as well as the landmarks $\mathbf{x}_\mathrm{L}$ by minimising the following cost function consisting of camera reprojection errors $\mathbf{e}_\mathrm{r}^{i,j,k}$ and IMU errors $\mathbf{e}_\mathrm{s}^{k}$:

$$J(\mathbf{x}) := \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j\in\mathcal{J}_\mathrm{v}(k)} \mathbf{e}_\mathrm{r}^{i,j,k\,T}\mathbf{W}_\mathrm{r}^{i,j,k}\mathbf{e}_\mathrm{r}^{i,j,k} + \sum_{k=1}^{K-1}\mathbf{e}_\mathrm{s}^{k\,T}\mathbf{W}_\mathrm{s}^{k}\mathbf{e}_\mathrm{s}^{k}, \tag{2.28}$$

with $\mathbf{x} := [\mathbf{x}_\mathrm{R}^\top, \mathbf{x}_\mathrm{L}^\top]^\top$, i the camera index of the I-camera setup, k the camera frame index, j the landmark index, $\mathbf{W}_\mathrm{r}^{i,j,k}$ the information matrix of the corresponding landmark observation and $\mathbf{W}_\mathrm{s}^{k}$ the information matrix of the k^th IMU error. The set $\mathcal{J}_\mathrm{v}(k)$ contains the indices of the landmarks visible in the k^th frame.

In this work, we use a dual camera setup ($I = 2$) which either consists of a Red Green Blue (RGB) stereo pair or an RGB-Depth camera setup shown in Figure 2.9 left and right respectively. For the RGB cameras we use the reprojection error as in the original OKVIS implementation in [Leutenegger et al., 2014]:

$$\mathbf{e}_\mathrm{r}^{i,j,k} = \mathbf{z}^{i,j,k} - \mathbf{h}(\boldsymbol{T}_{C_iS}\,\boldsymbol{T}_{SW}^{k}\,{}_W\mathbf{l}^j), \tag{2.29}$$

with $\mathbf{z}$ denoting the measurement in image coordinates, $\mathbf{h}(\cdot)$ the camera projection model and ${}_W\mathbf{l}^j$ the position of the landmark j in World coordinates.

In the RGB-Depth camera setup, we incorporate the depth information by adopting the approach of ORB-SLAM 2 [Mur-Artal and Tardós, 2017] and creating a *virtual* stereo camera. We obtain virtual keypoint measurements by projecting 3D points into the virtual second camera with frame $\underrightarrow{\mathcal{F}}_{C_v}$. Let $\mathbf{u} = \mathbf{h}({}_C\boldsymbol{r}_P)$ and ${}_C\boldsymbol{r}_P = \boldsymbol{h}^{-1}(\mathbf{u}, \mathbf{D})$ be the RGB camera projection and back-projection from image coordinates to a homogeneous point as defined in Section 2.2.4. We can now create *virtual* keypoint measurements $\mathbf{z}_\mathrm{v}^{j,k}$ from actual measurements $\mathbf{z}^{j,k}$, iff depth is given for said pixel, as:

$$\mathbf{z}_\mathrm{v}^{j,k} = \mathbf{h}_\mathrm{v}(\boldsymbol{T}_{C_\mathrm{v}C}\,\mathbf{h}^{-1}(\mathbf{z}^{j,k}, \mathbf{D})), \tag{2.30}$$

where the transformation $\boldsymbol{T}_{C_\mathrm{v}C}$, and $\mathbf{h}_\mathrm{v}(\cdot)$ stands for the virtual camera projection model that we define as $\mathbf{h}_\mathrm{v}(\cdot) := \mathbf{h}(\cdot)$ for convenience.

Given the *virtual* keypoint measurements $\mathbf{z}_\mathrm{v}^{j,k}$, the reprojection error is defined identical to (2.29). Note that this scheme allows for easy compatibility with a multi-camera system assumed in [Leutenegger et al., 2014]. Furthermore, since the depth map of the camera we use was indeed obtained through stereo triangulation,

formulating reprojection errors in a stereo setup (even if virtual), correctly accounts for noise in image space.

Chapter 3

Linear Model Predictive Control

Contents

Parts of this Chapter appear in: Tzoumanikas, D., Li, W., Grimm, M., Zhang, K., Kovac, M., and Leutenegger, S. (2019). Fully autonomous micro air vehicle flight and landing on a moving target using visual-inertial estimation and model-predictive control. *Journal of Field Robotics*. [Tzoumanikas et al., 2019]

3.1 Introduction

Due to its simple formulation as well as the benefits listed in Chapter 1, the concept of linear MPC has been successfully applied to a variety of aerial vehicles ranging from fixed wing aircraft [Oettershagen et al., 2016] to more relevant to this work multirotor platforms [Alexis et al., 2011, Alexis et al., 2012, Darivianakis et al., 2014, Kamel et al., 2017a, Kamel et al., 2017b]. Its successful deployment on real systems has been enabled by the existence of efficient toolboxes used to solve the underlying optimisation problem either online [Mattingley and Boyd, 2012, Ferreau et al., 2014, Frison and Diehl, 2020] or as an explicit solution in the form of lookup tables [Kvasnica et al., 2004, Herceg et al., 2013].

Here we present a simple linear MPC originally developed for our participation in the MBZIRC competition in 2017 as well as for the purposes of the AABM project. In the control design, we adopt the position-attitude separation discussed in Chapter 1 with the position MPC generating reference attitude commands tracked by an already implemented attitude controller. The control model was built, with minor adaptations, based on previously published works backed up with experimental results.

In addition to the position controller, we further present the experimental setup used for our participation in MBZIRC. This includes a generic framework for vision based detection and tracking, which although tested in an experimental scenario resembling the MBZIRC Challenge 1, can be used in more generic tracking applications while exclusively relying on onboard sensors.

3.2 Related work

Amongst the first applications of linear MPC in MAVs, we find the work in [Alexis et al., 2011] where an attitude controller was proposed. The MAV nonlinear attitude dynamics were approximated as a sequence of linear systems (each one linearised around different operating points) resulting in a switching mode MPC based on the current MAV state and the active subsystem. The same dynamics approximation principle was applied for the position and attitude control of an MAV in [Alexis et al., 2012]. The two controllers were connected in a cascaded way with the position MPC providing the attitude references to the attitude MPC. The reference attitude commands were assumed to be perfectly tracked by the attitude controller.

A better approximation, as a second order system, regarding the closed loop attitude dynamics was adopted in [Darivianakis et al., 2014]. There the same cascaded control architecture was used, with the attitude controller being a pre-tuned PID. In a very similar approach [Kamel et al., 2017a, Kamel et al., 2017b] the closed loop attitude dynamics were approximated as a first order system. Another difference between the approach in [Darivianakis et al., 2014] and [Kamel et al., 2017a, Kamel et al., 2017b] is that the linearised position dynamics used in the former include linear damping terms (e.g. to model the effect of aerodynamic friction) whereas the latter approach relies on the model derived from linearisation at hover. A much simpler modelling approach, but with very good experimental results, was adopted in [Beul et al., 2017] where the position dynamics were modelled as a per-axis triple integrator system with the position MPC producing jerk commands. These were integrated into acceleration and subsequently, assuming near hover operation, to orientation commands.

As far as the idea of landing aerial vehicles on moving platforms is concerned, this has been documented in several works as it can be e.g. used as a failsafe mechanism in case of a landing gear malfunction [Muskardin et al., 2016], or more relevant to MAVs observation of moving objects [Thomas et al., 2017] and in package delivery [Murray and Chu, 2015, Ham, 2018] where the aerial vehicle has to repeatedly takeoff from and return to a delivery truck upon completing a delivery task. Traditionally, the problem of landing on a moving platform has been either handled as a cooperative control task (e.g. control of both vehicles in a centralised way or exchange of data between the two) or as a tracking problem using onboard sensors such as cameras which is also the focus of our work.

Regarding the vision based tracking, the proposed methods rely on the specific experimental setup with e.g. tracking of a spherical object in [Thomas et al., 2017], In the most common scenario, determining the relative pose between the MAV and the moving platform can be achieved using a single fiducial marker (e.g. [Kaess, 2013]) attached to the latter as in [Shuo Yang et al., 2015, Lee et al., 2012] or in a different approach such as [Araar et al., 2017], multiple ones at different scales aiming at handling occlusions and increasing the detection range. Alternative ways of ensuring visibility of the landing platform after initial detection include the method in [Vlantis et al., 2015] where the visibility requirement is considered by the tracking controller, the method in [Lee et al., 2012] where the control law (ensuring

visibility) is formulated in image space as well as the hardware-enabled approach in [Bhargavapuri et al., 2019] where the omnidirectional vehicle used can translate without altering its orientation and thus not affecting the camera's field of view. In order to further increase detection robustness, most of the works with successful experimental results [Araar et al., 2017, Vlantis et al., 2015] rely on the fusion (usually in an EKF framework) of the vision observations with some motion model of the moving target. This further enables the estimation of the moving target velocity which can be used in the tracking controller and plays a crucial role for successful tracking and landing especially at high speed.

Finally, as examples of complete software and hardware frameworks, we would also like to highlight the work of other Mohamed Bin Zayed International Robotics Challenge (MBZIRC) participants [Bähnemann et al., 2017, Beul et al., 2017] with a direct comparison of their approach to ours being provided in Section 3.6.5.

3.3 Contribution

In this Chapter we present a linear MPC for MAV position control with the objective of accuracy (given the limitations of the linear modelling) and ease of use across different platforms. In short, we show the following contributions:

- A position MPC which relies on a linear model similar to the one presented in [Darivianakis et al., 2014, Kamel et al., 2017a], incorporates soft state constraints and is formulated as a QP in its canonical form that can be solved in realtime using any generic QP solver.
- A generic framework for vision based detection, tracking and landing on a moving target. We evaluate its performance in experiments emulating the MBZIRC 2017 Challenge 1.
- Experimental evaluation of the tracking accuracy in slow maneuvers as required for the purposes of the AABM project.

3.4 System overview

A generic system overview which is common in both the experiments presented in Section 3.6 as well as the ones in Section 3.7, is given in Figure 3.1. Main compon-

ents include: (i) the state estimation block which provides full state estimates and depending on the experiment's environment (indoor/outdoor) either relies on vision or on fusion of IMU and pose measurements provided by a motion capture system as described in Section 2.4 (ii) the linear MPC which receives trajectory commands and produces orientation and collective thrust reference commands which are forwarded to (iii) the attitude controller which is run by the flight controller and produces PWM commands.

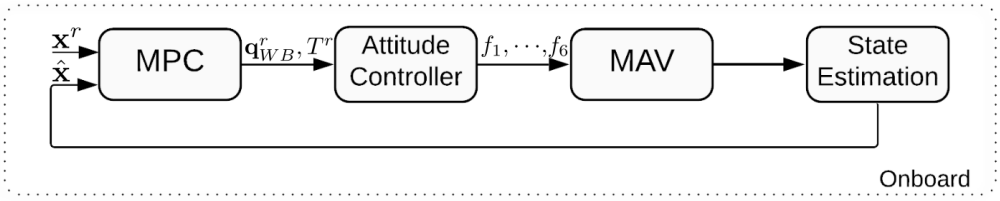


Figure 3.1: A generic system overview of the various software components used in the Experiments presented in Sections 3.6 and 3.7.

The coordinate frames used in the control formulation are shown in Figure 3.2. All motion is referenced relative to a World-frame $\underrightarrow{\mathcal{F}}_W$ (Earth-fixed and tangential to the surface with z-axis upward) that we approximate to be also an inertial frame. We further consider an MAV body fixed frame $\underrightarrow{\mathcal{F}}_B$ with its origin at the CoM of the vehicle. We formulate the MAV dynamics in a frame $\underrightarrow{\mathcal{F}}_N$ which is obtained when the frame $\underrightarrow{\mathcal{F}}_W$ is rotated by the yaw angle ψ around its z axis and translated to the origin of the body frame $\underrightarrow{\mathcal{F}}_B$.

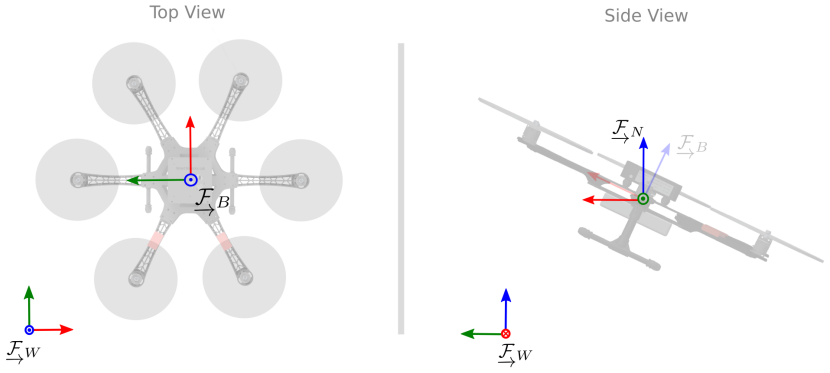


Figure 3.2: Illustration of the different coordinate frames from two different views. $\underrightarrow{\mathcal{F}}_W$: The World frame approximated as inertial where all motion is reference to. $\underrightarrow{\mathcal{F}}_B$: The MAV body fixed frame. $\underrightarrow{\mathcal{F}}_N$: The frame used for the control model formulation.

3.4.1 Translational dynamics

The linear model used for control uses Euler angles (following the ZYX convention with yaw $\psi \in [-\pi, \pi]$, pitch $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, roll $\phi \in [-\pi, \pi]$) as a parameterisation for the MAV orientation. Throughout this work quaternions were always used for the state estimation pipeline as this has to work for any possible motion or control algorithm not bound to the application at hand. In the linear MPC case, Euler angles were preferred as it was easier to derive the corresponding linear model. We ensure that the MAV operates at angles far from the gimbal lock through appropriate optimisation constraints.

Since the heading angle ψ does not contribute to the translational motion of the MAV and in order to eliminate it from the model used for control, we express the MAV position and linear velocities in the navigation frame $\underrightarrow{\mathcal{F}}_N$ shown in Figure 3.2. We consider the following control state and input:

$$\mathbf{x} := [x, \dot{x}, \theta, \dot{\theta}, y, \dot{y}, \phi, \dot{\phi}, z, \dot{z}]^\top, \tag{3.1a}$$

$$\mathbf{u} := [\theta^r, \phi^r, T^r]^\top, \tag{3.1b}$$

with ${}_N\mathbf{r}_B := [x, y, z]^\top$, ${}_N\mathbf{v}_B := [\dot{x}, \dot{y}, \dot{z}]^\top$ the MAV position and linear velocity expressed in the frame $\underrightarrow{\mathcal{F}}_N$ and θ, ϕ, $\dot{\theta}$, $\dot{\phi}$ the Euler angles and their respective derivatives. The superscipt r used in (3.1b) denotes the corresponding reference quantity (e.g. θ^r is the pitch angle commanded to the attitude controller while θ the one actually achieved). Estimates of ${}_N\mathbf{r}_B$ and ${}_N\mathbf{v}_B$ can be computed by appropriately transforming the world frame expressed ${}_W\mathbf{r}_B$, ${}_W\mathbf{v}_B$ while θ and ϕ by converting the estimated quaternion $\mathbf{q}_{WB}$ to Euler angles. Regarding the Euler angle time derivatives $\dot{\theta}$, $\dot{\phi}$, these can be computed using the following:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \end{bmatrix}}_{\mathbf{J}} {}_B\boldsymbol{\omega}\,, \tag{3.2}$$

with ${}_B\boldsymbol{\omega}$ the estimated rotational velocity of the MAV. Note that the Jacobian $\mathbf{J}$ is undefined for the boundary case $\theta = \pm\frac{\pi}{2}$ further signifying the necessity of operation away from the gimbal lock or ultimately the use of a quaternion based model such as the one introduced in Chapter 4.

Regarding the translational dynamics, we use the following linear model:

$$ {}_N\dot{\mathbf{v}} = \begin{bmatrix} g\theta - c_x\dot{x} \\ -g\phi - c_y\dot{y} \\ T^r - c_z\dot{z} \end{bmatrix}, \tag{3.3} $$

with $g = 9.81$ m/s^2 the gravitational acceleration constant and T^r the desired vertical acceleration which we will define later. The linear model was derived by assuming that the MAV is in a near hover operation and that the translational dynamics are controlled by the projection of the MAV thrust force on the $\underrightarrow{\mathcal{F}}_N$ in order to generate translational acceleration. The terms $c_i\dot{i}$, $\forall i \in \{x, y, z\}$ model the system damping (approximated as linear) due to aerodynamic friction.

Regarding the closed loop attitude dynamics, we assume that these can be approximated by the following second order system:

$$ \ddot{\theta} = -b_{\ddot{\theta}\theta}\theta - b_{\ddot{\theta}\dot{\theta}}\dot{\theta} + b_{\theta^r}\theta^r, \tag{3.4a} $$

$$ \ddot{\phi} = -b_{\ddot{\phi}\phi}\phi - b_{\ddot{\phi}\dot{\phi}}\dot{\phi} + b_{\phi^r}\phi^r. \tag{3.4b} $$

Regarding the motor dynamics, it was assumed that these are significantly faster than the close loop attitude and translational dynamics and can thus be ignored. This means that a desired thrust command can be instantly achieved by the motors. The relationship between the desired vertical acceleration T^r and the one commanded to the flight controller $\tilde{T^r}$ is given by:

$$ \tilde{T^r} = T^r + T_{ff}, \tag{3.5} $$

with $T_{ff} = g$, a feed forward term which compensates the acceleration due to gravity and can be easily identified by doing hovering experiments for which $\ddot{z} \approx 0$.

By combining (3.3) and (3.4) we can express the system dynamics in the following continuous time state space representation:

$$ \dot{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{A}_{\text{Lon}} & \mathbf{0}_{4\times4} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{4\times4} & \mathbf{A}_{\text{Lat}} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times4} & \mathbf{0}_{2\times4} & \mathbf{A}_{\text{Alt}} \end{bmatrix}}_{\mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix} \mathbf{B}_{\text{Lon}} & \mathbf{0}_{4\times1} & 0 \\ \mathbf{0}_{4\times1} & \mathbf{B}_{\text{Lat}} & 0 \\ \mathbf{0}_{2\times1} & \mathbf{0}_{2\times1} & \mathbf{B}_{\text{Alt}} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}, \tag{3.6a} $$

$$ \mathbf{y} = \mathbf{C}\mathbf{x}, \tag{3.6b} $$

where $\mathbf{C} = \mathbf{I}_{10\times10}$ and the submatrices $\mathbf{A}_{\text{Lon}}$, $\mathbf{A}_{\text{Lat}}$, $\mathbf{A}_{\text{Alt}}$, $\mathbf{B}_{\text{Lon}}$, $\mathbf{B}_{\text{Lat}}$, $\mathbf{B}_{\text{Alt}}$ are given by the following:

$$\mathbf{A}_{\text{Lon}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -c_x & g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -b_{\ddot{\theta}\theta} & -b_{\ddot{\theta}\dot{\theta}} \end{bmatrix}, \ \mathbf{B}_{\text{Lon}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ b_{\theta^r} \end{bmatrix}, \tag{3.7}$$

$$\mathbf{A}_{\text{Lat}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -c_y & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -b_{\ddot{\phi}\phi} & -b_{\ddot{\phi}\dot{\phi}} \end{bmatrix}, \ \mathbf{B}_{\text{Lat}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ b_{\phi^r} \end{bmatrix}, \tag{3.8}$$

$$\mathbf{A}_{\text{Alt}} = \begin{bmatrix} 0 & 1 \\ 0 & -c_z \end{bmatrix}, \ \mathbf{B}_{\text{Alt}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{3.9}$$

Since the controller is implemented in discrete time the above equations are discretised using zero order hold for the input $\mathbf{u}$. The discrete equivalent of the matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ can be obtained by:

$$\mathbf{A}_d = e^{\mathbf{A}dt}, \tag{3.10a}$$

$$\mathbf{B}_d = \Big(\int_0^{dt} e^{\mathbf{A}\tau} d\tau \Big) \mathbf{B}, \tag{3.10b}$$

$$\mathbf{C}_d = \mathbf{C}. \tag{3.10c}$$

3.4.2 System identification

The linear dynamics defined in (3.3) and (3.4) depend on the unknown constant parameters c_i and b_i. These were experimentally identified using frequency domain grey-box identification with data captured from manual flights. An accurate system identification would require the application of a chirp signal, which would expose the dominant frequencies of the system. This would yet lead to an unsafe experiment where the MAV rotates according to the chirp input but its position remains uncontrolled, and that is why the manual flight was preferred.

Figure 3.3 shows the real and the simulated response of the identified closed loop attitude dynamics when the inputs of the validation dataset are used and also the real and simulated response for the translational dynamics.

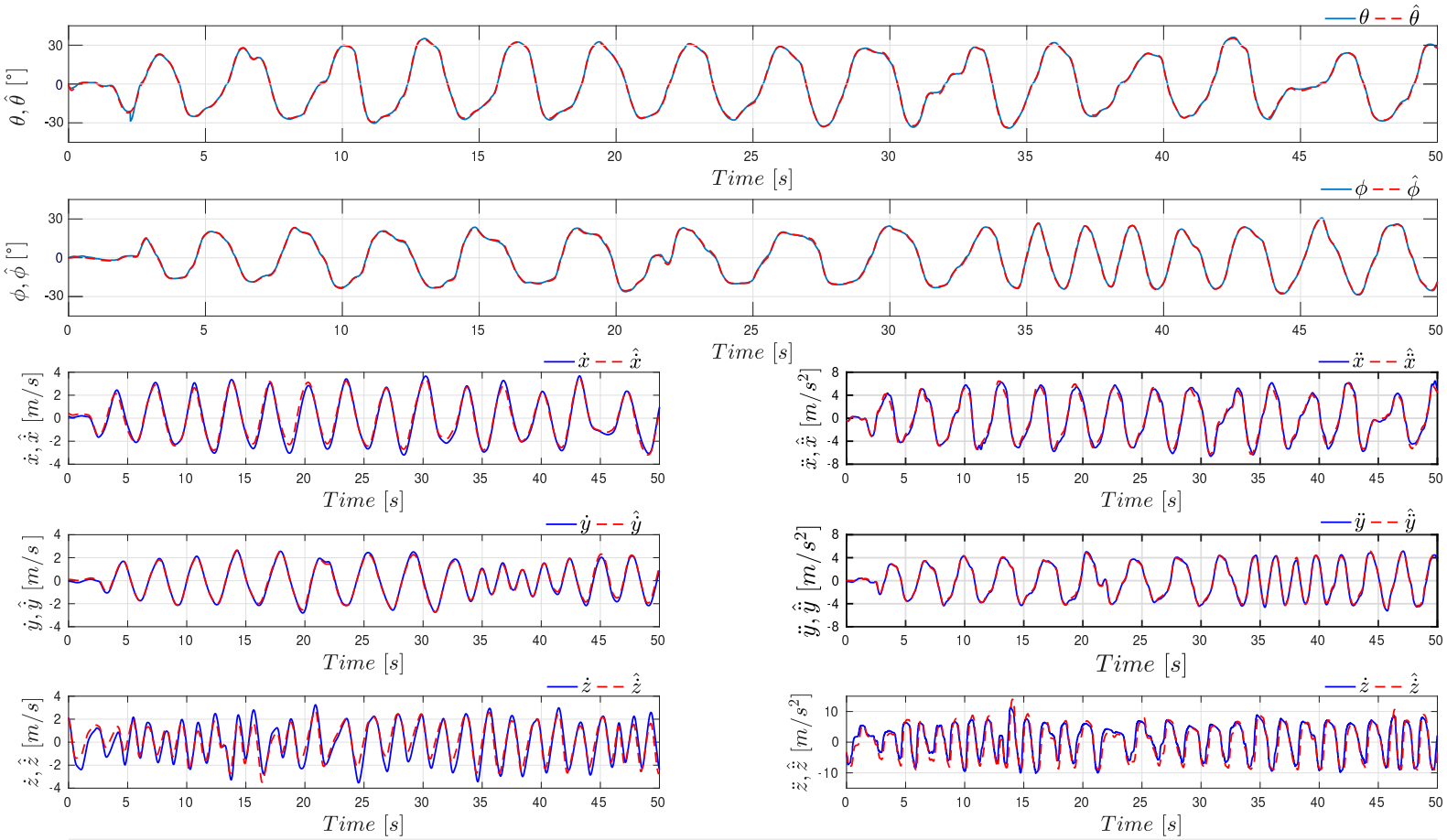


Figure 3.3: Real and simulated response for θ, ϕ, $\dot{x}$, $\dot{y}$, $\dot{z}$, $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$. Despite the use of a linear model, the simulated and the real attitude response (upper two plots) of the MAV almost always coincide. The simulated translational dynamics best match the real ones in the low velocity ($< 2.0\,\mathrm{m/s}$) and low acceleration ($< 1.0\,\mathrm{m/s^2}$) region.

Apart from the attitude and translational dynamics, we perform an experimental identification of the thrust command $\tilde{T^r}$ defined in (3.5) and the one actually achieved by the MAV. This is necessary as the flight controller interface does not support collective thrust commands in native units of thrust (i.e. Newtons) but only allows normalised collective thrust commands in the $[0, 1]$ range. Our model also considers the effect (approximated as linear) of the varying battery voltage. An example of the relationship between the normalised command for hovering thrust $\tilde{T^r} \approx T_{ff}$ and the battery voltage is given in Figure 3.4.

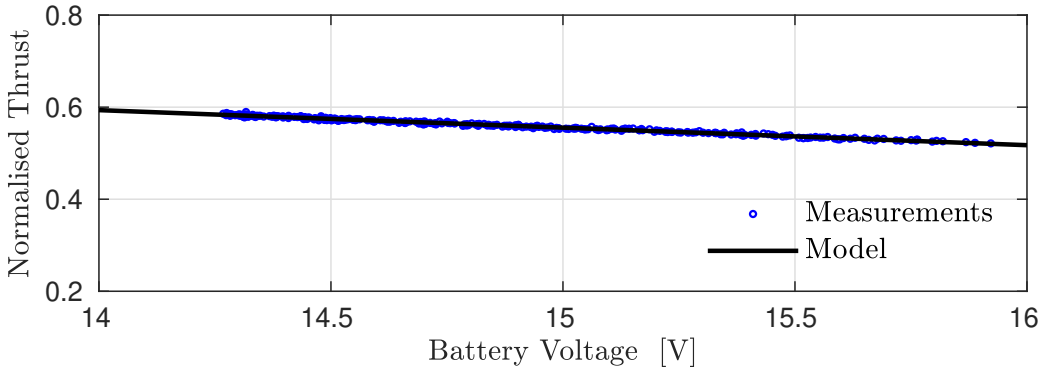


Figure 3.4: Experimental identification of the relationship between the normalised normalised command for hovering thrust $\tilde{T^r} \approx T_{ff}$ and the battery voltage.

3.5 MPC with soft constraints

The implemented MPC computes the optimal input sequence $\bar{\mathbf{u}}^* = \mathbf{u}_0^* \dots \mathbf{u}_{N-1}^*$ which is the solution of the following optimization problem:

$$\begin{aligned} \bar{\mathbf{u}}^* = & \underset{\mathbf{u}_0 \dots \mathbf{u}_{N-1}}{\operatorname{argmin}} J, \\ \text{s.t. } : & \mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k, \\ & \mathbf{y}_k = \mathbf{C}_d \mathbf{x}_k, \\ & \mathbf{x}_0 = \hat{\mathbf{x}}_0, \\ & \bar{\mathbf{u}}_{min} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{max}, \end{aligned} \tag{3.11}$$

where: $\mathbf{x}_k \in \mathbb{R}^n$ is the system state at time k, $\hat{\mathbf{x}}_0 \in \mathbb{R}^n$ the estimated state at time 0, $\mathbf{y}_k \in \mathbb{R}^p$ the system output at time k, $\mathbf{s}_k^y \in \mathbb{R}^p$ the reference output at time k, $\mathbf{s}_k^u \in \mathbb{R}^m$ is the reference input at time k, $N \in \mathbb{Z}^+$ the length of the prediction

horizon and $\mathbf{A}_d \in \mathbb{R}^{n\times n}, \mathbf{B}_d \in \mathbb{R}^{n\times m}, \mathbf{C}_d \in \mathbb{R}^{p\times n}$are the discrete state, input and output transition matrices as defined in (3.10).

The cost function

$$J = \sum_{k=0}^{N-1} \left(\left\|\mathbf{Q}_{k+1}(\mathbf{y}_{k+1} - \mathbf{s}^y_{k+1})\right\|_2^2 + \|\mathbf{R}_k(\mathbf{u}_k - \mathbf{s}^u_k)\|_2^2 \right) \tag{3.12}$$

is the quadratic penalty function on the states and inputs commonly used in optimal control, where the input and output gain matrices $\mathbf{R}_k \in \mathbb{R}^{m\times m}$ and $\mathbf{Q}_k \in \mathbb{R}^{n\times n}$ are tuning parameters. By concatenating the two squared 2-norms that appear in the cost function J, we can rewrite it as:

$$J = \left\| \begin{array}{c} \mathbf{Q}_1(\mathbf{y}_1 - \mathbf{s}^y_1) \\ \mathbf{Q}_2(\mathbf{y}_2 - \mathbf{s}^y_2) \\ \vdots \\ \mathbf{Q}_N(\mathbf{y}_N - \mathbf{s}^y_N) \\ \mathbf{R}_0(\mathbf{u}_0 - \mathbf{s}^u_0) \\ \mathbf{R}_1(\mathbf{u}_1 - \mathbf{s}^u_1) \\ \vdots \\ \mathbf{R}_{N-1}(\mathbf{u}_{N-1} - \mathbf{s}^u_{N-1}) \end{array} \right\|_2^2 \tag{3.13}$$

and by assigning $\bar{\mathbf{y}} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N]^T$, $\bar{\mathbf{u}} = [\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{N-1}]^T$, $\bar{\mathbf{s}}^y = [\mathbf{s}^y_1, \mathbf{s}^y_2, \cdots, \mathbf{s}^y_N]^T$, $\bar{\mathbf{s}}^u = [\mathbf{s}^u_0, \mathbf{s}^u_1, \cdots, \mathbf{s}^u_{N-1}]^T$, $\bar{\mathbf{Q}} = \mathrm{diag}(\mathbf{Q}_1, \mathbf{Q}_2, \cdots, \mathbf{Q}_N)$ and $\bar{\mathbf{R}} = \mathrm{diag}(\mathbf{R}_0, \mathbf{R}_1, \cdots, \mathbf{R}_{N-1})$ the original optimization problem, takes the following form:

$$\begin{aligned} \bar{\mathbf{u}}^* &= \underset{\mathbf{u}_0 \cdots \mathbf{u}_{N-1}}{\mathrm{argmin}} \left\| \begin{array}{c} \bar{\mathbf{Q}}(\bar{\mathbf{y}} - \bar{\mathbf{s}}^y) \\ \bar{\mathbf{R}}(\bar{\mathbf{u}} - \bar{\mathbf{s}}^u) \end{array} \right\|_2^2, \\ &\text{s.t. } : \mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k, \\ &\mathbf{y}_k = \mathbf{C}_d\mathbf{x}_k, \\ &\mathbf{x}_0 = \hat{\mathbf{x}}_0, \\ &\bar{\mathbf{u}}_{min} \le \bar{\mathbf{u}} \le \bar{\mathbf{u}}_{max}. \end{aligned} \tag{3.14}$$

We can eliminate the model equality constraints from (3.14) by substituting $\bar{\mathbf{y}} = \bar{\mathbf{C}}\bar{\mathbf{x}} = \bar{\mathbf{C}}\mathbf{\Phi}\hat{\mathbf{x}} + \bar{\mathbf{C}}\mathbf{\Gamma}\bar{\mathbf{u}}$ into the cost function J, where $\bar{\mathbf{x}} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]^T$ and the matrices $\mathbf{\Phi} \in \mathbb{R}^{Nn\times n}$ and $\mathbf{\Gamma} \in \mathbb{R}^{Nn\times Nm}$ can be obtained by applying the equations $\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k$ and $\mathbf{y}_k = \mathbf{C}_d\mathbf{x}_k$ to every element of $\bar{\mathbf{x}}$. Specifically, these are

given by:

$$\mathbf{\Phi} = \begin{pmatrix} \mathbf{A}_d \\ \mathbf{A}_d^2 \\ \vdots \\ \mathbf{A}_d^N \end{pmatrix}, \ \mathbf{\Gamma} = \begin{pmatrix} \mathbf{B}_d & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}_d\mathbf{B}_d & \mathbf{B}_d & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_d^{N-1}\mathbf{B}_d & \mathbf{A}_d^{N-2}\mathbf{B}_d & \cdots & \mathbf{B}_d \end{pmatrix}. \tag{3.15}$$

The optimization problem is transformed into the following equivalent QP with inequality constraints:

$$\begin{aligned} \bar{\mathbf{u}}^* = \underset{\bar{\mathbf{u}}}{\operatorname{argmin}} \ & \bar{\mathbf{u}}^T \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix}^T \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix} \bar{\mathbf{u}} - 2 \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{s}}^y - \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Phi}\hat{\mathbf{x}}_0 \\ \bar{\mathbf{R}}\bar{\mathbf{s}}^u \end{pmatrix}^T \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix} \bar{\mathbf{u}} \\ \text{s.t.} \ : & \ \bar{\mathbf{u}}_{min} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{max}. \end{aligned} \tag{3.16}$$

In order to ensure operation within a safe state envelope, it is common in MPC to impose additional state constraints. These can be modeled as hard constraints similar to the input constraints $\bar{\mathbf{u}}_{min} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{max}$. In this case the optimization solver may face an infeasible problem since the set of admissible points as defined by the problem constraints is empty e.g. when a large disturbance has occurred or when the real and the estimated model used for control behave differently.

In order to avoid the case of infeasibility we model the state constraints $\mathbf{G}\bar{\mathbf{x}} \leq \mathbf{h}$, with $\mathbf{G} \in \mathbb{R}^{Nl \times Nn}$ as soft constraints which can be violated if necessary. We incorporate them into the cost function J following a similar approach discussed in [Maciejowski, 2002] and [Kerrigan and Maciejowski, 2000].

The modified cost function J_m is:

$$J_m = J + \lambda \mathbf{1}^{\mathbf{T}} (\mathbf{G}\bar{\mathbf{x}} - \mathbf{h})_+. \tag{3.17}$$

The subscript + implies that $(\mathbf{G}\bar{\mathbf{x}} - \mathbf{h})_+ = \mathbf{G}\bar{\mathbf{x}} - \mathbf{h}$ when $\mathbf{G}\bar{\mathbf{x}} - \mathbf{h} \geq \mathbf{0}$ and $\mathbf{0}$ otherwise. Overall, the term $\mathbf{1}^{\mathbf{T}}(\mathbf{G}\bar{\mathbf{x}} - \mathbf{h})_+$ is the sum of constraints violation while the gain $\lambda \in \mathbb{R}$ is large enough in order to ensure that the modified optimization problem with soft constraints, when none of the constraints is active, is equivalent to the optimization problem where the state constraints are modeled as hard.

The optimisation problem with the soft state constraints can be rewritten as the

following equivalent QP by introducing the slack variables $\mathbf{s}$.

$$\begin{aligned}\bar{\mathbf{u}}^*, \mathbf{s}^* &= \underset{\bar{\mathbf{u}},\mathbf{s}}{\operatorname{argmin}}\, J + \lambda \mathbf{1}^T \mathbf{s} \\ \text{s.t. :} &\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} \\ \mathbf{G\Gamma} & -\mathbf{I} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \bar{\mathbf{u}} \\ \mathbf{s} \end{pmatrix} \leq \begin{pmatrix} \bar{\mathbf{u}}_{max} \\ -\bar{\mathbf{u}}_{min} \\ \mathbf{h} - \mathbf{G\Phi}\hat{\mathbf{x}}_0 \\ \mathbf{0} \end{pmatrix}\end{aligned} \tag{3.18}$$

By introducing $\mathbf{t} = [\bar{\mathbf{u}}, \mathbf{s}]^T \in \mathbb{R}^{N(l+m)}$ the above QP is written in its canonical form as:

$$\begin{aligned}\mathbf{t}^* = \underset{\mathbf{t}}{\operatorname{argmin}}\ \mathbf{t}^\top &\begin{bmatrix} \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix}^T \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{t} + \begin{bmatrix} -2 \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{s}}^y - \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Phi}\hat{\mathbf{x}}_0 \\ \bar{\mathbf{R}}\bar{\mathbf{s}}^u \end{pmatrix}^T & \lambda \mathbf{1}^\top \end{bmatrix} \mathbf{t}, \\ \text{s.t. :} &\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} \\ \mathbf{G\Gamma} & -\mathbf{I} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix} \mathbf{t} \leq \begin{pmatrix} \bar{\mathbf{u}}_{max} \\ -\bar{\mathbf{u}}_{min} \\ \mathbf{h} - \mathbf{G\Phi}\hat{\mathbf{x}}_0 \\ \mathbf{0} \end{pmatrix}.\end{aligned} \tag{3.19}$$

This can be solved in realtime using any generic QP solver. We use CVXGEN [Mattingley and Boyd, 2012] which generates a tailored to the specific problem interior-point based C code and in practice was the fastest QP solver tested. According to the authors of the optimization toolbox, nearly all the computational effort in each iteration stems from the solution of two linear systems resulting in a worst case computation complexity of $\mathcal{O}(n^3)$ where $n = N(l+m)$ corresponds to the number of optimization variables in the final QP problem and N, l and m to the length of the prediction horizon, the number of soft constraints and the number of inputs, respectively, as introduced above.

3.6 Experimental results: Landing on a moving platform at MBZIRC

In this Section we present the additional software components developed for the purposes of the MBZIRC were we participated in 2017 and corresponding experimental results. The challenge required an MAV to autonomously detect and land (within 15 minutes) on a flat ferrous target (1.5 m $\times$ 1.5 m) placed on top of a moving ground

vehicle. The moving ground vehicle had a speed of 15 km/h for the first 8 minutes and 5 km/h for the next 7. A landing was considered successful only if the MAV neither had visible damage nor fell off from the target. Performing the task in an autonomous way is challenging, as it requires fast response from sensing, precise state estimation, as well as good synchronisation to other subsystems i.e. controller and target tracking.

Rather than custom-tailoring an approach to the specific challenge, we developed and integrated a suite of algorithms that allow us fully autonomous flight in a wider range of scenarios and conditions, including the ones of MBZIRC. In the design of both the platform and the software stack, we decided for a low-cost, yet robust solution. Specifically, we adopt an extended visual-inertial odometry framework, OKVIS [Leutenegger et al., 2014], providing the basis for control, even if GPS is either not available or not reliable enough. A downward-looking fisheye camera is used for detection and tracking of a moving target, formulated as a tracking problem in 3D space. We close the position control loop with the MPC described in Section 3.5, which has proven extremely robust even in windy conditions. The overall system performance is evaluated in outdoor test flights simulating the MBZIRC challenge. We show that the proposed setup is capable of landing on a target moving at up to 18 km/h in presence of wind in the order of 15 km/h.

3.6.1 System overview

Coordinate frames

Apart from the World $\underrightarrow{\mathcal{F}}_W$ and body frame $\underrightarrow{\mathcal{F}}_B$ introduced in Figure 3.2, we further introduce the moving target frame $\underrightarrow{\mathcal{F}}_T$, the RGBD camera frame $\underrightarrow{\mathcal{F}}_C$, the downward-looking camera frame $\underrightarrow{\mathcal{F}}_D$ and the IMU coordinate frame $\underrightarrow{\mathcal{F}}_S$. The different coordinate frames used are illustrated in Figure 3.5.

Hardware components

For the experiments presented in this Section, we use the custom built hexacopter equipped with the onboard computer and flight controller described in Section 2.2. In order to cope with the additional payload we used the more efficient propulsion system #1 presented in Section 2.2.3.

We use two different camera sensors, one for MAV state estimation and an-

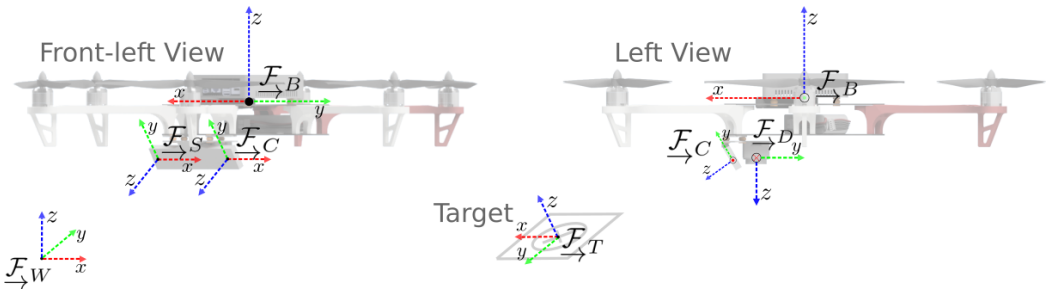


Figure 3.5: The different coordinate frames used. Namely, $\underrightarrow{\mathcal{F}}_W$: the Earth fixed frame, $\underrightarrow{\mathcal{F}}_B$: the MAV body fixed frame, $\underrightarrow{\mathcal{F}}_C$: the RGB-D camera frame, $\underrightarrow{\mathcal{F}}_S$: the IMU sensor frame, $\underrightarrow{\mathcal{F}}_D$: the downward looking camera frame and $\underrightarrow{\mathcal{F}}_T$: the moving target frame.

other for estimating the target position and velocity. Specifically, the RealSense ZR300 RGBD+IMU[1] sensor is used for the state estimation scheme while a FLIR Chameleon 3[2] is mounted on the lower plate of the MAV and is used for the target tracking. Special attention was payed on the mechanical assembly of the RGBD+IMU sensor which was soft mounted on the MAV frame to prevent the mechanical vibrations of the motors from significantly affecting the natively noisy IMU measurements which are used in OKVIS.

In order to absorb the impact energy when the MAV performs high speed landing maneuvers, an origami-folding inspired landing pad made from soft materials was designed using multi-material additive manufacturing techniques. The landing pad is attached to each of the MAV arms, consists of compliant hinges with geometry in single curvature shell and morphing shells which can be passively bent by the impact forces when the six feet make contact with the landing target. The MAV with all the extra hardware components including the landing mechanisms are illustrated in Figure 3.6.

Software components

On the software side there are three main components. The visual-inertial odometry is responsible for the estimation of the MAV position, orientation and the respective velocities. A target tracking EKF is responsible for the estimation of the landing target position and velocity. This information is later used by the MPC presented in Section 3.5 in order to stabilize the MAV and navigate it accordingly to a desired

[1] See https://click.intel.com/realsense.html. Accessed April 2020.
[2] See https://www.ptgrey.com/chameleon3-usb3-vision-cameras. Accessed April 2020.

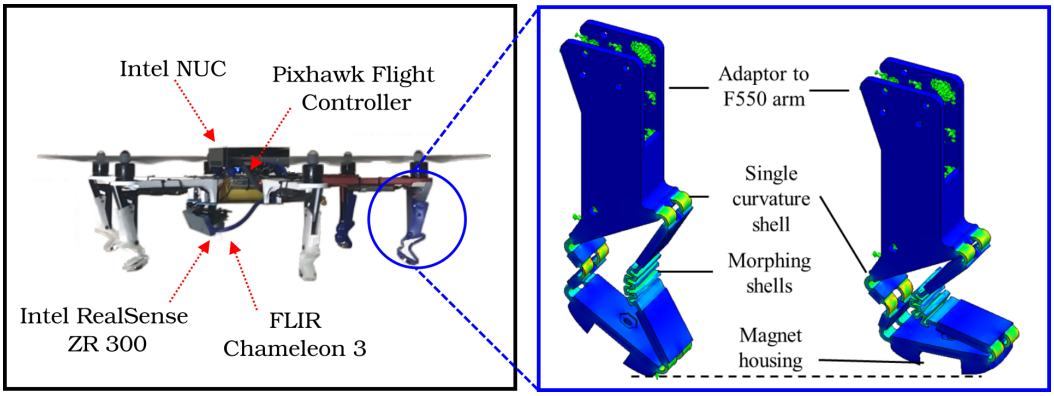


Figure 3.6: Left: MAV hardware explained with most important components. Right: The soft landing pad in its original configuration and the partially folded configuration where the magnet attaches to the horizontal surface.

position. We use ROS[3] as a middleware for the exchange of data between the individual software components. An overview of the software components is illustrated in Figure 3.7.

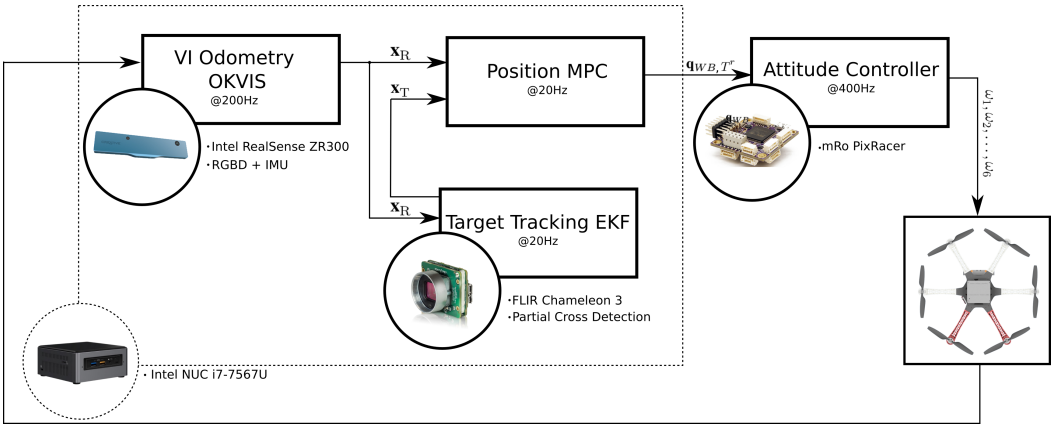


Figure 3.7: OKVIS [Leutenegger et al., 2014] is used in combination with the Intel RealSense ZR300 for the visual-inertial estimation of the MAV state $\mathbf{x}_R$. A monochrome FLIR Chameleon 3 is used for the detection of the moving target. The target state $\mathbf{x}_T$ and the MAV state $\mathbf{x}_R$ are further used in the MPC which generates a reference quaternion $\mathbf{q}^R_{WB}$ and reference collective thrust T^r for the attitude controller. The attitude controller is implemented on an mRo PixRacer flight controller which outputs the corresponding $\omega_1, \cdots, \omega_6$ motor commands.

[3] See http://www.ros.org/. Accessed April 2020.

3.6.2 Estimation of MAV and target state

As a basis for MAV state estimation we use OKVIS described in [Leutenegger et al., 2014]. We additionally adopt the modifications described in Section 2.4.2 in order to cope with RGBD camera in our sensor setup.

We track the landing target by means of an EKF, where we assume the MAV pose is given by the estimator described above, and poses are accurate enough. The dynamics employed for the prediction step consists of a constant velocity model for linear translation, and a constant orientation model, since rotation speeds of the target remain small. Inclusion of linear velocity of the target into its estimated state, however, is crucial, since we need it for accurate tracking.

In maths, we estimate the following target state:

$$\mathbf{x}_{\mathrm{T}} := \left[{}_W\mathbf{r}_T{}^T, \mathbf{q}_{WT}{}^T, {}_W\mathbf{v}_{WT}{}^T \right]^T \in \mathbb{R}^3 \times S^3 \times \mathbb{R}^3, \tag{3.20}$$

and we consider the following prediction model:

$$ {}_W\dot{\mathbf{r}}_T = {}_W\mathbf{v}_{WT}\,, \tag{3.21}$$

$$ \dot{\mathbf{q}}_{WT} = \frac{1}{2} \begin{bmatrix} \mathbf{w}_{\mathrm{rot}} \\ 0 \end{bmatrix} \otimes \mathbf{q}_{WT}\,, \tag{3.22}$$

$$ {}_W\dot{\mathbf{v}}_{WT} = \mathbf{w}_{\mathrm{vel}}, \tag{3.23}$$

where $\mathbf{w}_{\mathrm{rot}}$ and $\mathbf{w}_{\mathrm{vel}}$ denote 3-dimensional uncorrelated Gaussian white noise processes affecting orientation, and velocity, respectively. In our experiments, we set the noise parameters of said processes to $\sigma_{\mathrm{rot}} = [0.02, 0.02, 0.2]^T \mathrm{rad}/\sqrt{\mathrm{hz}}$ and $\sigma_{\mathrm{vel}} = [0.1, 0.1, 0.1]^T \mathrm{m}/(\mathrm{s}\sqrt{\mathrm{hz}})$, respectively.

For the update step, we employ observations of pre-defined landing pattern keypoints. Please refer to Figure 3.8 for an illustration of the specific keypoint locations on the target pattern at hand.

As the measurement function, we therefore use the predicted keypoint location of the t^{th} keypoint into the (undistorted) downward-looking fisheye camera:

$$\mathbf{h}_t(\mathbf{x}_{\mathrm{T}}) = \mathbf{u}_{\mathrm{D}}(\boldsymbol{T}_{WD}^{-1}\, \boldsymbol{T}_{WT}\, {}_T\boldsymbol{r}_t), \tag{3.24}$$

where $\mathbf{u}_{\mathrm{D}}$ denotes the (undistorted) projection model of the downward looking camera, the transformation $\boldsymbol{T}_{WD}$ is obtained through visual-inertial MAV state estimation, and the location of the keyoint on the target, ${}_T\boldsymbol{r}_t$, is a known constant.

Note that in our implementation, we use a tangent space representation of the orientation, $\delta\boldsymbol{\alpha}_{WT}$, around the current estimate $\bar{\mathbf{q}}_{WT}$ analogous to [Leutenegger et al., 2014], as

$$\mathbf{q}_{WT} = \exp(\delta\boldsymbol{\alpha}_{WT}) \otimes \bar{\mathbf{q}}_{WT} = \begin{bmatrix} \operatorname{sinc}\left\| \frac{\delta\boldsymbol{\alpha}_{WT}}{2} \right\| \frac{\delta\boldsymbol{\alpha}_{WT}}{2} \\ \cos\left\| \frac{\delta\boldsymbol{\alpha}_{WT}}{2} \right\| \end{bmatrix} \otimes \bar{\mathbf{q}}_{WT}\,, \tag{3.25}$$

with $\exp(\cdot)$ the exponential map.

In order to obtain the keypoint measurements to formulate the EKF update residual, we employ tracking in image space: we use 20 by 20 pixel patches around the keypoints warped from the pattern template image using the predicted pose relative to the camera observing them. Then, using keypoint location predictions from (3.24), we brute-force search their neighbourhood in the binarised image (60 by 60 pixel) by means of finding the square difference minimum w.r.t. the warped template patches. In our implementation, we used a keypoint measurement standard deviation of 5 pixels – partly accounting for the lack of proper hardware synchronisation between the RealSense (used for OKVIS) and the downward looking fisheye camera. For outlier removal, we discard keypoint measurements where this difference is two high, and also those whose residuals don't pass a Chi-square test (threshold: 9).

We illustrate these steps in Figure 3.8.

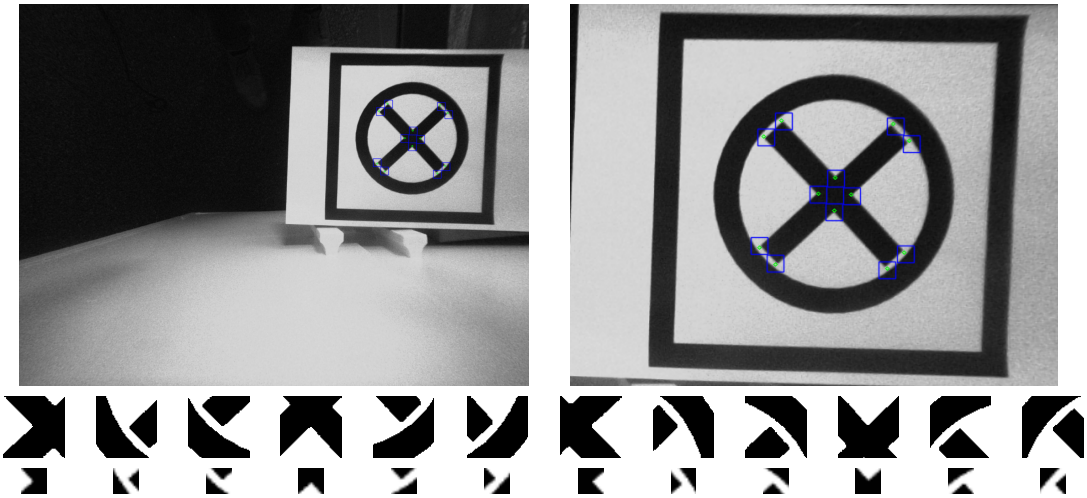


Figure 3.8: Top row: projected predicted keypoint neighbourhoods (search area) as blue boxes, and actual found detections as green circles (right: zoomed in). Middle row: binarised search neighbourhoods. Bottom row: respective warped templates to match.

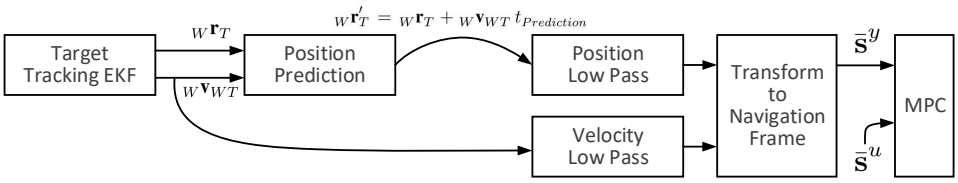


Figure 3.9: The reference generation scheme. The incorporation of the reference velocities and not only the position, improves the overall tracking performance of the system. The low pass filters for the position and velocity are used in order to filter out noisy target detections. The target position prediction by the $t_{\text{Prediction}}$ time offset, is performed to counteract unmodeled delays, from target detection to actual maneuver execution, that exist in the system.

3.6.3 High level mission profile

Reference generation

Although our controller can natively handle a time varying (over the prediction horizon) input $\bar{\mathbf{s}}^u = [\mathbf{s}_0^u, \mathbf{s}_1^u, \cdots, \mathbf{s}_{N-1}^u]^T$ and output $\bar{\mathbf{s}}^y = [\mathbf{s}_1^y, \mathbf{s}_2^y, \cdots, \mathbf{s}_N^y]^T$, for simplification, we do not explicitly generate time varying references but static ones which remain constant over the prediction horizon.

The approach of position setpoint instead of trajectory control is sufficient for waypoint navigation but insufficient when tracking of a moving target is required. Generating a sequence of reference position setpoints based on the observed position of the moving target will result in a constant offset, similar to the case of PD control, between the position of the MAV and the target. This is the result of the zero velocity requirement at each position setpoint which does not hold in the case of a moving target. We tackle this problem by not only penalizing deviation from a reference position but also from a reference velocity which in this case corresponds to the estimated velocity of the target. It is also worth mentioning that in our implementation, the prediction horizon is relatively short 0.1 s. Similarly, we can penalize deviation from a reference acceleration by incorporating it in $\bar{\mathbf{s}}^u$.

The scheme for the generation of the commanded setpoint (which includes both reference position and velocity) is illustrated in Figure 3.9. For simpler tasks such as waypoint navigation we follow the standard approach where position commands with zero reference velocities are sent to the controller.

We use low pass filters for the reference position and velocity with cutoff frequen-

cies $f_{\text{CutOffPosition}}$, $f_{\text{CutOffVelocity}}$ to filter out the noisy target position and velocity estimates. The target position prediction by the time offset $t_{\text{Prediction}}$, assuming constant velocity, is used in order to counteract the delay between a successful target detection and the actual maneuver of the MAV. The use of such a time offset was also motivated to compensate significant unmodeled delays that exist in the system such as the one related to the downward looking camera vision processing which arises from the lack of its hardware synchronization. Notice that the prediction of the target position also depends on its estimated velocity, while the time offset $t_{\text{Prediction}}$ was a constant parameter determined by tuning of the whole system before conducting the series of reported experiments. Numeric values of the parameters used are given in Table 3.2.

State machine

The overall behavior of the MAV is controlled by a state machine with the following modes which in an ideal experiment, are triggered sequentially.

Idle Mode: This is the default mode when the mission is initially triggered. The motors are disarmed and a zero thrust and orientation command $T^r = 0$, $\mathbf{q}^r_{WB} = [1, 0, 0, 0]^T$ is sent to the MAV.

Taking Off Mode: This mode is triggered once the first valid state estimation message becomes available. An arming command is sent to the motors and the MAV takes off to the predefined altitude $z_{\text{TakeOffHeight}}$ with a predefined ascending velocity $\dot{z}_{\text{TakeOffVelocity}}$.

Waypoint Mode: Immediately after take off, the MAV flies towards a predefined waypoint which coincides with the cross point of the figure-eight trajectory that the ground vehicle was following in the MBZIRC challenge.

Follow Mode: This mode is triggered when the landing pattern is successfully detected for the first time. A position and velocity command based on the procedure described in 3.6.3 is sent to the controller. When the difference between the current time $t_{\text{CurrentTime}}$ and the time of the last target detection $t_{\text{LastDetection}}$ is greater than the timeout parameter $t_{\text{DetectionTimeOut}}$, we assume that the target is not visible anymore and return back to the waypoint mode. The $t_{\text{DetectionTimeOut}}$ has to be set high enough to allow the MAV to move close

to where the target was detected but low enough to enable fast recovery to the waypoint in the case of an actual detection loss.

Landing Phase 1: Once the MAV has properly caught up the moving target, it starts descending towards it with a predefined velocity $\dot{z}_{\text{DescendingP1}}$. This mode is triggered when $\sqrt{(x_B - x_T)^2 + (y_B - y_T)^2} \leq d_{\text{PositionThreshold}}$ and $\sqrt{(\dot{x}_B - \dot{x}_T)^2 + (\dot{y}_B - \dot{y}_T)^2} \leq d_{\text{VelocityThreshold}}$ where x_B, x_T, y_B, y_T stand for the x, y position and velocity, expressed in the $\underrightarrow{\mathcal{F}}_W$, of the MAV and the target respectively. Similarly to the follow mode, if $t_{\text{CurrentTime}} - t_{\text{LastDetection}} \geq t_{\text{DetectionTimeOut}}$ we consider that the target is not visible anymore and return back to the waypoint mode.

Landing Phase 2: This is similar to the Landing Phase 1 mode apart from the facts that the descending velocity is now $\dot{z}_{\text{DescendingP2}}$ and that the target detection timeout check is ignored. This means that once the MAV enters this mode when $z_B - z_T \leq z_{\text{LandingP2Offset}}$ – where z_B, z_T are the z position expressed in the $\underrightarrow{\mathcal{F}}_W$ of the MAV and the target – a landing that cannot be canceled will be attempted. The above feature is necessary since reliable target tracking cannot be guaranteed when the MAV is really close to the target (due to partial visibility, blur, etc.).

Landed: This is the last operation mode which is triggered when $z_B - z_T \leq z_{\text{SuccessfulLandingOffset}}$. After successful landing a disarming command is sent to the motors and the mission has ended.

Figure 3.10 shows a flowchart with the described operating modes including the strategies in the case of target detection loss.

3.6.4 Field experiments

In order to validate the performance of our platform and the developed algorithms, we performed a series of outdoor experiments replicating a similar to the MBZIRC Challenge scenario. Note that we performed these experiments *after* our participation in MBZIRC, where the preliminary version of the described system proved too unreliable, specifically in the extreme wind conditions, and triggered us to improve several aspects of hardware, algorithms, and software.

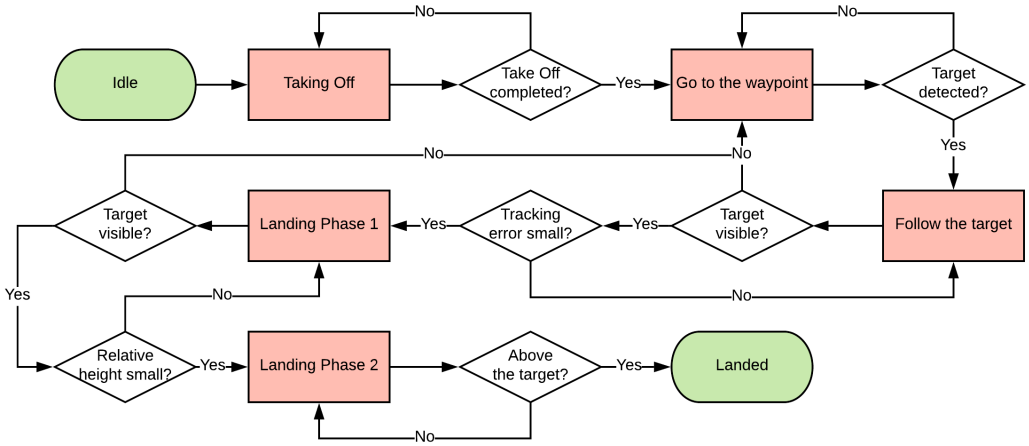


Figure 3.10: Flowchart indicating the transition between the various operating modes.

The experimental flow is as follows: The MAV takes off from the starting point location where it is initially placed on a 45cm height box. This is necessary in order to make sure that OKVIS can detect and track salient points on the RGB image. Once take off is completed, the MAV navigates to the predefined waypoint where it hovers till the target gets detected. After successful detection, the MAV tries to follow the target and land on it. The transition between the different operating modes is the one described in Section 3.6.3. The MAV executes the whole task autonomously and does not use any user provided information about the target position and velocity but it exclusively relies on its onboard sensors and algorithms. A human pilot triggers the start of the mission and can also manually intervene in order to prevent a crash in case of an algorithmic failure. We annotate every experiment as *Successful* or *Failed* based on whether the MAV managed to land successfully on the target according to the MBZIRC Challenge specifications [4]. Specifically, a successful landing is when the MAV comes to a rest on the landing target, with the MAV intact.

The landing target is identical to the one used in the MBZIRC Challenge and was being pulled manually, in a random way, with a rope. We tested moving the target with velocities between 1.3 m/s and 5.7 m/s which exceed the maximum target velocity of 4.1 m/s during the MBZIRC Challenge. Table 3.1 contains the results of the conducted experiments while Table 3.2 contains a list for all the

[4] See https://www.mbzirc.com/faqs/2017. Accessed April 2020.

parameters used in the developed algorithms. Alongside the outcome (*Successful*/*Failed*) for each experiment, Table 3.1 contains the norm of the position error $e_{\text{xy}} = \sqrt{(x_B - x_T)^2 + (y_B - y_T)^2}$ between the MAV and target position when the MAV has landed and the time $t_{\text{TrackingDuration}}$ which corresponds to the time that was needed for a successful landing from the time instant where the target was initially detected.

Since no motion capture system or RTK GPS was used, we cannot provide ground truth for the MAV and target position. All the data presented in this section corresponds to the estimates from OKVIS and the target tracking EKF.

Table 3.1: Successful/unsuccessful landings

Experiment #	1	2	3	4	5	6	7	8	9	10	11	12
Average Velocity $\bar{v}$ [m s^{-1}]	1.3	1.7	2.0	2.0	2.3	2.4	3.5	3.8	4.0	4.6	5.0	5 .7
error e_{xy} [cm]	2.5	11.8	5.2	30.6	16.5	6.6	30.2	18.6	15.6	29.8	45.8	7.6
Duration [s]	5.1	4.8	5.6	7.7	7.1	5.7	5.0	5.9	7.5	6.1	5.6	5.0
Outcome*	*S*	*S*	*S*	*S*	*S*	*S*	*S*	*S*	*S*	*F*	*S*	*F*

S = *Successful*, *F* = *Failed*.

Table 3.2: Landing software parameters.

$f_{\text{CutOffPosition}}$	30 hz	$z_{\text{TakeOffHeight}}$	5.2 m	$\dot{z}_{\text{TakeOffVelocity}}$	0.5 m/s
$f_{\text{CutOffVelocity}}$	30 hz	$d_{\text{PositionThreshold}}$	0.8 m	$d_{\text{VelocityThreshold}}$	0.8 m/s
$t_{\text{Prediction}}$	0.1 s	$z_{\text{LandingP2Offset}}$	2.5 m	$\dot{z}_{\text{DescendingP1}}$	1.3 m/s
$t_{\text{DetectionTimeOut}}$	0.5 s	$z_{\text{SuccessfulLandingOffset}}$	0.35 m	$\dot{z}_{\text{DescendingP2}}$	1.8 m/s

Selected experiments

We present the MAV and target position and attitude data for three selected experiments. Specifically, two of them (Figures figs. 3.11 to 3.14) were successful (one with low target velocity and another with high), whereas the third one (Figures 3.15, 3.16) was unsuccessful. The figures for the remaining nine experiments can be found in [Tzoumanikas et al., 2019]. In all the figures presented, solid lines refer to the estimated values given by OKVIS while the dashed ones refer to their corresponding reference. Reference position x^r, y^r, z^r was generated as described in Section 3.6.3 while θ^r, ϕ^r and T^r were obtained by solving the MPC optimization problem. ψ^r was always kept to zero. Also, notice that there is no solid line for T^r (plotted in Newtons) since – as explained in Section 3.4 – the dynamics of the motors were ignored and it was assumed that the applied thrust coincides with the reference.

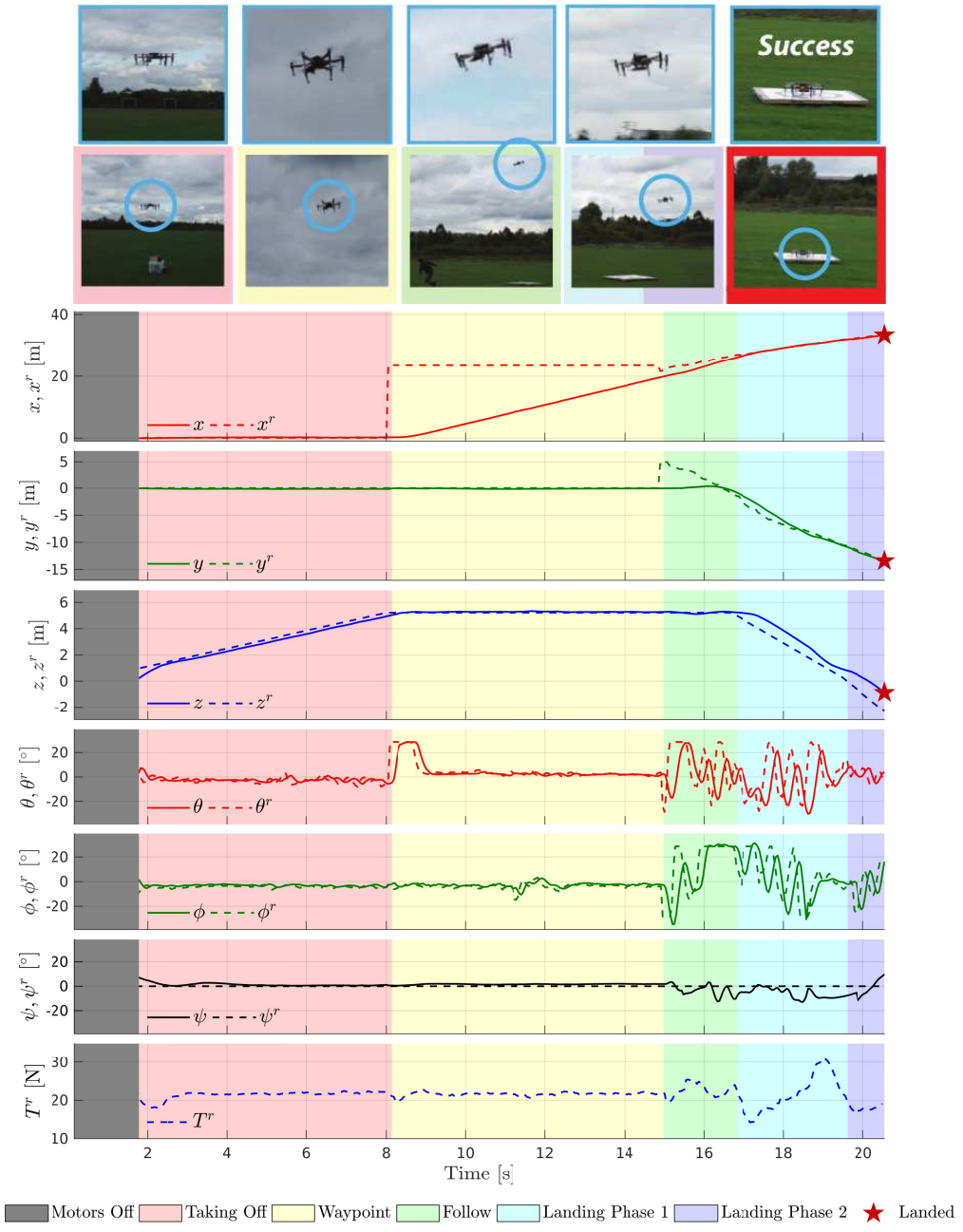


Figure 3.11: Position and attitude data for experiment #11. A *successful* landing on a target moving with 5.0 m/s was achieved. The dashed lines on the upper 3 plots correspond to the reference position which is generated based on the estimated position and velocity of the target while the dashed lines on the lower 3 plots correspond to the attitude angles and thrust generated by the MPC. The MAV lands 5.59 s after the first target detection and 45.8 cm away from the target center.

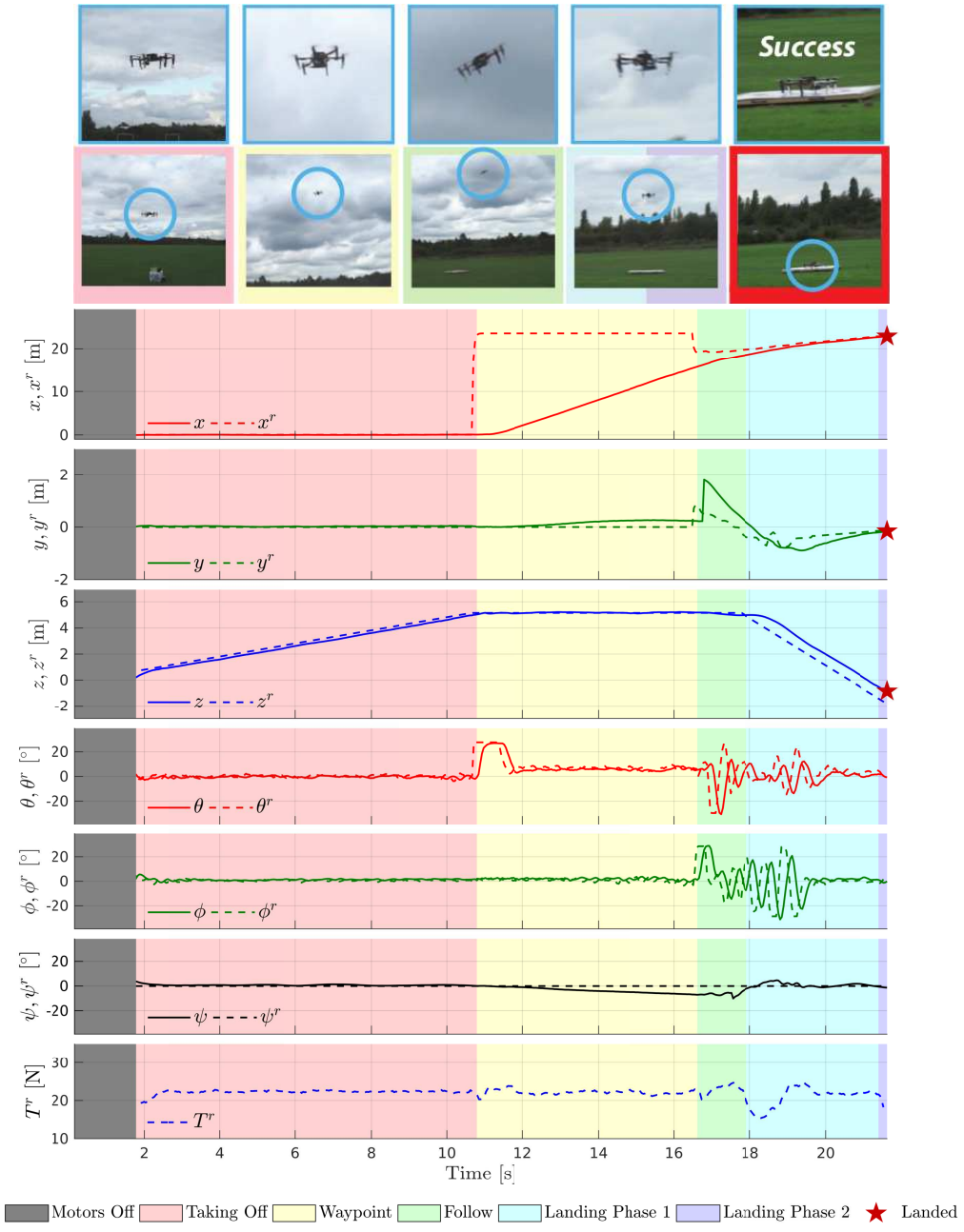


Figure 3.12: Position and attitude data for experiment #1. A *successful* landing was achieved on a target moving with 1.3 m/s. The MAV lands 5.06 s after the first target detection and 2.5 cm away from the target center.

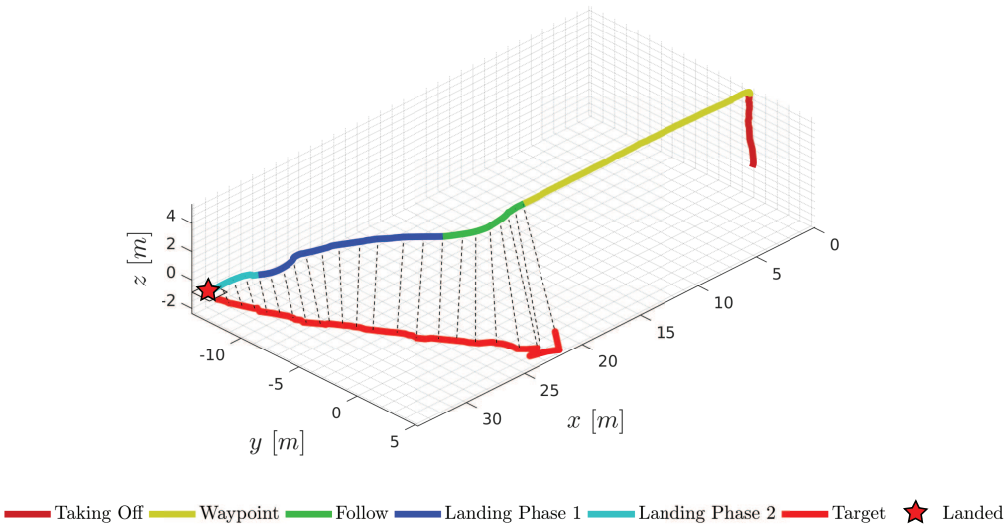


Figure 3.13: The MAV and target position visualised as a 3D plot for the experiment #11. Dashed lines indicate the corresponding target detection/tracking.

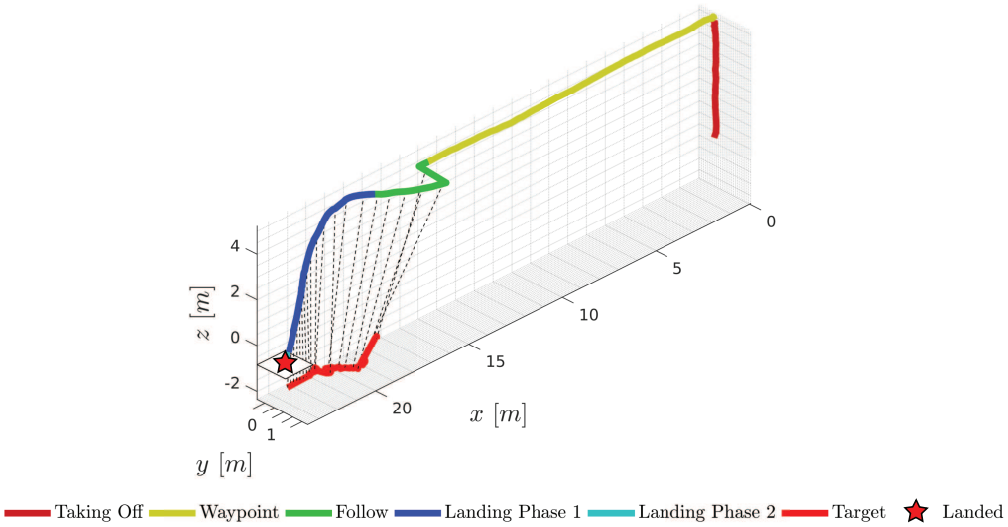


Figure 3.14: The MAV and target position in 3D for the experiment #1.

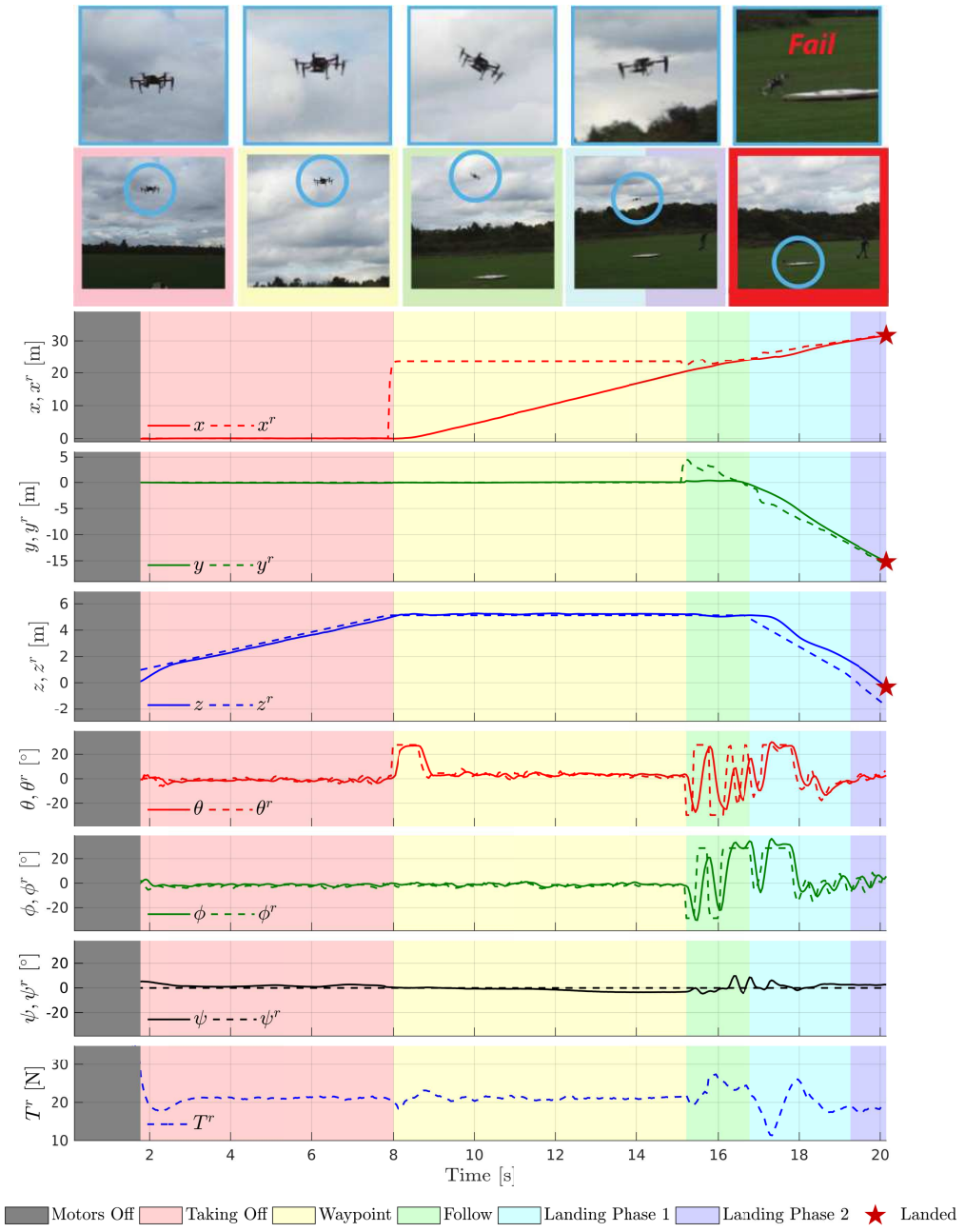


Figure 3.15: Position and attitude data for experiment #12. A *failed* attempt of landing on a target moving with 5.7 m/s was performed. Although that the final position error e_{xy} between the target and the MAV is only 7.6 cm, the real MAV position differs from the target position (See also Figure 3.17). Unreliable estimation of the target position especially when flying close to it led to a landing on a predicted target position different from the real one.

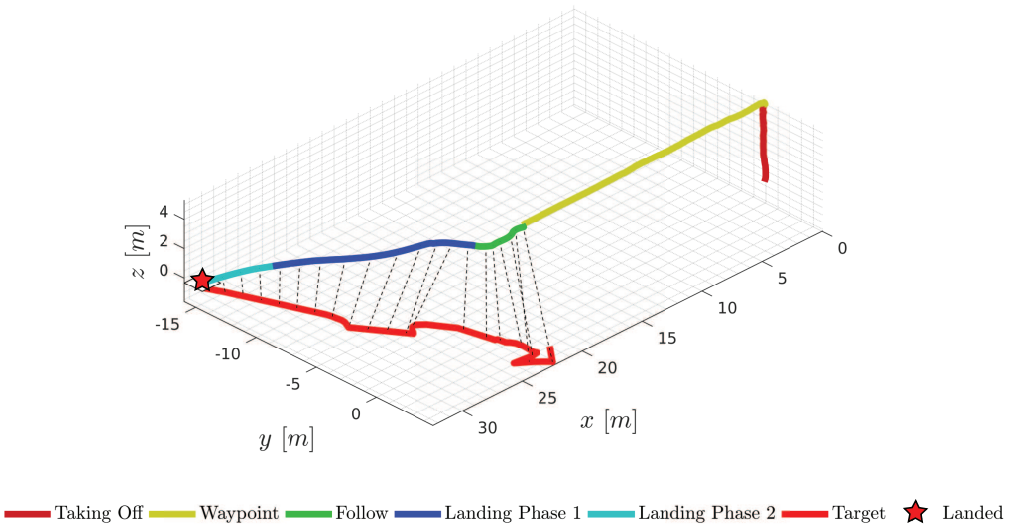


Figure 3.16: The MAV and target position in 3D for the experiment #12.

3.6.5 Comparison with other MBZIRC teams

From the experimental results, we realize that our hardware and software framework achieves the original goal that it was designed for with high repeatability. The majority of the experiments (10/12) were successful whereas there were only 2 failed attempts, both in the high velocity region 4.6 m/s and 5.7 m/s. The worst case position error e_{xy} between the target and the MAV after landing was 45.8 cm which corresponds to the fastest successful attempt #11. The worst case tracking accuracy achieved by the controller was more than sufficient especially considering the size of the MAV and the landing target and the fact that the experiments were conducted in an outdoor environment. The position error is generally larger in the high velocity experiments (No 7, 8, 9, 11) and lower in the slower ones (No 1, 2, 3, 4, 5, 6). We consider the noisy and uncontrolled outdoor environment (presence of wind and varying lighting conditions, which affect the target state estimation) to be the main reason.

Concerning MAV state estimation, with appropriate settings of gain and brightness on the Realsense RGB-D camera, we have not experienced any problems related to lack of tracked keypoints, therefore leaving the visual-inertial estimation system working flawlessly.

Regarding the attitude commands generated by the MPC, these are generally not smooth in the high velocity Experiments but smoother for the low velocity ones.

This was the result of over-penalizing the position/velocity error compared to the penalization of the smoothness terms. In other words, the gains of the optimisation problem are chosen in a way such that position/velocity tracking is prioritized over smooth attitude changes. Regarding the MAV response to the attitude commands, there exist a phase and amplitude difference between the commanded attitude angles and the real ones. Since the linear model also includes an estimated model for the closed loop attitude dynamics (Equation (3.4)), the difference between a command and the actual response is taken into account by the MPC. We also observe that although the ψ^r was always kept to zero, ψ was minimally affected when aggressive θ^r, ϕ^r were required. This is the result of existence of coupling between the MAV degrees of freedom; something not captured by the linear model used for control.

It is important to highlight that our controller does not contain any integral error term and generally it is not aware of any not modeled disturbance acting on the system. Hence, it is unable to counteract any constant position/velocity error offset which is either the result of an external disturbance or imperfect system calibration. The controller can only compensate non-persistent external disturbances due to the feedback on the system. Rejection of constant disturbances would only be possible, if these could be estimated and taken into account from the control model. In order to make sure that the MAV will remain in a state envelope where the used linear model remains valid, we use appropriate constraints in the optimization problem. These also guarantee that the MAV operates far from the gimbal lock (since Euler angles are used for the control formulation).

In all the successful experiments, the MAV managed to land within less than 7.65 s after the first target detection.The final time in all the experiments did not exceed 30 s, and this is mainly dominated by the time required for take off and moving to the waypoint. We cannot make a direct comparison with the performance of the teams who participated in the real challenge, as the final time also depends on when the target becomes visible for the first time. We can, however, compare our approach with the ones by team ETH-MAV [Bähnemann et al., 2017] and NimbRo [Beul et al., 2017] which both successfully landed on the moving target during the MBZIRC competition within 56 s and 32 s, respectively.

- ETH-MAV team used two cascaded EKFs that combine data from an RTK GPS and a Visual Inertial sensor for the MAV state estimation. A downward

looking camera is used for the target detection. Target tracking and landing is achieved through a non-linear MPC and a path planning algorithm which also relies on the prior knowledge of the MBZIRC track shape. They also use a Lidar sensor in order to check the distance between the MAV and the target and trigger the final stage of the landing procedure.

- NimbRo team used a GPS + IMU fusion algorithm for the MAV state estimation, along with a fast trajectory planning algorithm. In order to detect and track the target they use two different cameras. One is looking downwards, while the other forward-downward. The usage of such a camera configuration, not only robustifies the target detection but also enables faster initial detection of the moving target. Their MAV was programmed to wait at the waypoint while rotating around its vertical axis at the same time. Mechanical switches are placed on the legs of the MAV in order to detect contact and thus successful landing on the moving target.

Both teams followed a mission profile similar to ours with *Wait at waypoint – Follow – Final approach and landing* to be the main modes of operation, while the main difference was the active rotating/searching performed by the NimbRo team MAV which can greatly reduce the time needed for initial target detection. Our approach does not rely on RTK-GPS making it appropriate for similar tasks in GPS denied environments, and moreover removes the necessity for a ground station (which wirelessly sends the GPS corrections to the MAV). We consider our approach to be simple and general enough to be implemented in similar scenarios. Regarding the software developed, we consider the lack of dynamically feasible trajectories planning and the usage of a linear MPC to be the main limitation of our approach compared to the above. Regarding the hardware, we used – in our opinion – the absolute minimum number of sensors/components required for autonomous navigation. Since we employ only the necessary hardware components, we don't get the benefits of more reliable target detection (through use of multiple cameras) and prevention of unsuccessful landing attempts (through use of extra sensors such as Lidar).

As far as the failed attempts (experiment 10 and 12) are concerned, both of them seem successful when we look at the position plots and the final position error after landing. As shown in Figure 3.17 in both Experiment 10 and 12 the MAV landed really close to the true target position. We argue that the reason behind the failed

attempts is the quality of the estimated target state especially when the MAV flies really close to the target. Specifically, in Experiment 12 (Figure 3.15) the target was pulled sharply in a different direction while the MAV was in Landing Phase 2. Due to the proximity between the MAV and target, the latter was partially or completely invisible due to the horizontally mounted downward looking camera and MAV tilt. Given the average target velocity in these experiments (4.6 m/s and 5.7 m/s) and the fact that the frame rate of the downward looking camera was 20 hz, the target position between two consecutive camera frames was changing by 23 cm and 28.5 cm respectively. It becomes clear that in this frame rate and high target velocity, even one frame of lost tracking can result a failed landing attempt.

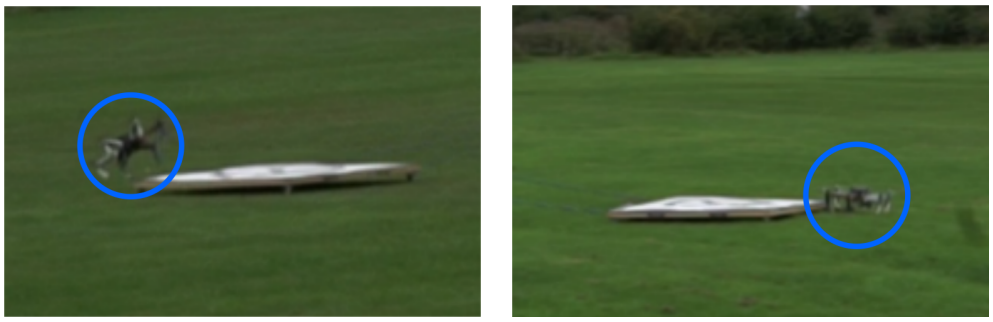


Figure 3.17: The two *failed* landing attempts. Experiment #12 on the left and experiment #10 on the right.

We consider that our approach has two main limitations. The first one is unreliable target tracking when the MAV flies close above the target. This could by either fixed by improving the target tracking when it is partially visible or by mechanically rotating the downward looking camera such that the target always remains in the field of view. The second limitation is the criterion used for triggering the landing attempt when flying above the MAV. This depends only on the relative height between the MAV and the target. It would be better to use a criterion which takes into account the confidence/uncertainty of the estimated target states or use additional hardware (e.g. Lidar sensor) for more accurate relative height estimation. This would hopefully lead to higher probability for a successful landing. Nevertheless, we believe to have achieved a respectable success rate given the minimal hardware/sensor complexity.

3.7 Experimental results: Application to AABM

In this Section we present experimental results related to the AABM[5] project which aims at developing the hardware and software which will enable aerial robots to autonomously 3D print small structures. Our role in the project was to provide the required state estimation and control algorithms which will enable precise trajectory tracking.

Briefly, the experimental setup consisted of two MAVs with a 4.5 kg payload capacity which can deposit building material, while flying, through an appropriate extrusion mechanism. Trajectories containing multiple printing layers were commanded to the two MAVs which were serially executing the building task. In order to enable precision and mainly cope with the limited material extrusion speed, the maximum velocity of the commanded trajectories was limited to 5 cm/s.

An example of a 10-layer reference printing trajectory and the actual MAV response in a virtual printing scenario (without extruding building material) is given in Figure 3.18.

3.7.1 Linear MPC modifications

In order to cope with the increased need of precision for the AABM and handle the slowly varying MAV mass due to building material extrusion, we have done the following modifications to the linear MPC presented in Section 3.5:

1. We reduced the order of the closed loop attitude dynamics given in (3.4) from a second order model to a first one given by:

$$\dot{\theta} = -b_{\dot{\theta}\theta}\theta + b_{\theta^r}\theta^r, \tag{3.26a}$$

$$\dot{\phi} = -b_{\dot{\phi}\phi}\phi + b_{\phi^r}\phi^r. \tag{3.26b}$$

   This approximation also adopted in similar works such as [Kamel et al., 2017a] results in dimension reduction of the control state $\mathbf{x} := [x, \dot{x}, \theta, y, \dot{y}, \phi, z, \dot{z}]^\top$, while the control input remains unchanged $\mathbf{u} := [\theta^r, \phi^r, T^r]^\top$.

2. We increased the length of the prediction horizon to 1.0 s while keeping the same discretisation timestep $dt = 50$ ms.

[5] See http://www.aerial-abm.com/. Accessed April 2020.

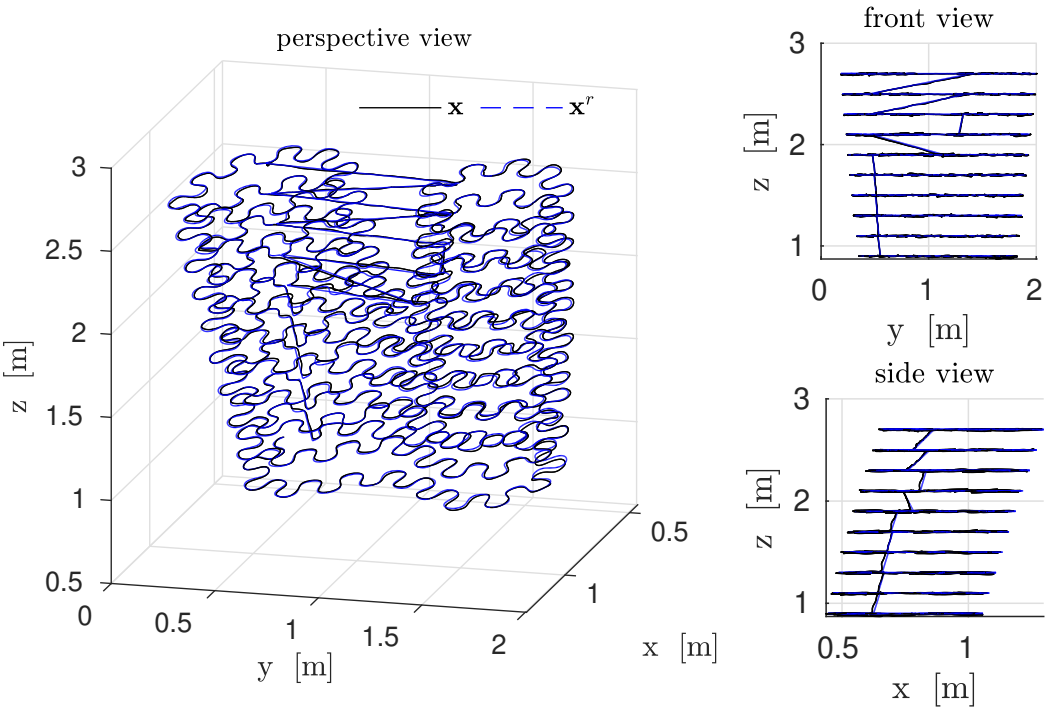


Figure 3.18: Three different views of a 10-layer virtual 3D printing experiment. In order to enable precise tracking of the wavy reference trajectories and cope with the limited material extrusion speed, the maximum velocity of the commanded trajectories was limited to 5 cm/s.

3. We augmented the reference orientation and thrust generated by the MPC $\mathbf{u}$ with integral terms $\mathbf{u}_{\mathrm{I}} := [\theta_{\mathrm{I}}, \phi_{\mathrm{I}}, T_{\mathrm{I}}]^{\top}$. This was done in order to counteract small orientation offsets (originating from imperfect MAV hover-orientation calibration) and the MAV changing mass due to the deposition of print material. The integral term $\mathbf{u}_{\mathrm{I}}$ is computed as follows:

$$\mathbf{u}_{\mathrm{I}}(t) = \mathbf{Q}_I \int_0^t ({}_N\mathbf{r}_B - {}_N\mathbf{r}_B^r) d\tau, \tag{3.27}$$

with $\mathbf{Q}_I$ an experimentally tuned gain. As common practice in control, we prevent the integral term from accumulating above pre-determined bounds. Specifically we constraint the orientation components to be less than 1.5° and the thrust component to be less than 1.0 kg (which coincides with maximum possible change in the MAV mass).

4. Since precision is the main objective for the printing experiments, we provide the controller with time varying full state $\bar{\mathbf{s}}^y$ and input $\bar{\mathbf{s}}^u$ trajectory commands (starting from the current time instant till the end of the receding horizon).

This way the controller can react in advance for upcoming references which is particularly useful for e.g. maneuvers containing sharp turns.

3.7.2 Selected printing experiments

In Figures 3.19-3.20 we show the tracking performance achieved by our controller in an indoors-performed printing experiment with MAV pose data being provided by a motion capture system. Specifically, Figure 3.19 (left) shows the complete multi-layer reference trajectory (subsampled for visibility purposes), while the tracking performance for a single layer is shown in Figure 3.19 (right). Despite the saw-tooth like x, y references, the controller enables centimeter-level tracking as at the visualised scale the reference and actual MAV position are indistinguishable.

The boxplot in Figure 3.20, which contains the tracking statistics of the first ten layers, reveals that the per-axis error does not exceed 1.5 cm.

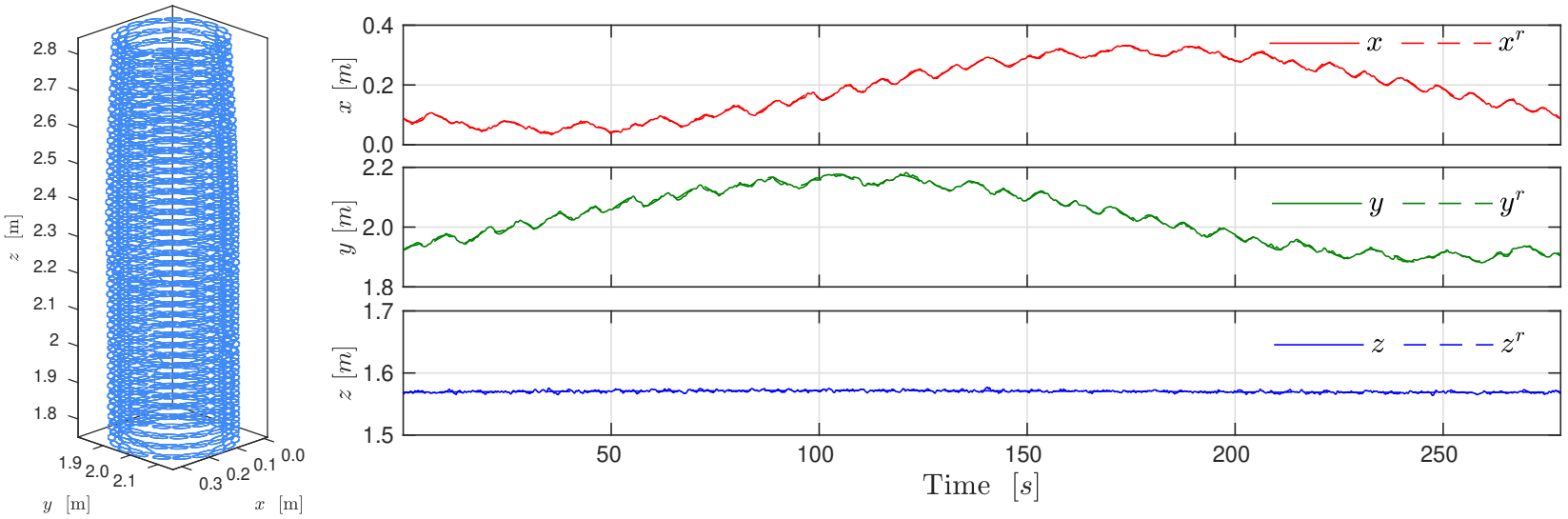


Figure 3.19: Reference and actual MAV trajectory during an AABM printing experiment. Left: The multi-layer 3D printing reference position. Right: Per-axis reference and actual MAV position for a selected layer.

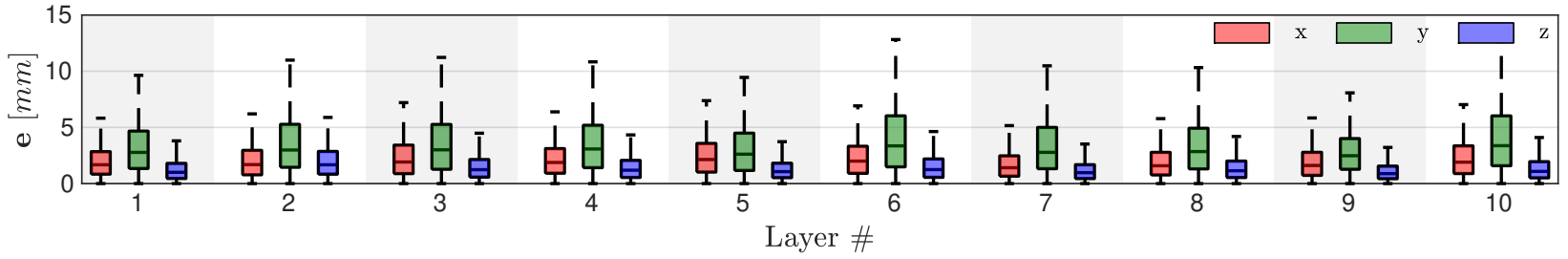


Figure 3.20: MAV position error statistics for the first ten layers of the AABM printing experiment. The per-axis error for the tested trajectories, does not exceed 1.5 cm.

3.8 Discussion

In this Chapter we presented a linear MPC with soft state constraints formulated as a QP that can be solved with any generic solver. As it can be seen from the experimental results, it can be successfully used for dynamic tasks such as the tracking and landing scenario while its performance is maximised when slow manoeuvres are required. This is owing to the linear model being based on near hover operation assumptions.

Despite the limitations that arise from the use of an approximate linear model, as well as the use of Euler angles as a parameterisation for orientation, we would like to comment on the specific advantages which become apparent from the presented experiments as well as its use in multiple live demonstrations the past four years. First and foremost, its easy adaptation to other platforms as the linear model is platform agnostic. The same controller can be applied without modification to a quadcopter, hexacopter or any other underactuated platform as the only requirement is the existence of a closed loop attitude controller. Secondly, the position-attitude dynamics separation allows the position loop to be run at a low update rate which in our case was 20 Hz. This makes the system less sensitive to e.g. delays in the state estimation scheme since the fast attitude dynamics are separately controlled using the onboard IMU. Lastly, the resulting optimisation problem is computationally cheap to solve (consistently less than 1 ms per control iteration with our hardware) and is thus suitable for platforms with limited computing resources.

Regarding the possible extensions of the presented linear MPC, the generalisation to a NMPC is given in the upcoming Chapter. Future work, thus should be focused on the engineering related improvements. As an example we consider the use of the fastest QP solver [Frison and Diehl, 2020] currently available. Lastly, a useful extension would be the online estimation of external forces and moments as in [Kamel et al., 2017a] and their incorporation in the control model. This would allow for offset free navigation without the need of additional integral terms and would further enable the compensation of external disturbances before their integration into position/velocity error.

Chapter 4

Nonlinear Model Predictive Control

Contents

Parts of this Chapter appear in: Tzoumanikas, D., Yan, Q., and Leutenegger, S. (2020). Nonlinear MPC with Motor Failure Identification and Recovery for Safe and Aggressive Multicopter Flight. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).* [Tzoumanikas et al., 2020b]

4.1 Introduction

The linear MPC presented in Chapter 3 is characterised by its ease of use and its effortless adaptation to any type of multirotor without the absolute need of a powerful onboard computer. Its per-axis tracking accuracy heavily depends on the aggressiveness of the reference trajectories and varies from 1 cm for slow trajectories ($< 5\,\text{cm/s}$) to 50 cm for fast ones ($> 5\,\text{m/s}$) as tested in the landing experiment. This is mainly because the linear model is derived assuming a near hover operation which does not hold when the MAV has to exert motion with significant linear velocity and acceleration. Additionally, the linear model uses Euler angles for the representation of orientation and thus the orientation close to gimbal lock is prevented from additional orientation constraints. Constraining the maximum tilting angle results in limiting the maximum translational acceleration.

Another issue is the lack of robustness in the case of actuator failures such as a partially or completely damaged motor/propeller. This is because it relies on the existence of a stable attitude controller which on most occasions is based on simple PID implementations and thus unable to handle an actuator failure without appropriate modifications.

These problems can to some extend be eliminated when the true non linear MAV model is taken into account in the control design and a unified controller is responsible for the 6DoF motion of the MAV. This is a research topic that has become particularly popular during the past years thanks to the increasing computation capabilities and the open source availability of optimisation and control toolboxes such as [Houska et al., 2011, Giftthaler et al., 2018, Mattingley and Boyd, 2012].

At the same time, robust performance under mechanical failures (such as motor failures), can only be achieved when the failure can be correctly identified and appropriately handled by the control design. To this purpose, the existence of a model based supervisory algorithm that monitors the health of the system and accordingly notifies the controller in the event of a failure, is necessary.

In this Chapter we propose a NMPC aiming at overcoming the limitations of its linear counterpart. We further suggest an EKF based fault detection algorithm and a specific to our MAV type stabilisation technique in the event of a motor failure. An example of our MAV performing online fault detection and stable position and yaw control despite the loss of a motor is shown in Figure 4.1.

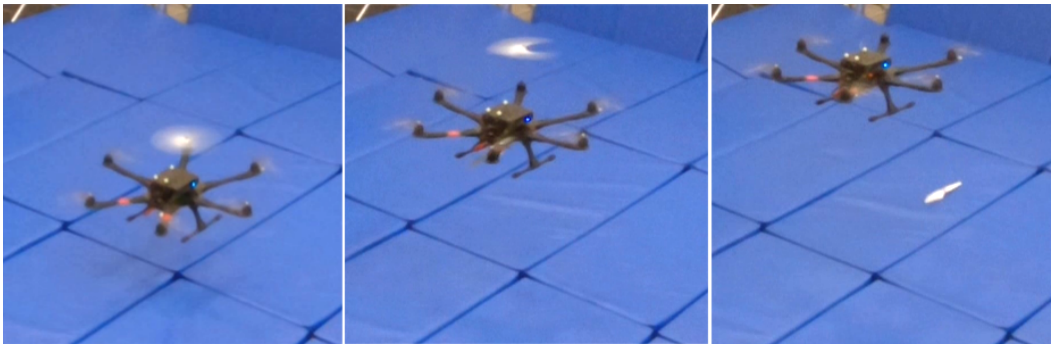


Figure 4.1: Our MAV experiencing a propeller loss. The fault is identified online, the failsafe enabled and the MAV can still control its position and orientation.

4.2 Related work

As mentioned in Chapter 3, the most common approach regarding MPC in MAVs is that of a cascade connection between a position and an attitude controller. In the simplest form, a linear model can be used for the translational dynamics, resulting in an optimisation problem whose solution can be solved online [Tzoumanikas et al., 2019, Sa et al., 2018a] or pre-computed in the form of lookup tables [Darivianakis et al., 2014, Papachristos et al., 2016]. The use of a non-linear model for the translational dynamics such as the one presented in [Kamel et al., 2017a] presents performance improvements especially when tracking of aggressive trajectories is required. The approach of the cascaded position-attitude controllers has become popular due to its ease of use, since most of the available platforms come with a pre-tuned attitude controller. However, it relies on the assumption that the attitude dynamics can be controlled independently, requiring bandwidth separation between the successive loops, i.e. slow control of attitude. The aforementioned works furthermore use Euler angles for the vehicle orientation which prohibits the operation close to gimbal lock. Analogously to the position-attitude approach the authors of [Falanga et al., 2018] and [Foehn and Scaramuzza, 2018] propose a quaternion based position controller which uses the angular rates as control inputs. These were assumed to be tracked perfectly by a separate angular rate controller.

The benefits of using the true non-linear model of the MAV has been successfully illustrated in [Kamel et al., 2015] where an attitude NMPC was employed to stabilise the position of a hexacopter with a motor failure while control of yaw was lost. Additionally, the authors of [Neunert et al., 2016] proposed an Sequential Linear Quadratic (SLQ) MPC algorithm able to run onboard an MAV and capable of

following full state trajectory commands. Similarly, in [de Crousaz et al., 2015] simulation results of an SLQ controller stabilizing a quadrotor with slung load and a quadrotor with motor failure were presented.

For the control allocation problem – that is, the mapping of the body moments and thrust to actuator commands – the most commonly used method employs the pseudo-inverse or weighted pseudo-inverse of the allocation matrix (e.g. [Achtelik et al., 2013, Lee et al., 2010]). The main advantage of this approach is its simplicity since the pseudo-inverse has to be computed once and the actuator commands can be obtained through a simple matrix by vector multiplication. However, the main drawback is the fact that it can produce actuator commands that are not feasible (e.g. greater thrust than what each motor can provide). Naively, this can be handled by clipping the actuator command such that it lies in the achievable range. Control allocation techniques that respect the actuator limits, such as the ones presented in [Faessler et al., 2017, Brescianini and D'Andrea, 2018], result in better trajectory tracking. This is partially due to the prioritisation of the roll/pitch moments and collective thrust over the yaw moment which does not directly contribute to position tracking.

Another alternative approach is the one presented in [Brescianini and D'Andrea, 2018b], where the minimum energy solution is obtained by solving an optimisation problem. The authors exploit the structure of the allocation matrix nullspace in order to transform the original optimisation problem into a computationally cheaper one. Similar to our method, theirs can be used on platforms equipped with bidirectional capable motors but requires the use of symmetrical propellers. When non symmetric propellers are used, the resulting allocation matrix is not constant but depends on the direction of rotation of each motor.

Regarding the fault tolerant control, the different approaches vary based on whether the hardware platform can natively handle a motor failure or not. Examples include the cases of octacopters [Saied et al., 2015, Saied et al., 2017] or, more relevant to our research, hexacopters with unconventional motor layouts such as [Schneider et al., 2012, Mazeh et al., 2018, Nguyen et al., 2019] where it was shown that stable position and yaw tracking can be achieved for failure occurring on specific motors. When full position and yaw control is not physically possible by the hardware platform (e.g. quadcopters, hexacopters with symmetric motor layout), software based approaches such as [Mueller and D'Andrea, 2014, de Crousaz et al., 2015, Kamel

et al., 2015] attempt to prioritise tracking of the controllable states (e.g. position) over the uncontrollable ones (e.g. yaw). As far as the fault detection scheme is concerned, a simple and accurate way of detecting failures is by using direct measurements of motor speed and/or electrical current [Saied et al., 2017]. The deviation between the speed measurements and the commanded ones can be used to identify a motor stall/malfunction while the current measurements for detecting a propeller loss. Alternative software-based methods [Saied et al., 2015, Nguyen et al., 2019] rely on the formulation of residuals between the measured and predicted –assuming no failure– system states such as orientation, angular rate or vertical velocity. The sign of the residuals in combination with some lookup table (that covers all possible sign combinations) are used for detecting motor failures. The underlying idea is that losing thrust from a specific motor results in a specific system response which can be directly compared to the nominal one e.g. a hovering MAV experiencing a sudden thrust loss from the front-left motor will loose altitude and tilt in the forward left direction.

4.3 Contribution

With respect to the related works described above, we present a series of algorithms which address the problem of aggressive, precise and fault tolerant MAV navigation. Our contributions are as follows:

- The design of a non-linear model predictive controller with body torques and collective thrust as the control inputs which jointly controls the MAV's 6D DoFs. A quaternion based orientation parameterisation is chosen thus avoiding problems related to gimbal lock while the unified control approach eliminates assumptions related to the position/attitude dynamics separation since the NMPC does not require the existence of a separate attitude or rate controller.

- The design of an optimisation based control allocation algorithm that maps the desired control inputs into feasible thrust commands for each motor. We consider the general case where the motors can generate both positive and negative thrust. Our method aims at combining the advantages of the various methods described above. Namely, the ability to: (i) prioritise certain control inputs over others, (ii) generate feasible actuator commands, (iii) cope

with bidirectional capable motors without the absolute need of symmetrical propellers.

- The design of a stabilisation technique, based on reversing the motor direction of rotation, which enables stable position and yaw control. This can be additionally applied to hexacopters with symmetric motor layouts ensuring that due to symmetry, failure in any motor can be handled equally well.

- The design of an EKF, exclusively relying on IMU measurements, which monitors the health of each individual actuator (motor/propeller) which we use for fast identification of an actuator failure. Compared to similar in concept approaches, our EKF directly estimates the per-motor ratio of the observed to commanded thrust (while also considering the motor dynamics), thus providing a natural way of signalling a motor/propeller failure.

4.4 System overview

The software pipeline presented in this Chapter consists of the following different blocks: (i) the NMPC which receives the state estimates and reference trajectory commands and produces body torques and collective thrust as the control inputs; (ii) the control allocation block which transforms the control inputs to feasible actuator commands; (iii) the failure detection algorithm which estimates the health status of each motor and notifies the control allocation block in the case of a failure. We use the state estimation algorithm outline in Section 2.4.1 which fuses data from the onboard IMU and the motion capture system. An overview of the system is given in Figure 4.2.

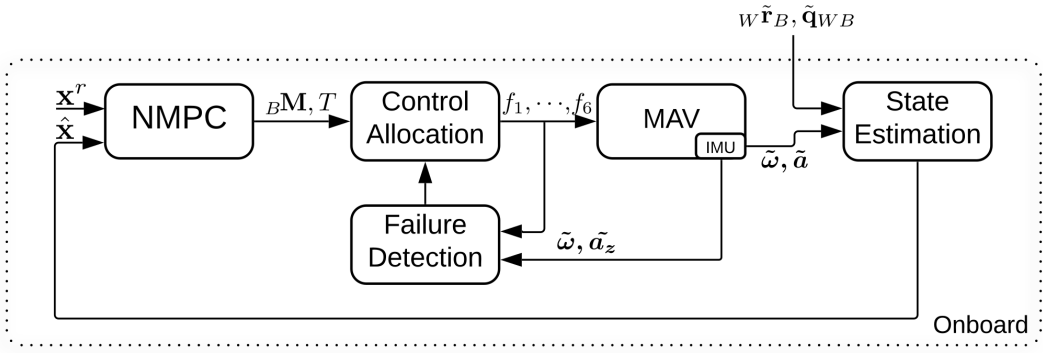


Figure 4.2: Overview of the various software components that run onboard the MAV. The control loop runs at 100Hz while the failure detection EKF at 400Hz.

4.5 Model based control

4.5.1 Nonlinear modelling

The Newton-Euler equations are used to model the MAV dynamics. We ignore less significant phenomena such as the effect of the aerodynamic friction and the gyroscopic moments due to the rotation of the propellers (but our model could be extended accordingly with ease). The MAV dynamics can then be written in the following form:

$$ {}_W\dot{\mathbf{r}}_B = {}_W\mathbf{v}_B \,, \tag{4.1a} $$

$$ \dot{\mathbf{q}}_{WB} = \frac{1}{2}\boldsymbol{\Omega}({}_B\boldsymbol{\omega})\mathbf{q}_{WB} \,, \tag{4.1b} $$

$$ {}_W\dot{\mathbf{v}}_B = \frac{1}{m}\mathbf{C}_{WB}\,{}_B\mathbf{T} + {}_W\mathbf{g}, \tag{4.1c} $$

$$ {}_B\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}({}_B\mathbf{M} - {}_B\boldsymbol{\omega} \times \mathbf{J}\,{}_B\boldsymbol{\omega}), \tag{4.1d} $$

$$ \boldsymbol{\Omega}({}_B\boldsymbol{\omega}) = \begin{bmatrix} {}_B\boldsymbol{\omega}^{\times} & {}_B\boldsymbol{\omega} \\ -{}_B\boldsymbol{\omega}^{\top} & 0 \end{bmatrix}, \tag{4.1e} $$

where ${}_W\mathbf{g} := [0, 0, -9.81]^{\top}$ stands for the gravitational acceleration, m for the MAV mass, and $\mathbf{J}$ for its inertia tensor. The thrust vector ${}_B\mathbf{T} := [0, 0, T]^{\top}$ acting on the MAV CoM solely depends on the collective thrust T generated by the motors. This together with the moments ${}_B\mathbf{M}$ are considered as the control input $\mathbf{u} := [{}_B\mathbf{M}^{\top}, T]^{\top} \in \mathbb{R}^4$. The control state $\mathbf{x} := [{}_W\mathbf{r}_B^{\top}, \mathbf{q}_{WB}^{\top}, {}_W\mathbf{v}_B^{\top}, {}_B\boldsymbol{\omega}^{\top}]^{\top} \in \mathbb{R}^3 \times S^3 \times \mathbb{R}^6$, consists of the MAV position, orientation, linear and angular velocities respectively. We consider that the motor dynamics are significantly faster than the MAV body dynamics and are thus neglected. Based on the analysis provided in Section 2.2.1, we use the following model for the thrust and moment generated by the i^{th} motor:

$$ f_i = k_T \omega_i^2, \tag{4.2a} $$

$$ M_i = (-1)^{i+1} k_M f_i. \tag{4.2b} $$

Regarding the relationship between a PWM command and the generated thrust and moment, we follow the method provided in Section 2.2.1 where the coefficients of Equation (4.2) were identified using a thrust stand while the relationship between the PWM command and the achieved angular velocity was identified for different input voltage levels. As our approach also works for MAV platforms that can generate both positive and negative thrust, we identified two sets of coefficients: (i) for normal

motor rotation k_T^+, k_M^+ corresponding to positive thrust; (ii) inverted motor rotation k_T^-, k_M^- corresponding to negative thrust.

4.5.2 Nonlinear Model Predictive Control

For the control formulation, we define the following time-varying error functions for the position, linear and angular velocity, orientation and control input respectively:

$$\mathbf{e}_r = {}_W\mathbf{r}_B - {}_W\mathbf{r}_B^r, \tag{4.3a}$$
$$\mathbf{e}_v = {}_W\mathbf{v}_B - {}_W\mathbf{v}_B^r, \tag{4.3b}$$
$$\mathbf{e}_\omega = {}_B\boldsymbol{\omega} - \mathbf{C}_{BB^r}\, {}_{B^r}\boldsymbol{\omega}^r, \tag{4.3c}$$
$$\mathbf{e}_q = [\mathbf{q}_{WB}^{-1} \otimes \mathbf{q}_{WB}^r]_{1:3}, \tag{4.3d}$$
$$\mathbf{e}_u = \mathbf{u} - \mathbf{u}^r. \tag{4.3e}$$

Apart from the orientation error which is obtained through quaternion multiplication, the rest corresponds to the Euclidean difference between the actual and desired (here denoted with the superscript r) quantity.

We compute the optimal control input $\mathbf{u}^*$ sequence online by solving the following optimisation problem:

$$\mathbf{u}^* = \underset{\mathbf{u}_0,\dots,\mathbf{u}_{N_f-1}}{\operatorname{argmin}} \Big\{ \Phi(\mathbf{x}_{N_f}, \mathbf{x}_{N_f}^r) + \sum_{n=0}^{N_f-1} L(\mathbf{x}_n, \mathbf{x}_n^r, \mathbf{u}_n) \Big\}, \tag{4.4a}$$
$$\text{s.t. : } \mathbf{x}_{n+1} = \mathbf{F}_d(\mathbf{x}_n, \mathbf{u}_n), \tag{4.4b}$$
$$\mathbf{x}_0 = \hat{\mathbf{x}}, \tag{4.4c}$$
$$u_{\text{lb}} \le u_i \le u_{\text{ub}}, \quad i = 1, \dots, 4 \tag{4.4d}$$

, where N_f is the number of time steps, $\hat{\mathbf{x}}$ the known initial state, $\mathbf{F}_d$ the discrete-time version of the MAV dynamics given in (4.1) and u_{lb}, u_{ub} lower and upper bounds for the inputs u_i. We use quadratic costs for the final and intermediate terms defined as:

$$\Phi(\mathbf{x}_{N_f}, \mathbf{x}_{N_f}^r) = \mathbf{e}_r^\top \mathbf{Q}_r \mathbf{e}_r + \mathbf{e}_v^\top \mathbf{Q}_v \mathbf{e}_v + \mathbf{e}_q^\top \mathbf{Q}_q \mathbf{e}_q + \mathbf{e}_\omega^\top \mathbf{Q}_\omega \mathbf{e}_\omega, \tag{4.5a}$$
$$L(\mathbf{x}, \mathbf{x}^r, \mathbf{u}) = \mathbf{e}_r^\top \mathbf{Q}_r \mathbf{e}_r + \mathbf{e}_v^\top \mathbf{Q}_v \mathbf{e}_v + \mathbf{e}_q^\top \mathbf{Q}_q \mathbf{e}_q + \mathbf{e}_\omega^\top \mathbf{Q}_\omega \mathbf{e}_\omega + \mathbf{e}_u^\top \mathbf{Q}_u \mathbf{e}_u, \tag{4.5b}$$

with $\mathbf{Q} \succcurlyeq 0$ gain matrices of appropriate dimensions which are considered tuning parameters. In our implementation we use a 10 ms discretisation step and a

constant time horizon $T_f = 2.0\,\text{s}$. For the online computation of the optimal input we use the CT toolbox [Giftthaler et al., 2018] and the Gauss-Newton Multiple Shooting (GNMS) algorithm (outlined in [Giftthaler et al., 2017]) which result in an average computation time of 5.2 ms with standard deviation of 0.6 ms. At each GNMS iteration dynamically feasible state and input increments $\delta\mathbf{x}_n$, $\delta\mathbf{u}_n$ (for every stage n) around the state and input trajectories $\bar{\mathbf{X}} = \{\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \cdots, \bar{\mathbf{x}}_{N_f}\}$, $\bar{\mathbf{U}} = \{\bar{\mathbf{u}}_0, \bar{\mathbf{u}}_1, \cdots, \bar{\mathbf{u}}_{N_f-1}\}$ are computed. We obtain the dynamics of the minimal state perturbation $\delta\mathbf{x} := [\delta\mathbf{r}, \delta\boldsymbol{\theta}, \delta\mathbf{v}, \delta\boldsymbol{\omega}]^\top \in \mathbb{R}^{12}$ around the state $\bar{\mathbf{x}}$ by introducing the local quaternion perturbation $\mathbf{q} = \bar{\mathbf{q}} \otimes \boldsymbol{\delta}\mathbf{q}$ with $\boldsymbol{\delta}\mathbf{q} := \left[\operatorname{sinc}\left\|\frac{\delta\boldsymbol{\theta}}{2}\right\| \frac{\delta\boldsymbol{\theta}}{2}, \cos\left\|\frac{\delta\boldsymbol{\theta}}{2}\right\|\right]^\top$. The dynamics for the rotation vector $\delta\boldsymbol{\theta} \in \mathbb{R}^3$ are given by: $\dot{\delta\boldsymbol{\theta}} = {}_B\boldsymbol{\omega} - \frac{1}{2}{}_B\boldsymbol{\omega}^\times\delta\boldsymbol{\theta}$, while the rotation matrix $\mathbf{C}_{WB}$ can be approximated as: $\mathbf{C}_{WB} \approx \bar{\mathbf{C}}_{WB}(\mathbf{I} + \delta\boldsymbol{\theta}^\times)$. After each iteration we use the following rule for the state and input update: $\mathbf{x}' = [{}_w\bar{\mathbf{r}}_B + \delta\mathbf{r}, \bar{\mathbf{q}}_{WB} \otimes \delta\mathbf{q}, {}_W\bar{\mathbf{v}}_B + \boldsymbol{\delta}\mathbf{v}, {}_B\bar{\boldsymbol{\omega}} + \delta\boldsymbol{\omega}]^\top$.

4.5.3 Control allocation

As stated earlier, the control allocation problem involves mapping the control inputs $\mathbf{u}^*$ to feasible actuator commands $\mathbf{f} := [f_1, \ldots, f_N]^\top$. We tackle this by solving the following QP:

$$\mathbf{f}^* = \underset{\mathbf{f}}{\operatorname{argmin}} \left(\left\|\mathbf{A}\mathbf{f} - \mathbf{u}^*\right\|_{\mathbf{W}}^2 + \lambda \left\|\mathbf{f}\right\|_2^2 \right), \tag{4.6a}$$

$$\text{s.t.} : f_{\min} \leq f_i \leq f_{\max}, \quad i = 1, \ldots, N, \tag{4.6b}$$

where N is the number of motors. The allocation matrix $\mathbf{A} \in \mathbb{R}^{4\times N}$, which we will present later, depends on the MAV geometry and its motor coefficients, whereas $f_{\min}, f_{\max}$ correspond to the minimum and maximum attainable thrust. In order to prioritise the roll/pitch moments and the collective thrust over the yaw moment, we use the weighting matrix $\mathbf{W} \in \mathbb{R}^{4\times4}$. The scalar $\lambda \in \mathbb{R}^+$ is used such that solutions with smaller norms are preferred. When a feasible control input is commanded, solution of (4.6) coincides with the one obtained by using the pseudo-inverse of $\mathbf{A}$, namely $\mathbf{f} = \mathbf{A}^\dagger\mathbf{u}^*$.

Since we are interested in solving the control allocation problem for the general case where the motors can produce both positive and negative thrust, we introduce the vector $\mathbf{d} := [d_1, d_2, \cdots, d_N]^\top$ with $d_i \in \{0, 1\}, \forall i = 1, \ldots N$. We thus use the binary variables d_i to indicate whether the i^{th} motor is spinning in its intended nor-

mal direction–corresponding to positive thrust ($d_i = 0$)–or otherwise in the inverse ($d_i = 1$).

The original optimisation problem (4.6) is transformed to:

$$\mathbf{f}^*, \mathbf{d}^* = \underset{\mathbf{f},\mathbf{d}}{\operatorname{argmin}} \left(\left\| \mathbf{A}(\mathbf{d})\mathbf{f} - \mathbf{u}^* \right\|_{\mathbf{W}}^2 + \lambda \left\| \mathbf{f} \right\|_2^2 \right) \tag{4.7a}$$

$$\text{s.t.} : f_{\min}^+(1 - d_i) + f_{\min}^- d_i \leq f_i \leq f_{\max}^+(1 - d_i) + f_{\max}^- d_i, \tag{4.7b}$$

where the superscript $+$ or $-$ in $f_{\min}^+$, $f_{\min}^-$, $f_{\max}^+$, $f_{\max}^-$ has been used to indicate normal and inverted rotation respectively. The binary vector $\mathbf{d}$ which encodes the direction of rotation is now an optimisation variable and the allocation matrix $\mathbf{A}$ is a function of $\mathbf{d}$. For the case of the hexacopter with motor layout and normal motor rotation as shown in Figure 4.3, $\mathbf{A}(\mathbf{d})$ takes the following form:

$$\mathbf{A}(\mathbf{d}) = \begin{bmatrix} ls_{30} & l & ls_{30} & -ls_{30} & -l & -ls_{30} \\ -lc_{30} & 0 & lc_{30} & lc_{30} & 0 & -lc_{30} \\ k_M(d_1) & -k_M(d_2) & k_M(d_3) & -k_M(d_4) & k_M(d_5) & -k_M(d_6) \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \tag{4.8}$$

where l stands for the MAV arm length, $s_{30} = \sin(30^o)$, $c_{30} = \cos(30^o)$, $k_M(d_i) = (1 - d_i)k_M^+ + d_i k_M^-$.

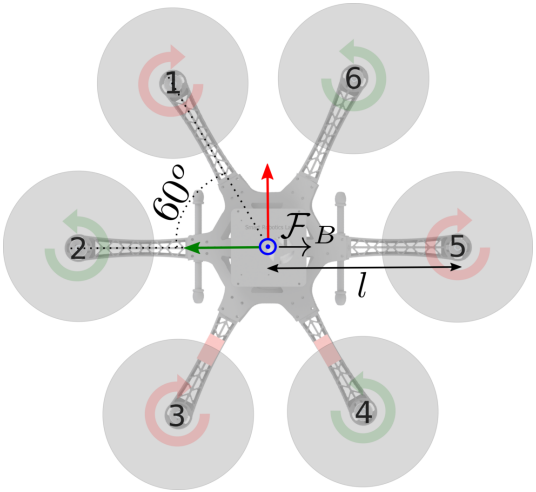


Figure 4.3: Motor layout of the hexacopter used in our experiments. Rotation arrows show the direction of rotation that corresponds to positive thrust.

The resulting optimisation described in (4.7) is a Mixed Integer Quadratic Programming (MIQP). However, since the possible values of $\mathbf{d}$ are finite (64 in the

case of a hexacopter), we can solve a single QP for every possible value of $\mathbf{d}$. The global optimum $\mathbf{f}^*$ of Equation (4.7) corresponds to the solution of the QP with the minimum cost. From a practical perspective solving 64 QPs instead of a single one does not affect significantly the overall control computation time as this is dominated by the computation of the optimal input $\mathbf{u}^*$ as described in the Section 4.5.2. This is owing to the small number of optimisation variables in a single QP tailored to the solver, CVXGEN [Mattingley and Boyd, 2012]. In our implementation, solving 64 QPs, storing the results in a vector and finally sorting it in ascending order consistently takes less than 0.4 ms. We acknowledge, however, that our method is more resource demanding compared to methods using the pseudoinverse which can be easily implemented on a microcontroller.

It was experimentally found that reverting the direction of rotation during flight is particularly impractical. This is because the motor dynamics are significantly slower when a direction change is commanded. As our control model does not capture this behaviour, we can prevent unnecessary direction change commands by augmenting the optimisation (4.7) similarly to [Brescianini and D'Andrea, 2018b] with the $\mathbf{f} \in \mathcal{F}_{\text{hyst}}$ constraint, where $\mathcal{F}_{\text{hyst}}$ is the set of rotor thrusts that does not require a per motor direction change when this has already happened during the past time interval t_{hyst}. The solution satisfying this constraint can be found with a single iteration over the vector of 64 possibilities. The threshold t_{hyst} can be iteratively decreased until a good (e.g. $\left\|\mathbf{Af} - \mathbf{u}^*\right\|_{\mathbf{W}}^2 < \epsilon$) solution is found.

4.6 Motor failure in a hexacopter

As mentioned earlier our control allocation can be used in multirotors that can generate both positive and negative thrust. This capability can be used e.g. to fly novel maneuvers as shown in Figure 4.4, or in order to increase the agility of the vehicle by providing negative acceleration along the z-axis of the $\underrightarrow{\mathcal{F}}_B$ frame. In this work we use this property to perform stable position and yaw control in the event of a motor failure.

It is important to clarify a common misconception regarding the fault tolerant capabilities of hexacopters with alternating rotor-turn directions as the one shown in Figure 4.3. It is often assumed that a hexacopter can handle a motor failure by simply switching off the opposite motor and consequently flying as a quadcopter.

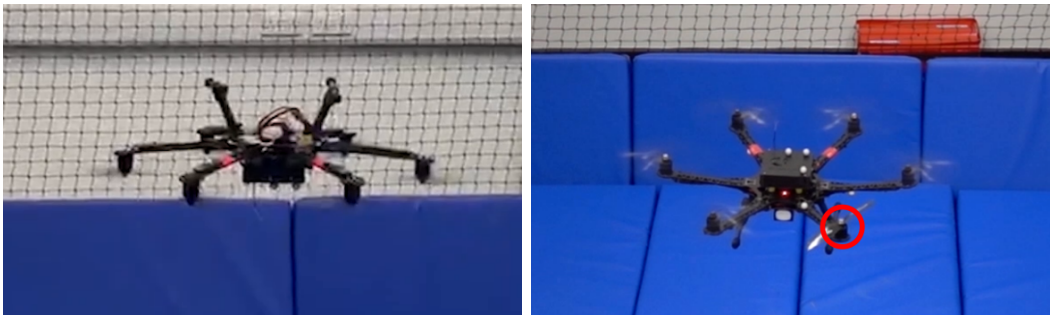


Figure 4.4: Potential uses of the reverse thrust capability. Left: provide negative acceleration along the z-axis of the $\underrightarrow{\mathcal{F}}_B$ thus enabling flying novel maneuvers such as inverted fligh. Right: stabilise the MAV position and yaw in the event of a motor failure.

However, this is not possible in practice. This is because, compared to a normal quadcopter where the opposite motors spin in the same direction, the opposite motors of a conventional hexacopter rotate in the inverse direction and thus introduce a positive coupling between the roll/pitch control moments M_x, M_y and yaw moment M_z. This can be verified by examining the set of feasible control inputs defined as:

$$\mathcal{V} = \left\{ \mathbf{A}(\mathbf{d})\mathbf{f} \in \mathbb{R}^4 | \ \mathbf{f} \in \mathbb{R}^6, \ \mathbf{d} \in \mathbb{Z}_2^6, \ f_{\min}(d_i) \leq f_i \leq f_{\max}(d_i), \ \forall i \in 1, \ldots, 6 \right\}, \quad (4.9)$$

with $\mathbf{A}(\mathbf{d})$ the allocation matrix defined in (4.8) and $\mathbf{f} := [f_1, \ldots, f_6]^\top, \mathbf{d} := [d_1, \ldots, d_6]^\top$, $f_{\min}(d_i) := f_{\min}^+(1 - d_i) + f_{\min}^- d_i$, $f_{\max}(d_i) := f_{\max}^+(1 - d_i) + f_{\max}^- d_i$ the quantities introduced in Section 4.5.3.

The set of feasible control inputs $\mathcal{V}$ lies in a four dimensional space and we thus visualise intersections with $M_z = 0$ in Figure 4.6, $T = mg$ in Figure 4.7 and $M_z = 0, T = mg$ in Figure 4.8 that correspond to operation around hover. In order to assess the controllability in the event of a single motor failure, we set $f_{\max} = f_{\min} = 0$ for motor #1. We consider the following different cases (also shown in Figure 4.5) depending on the capability of the remaining functioning motors to produce positive and negative thrust:

1. All the functioning motors #2,...,#6 are capable of producing both positive and negative thrust (shown in the left side of Figures 4.6, 4.7, 4.8),

2. The opposite of the failed motor is capable of producing both positive and negative thrust while the rest can only produce positive thrust (shown in the middle of Figures 4.6, 4.7, 4.8),

3. The functioning motors #2,...,#6 can only produce positive thrust (shown in the right side of Figures 4.6, 4.7, 4.8).

For the maximum and minimum motor thrust as well as the moment coefficients, we have used the ones corresponding to our platform and given in Table 4.2. More specifically, we set $f^{+}_{\max} = 7.5\,\mathrm{N}$, $f^{-}_{\min} = -5.0\,\mathrm{N}$, $f^{+}_{\min} = f^{-}_{\max} = 0.0\,\mathrm{N}$, $k^{+}_{M} = 1.4873 \times 10^{-2}\,\mathrm{Nm/N}$, $k^{-}_{M} = 2.7928 \times 10^{-2}\,\mathrm{Nm/N}$.

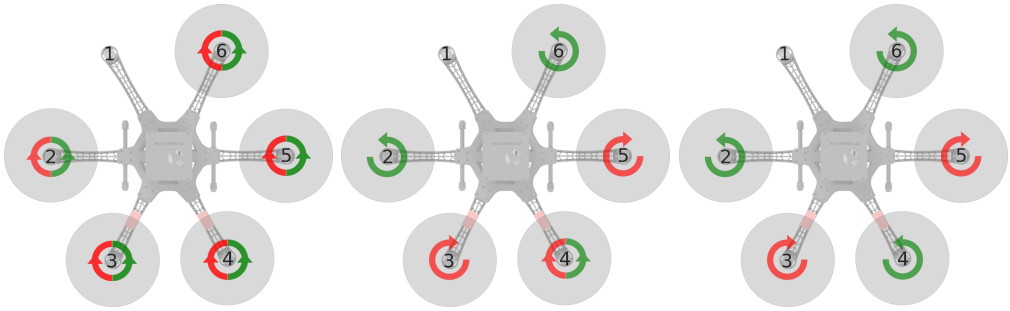


Figure 4.5: The three different cases considered for the controllability analysis. Left: the remaining functioning motors #2,...,#6 can produce both positive and negative thrust. Middle: the opposite of the failed motor can produce both positive and negative thrust while the rest only positive. Left: the remaining functioning motors #2,...,#6 can produce only positive thrust.

As it can be visually verified in Figures 4.6, 4.7, 4.8, the existence of bidirectional capable motors results a volume increase of the admissible control input set which can be interpreted as a metric of the vehicle's agility. By observing the Figures 4.7, 4.8 (right), we verify our claim that the 6D position and orientation of the vehicle are not controllable when the remaining functioning motors can produce only positive thrust. This is because the red asterisk, which corresponds to the nominal control input at hover $[{}_{B}\mathbf{M}^{\top}, T]^{\top} = [\mathbf{0}^{\top}, mg]^{\top}$, lies on the boundary of the admissible set. Any deviation from that point caused by e.g. a small disturbance or a model mismatch can result a control input outside the admissible set. An infeasible control input, cannot be executed by any control allocation algorithm not only the one presented in this work. In this case, our control allocation algorithm presented in Section 4.5.3 will prioritise the execution of the roll/pitch moments (which are crucial for position tracking) over the yaw and thus the yaw error will increase. This indicates that a hexacopter with five functioning motors capable of producing only positive thrust can perform a stable hover when it is initialised with exactly zero position, orientation, velocity error, its real model and state are known perfectly

and there are no external disturbances. Unfortunately, this cannot be achieved in a real system. An alternative solution, also shown in [Kamel et al., 2015] is to drop the control of yaw and perform position only control.

When the MAV is equipped with bidirectional capable motors as shown in left and middle of Figures 4.7, 4.8, the nominal control input at hover $[{}_B\mathbf{M}^\top, T]^\top = [\mathbf{0}^\top, mg]^\top$ lies inside the admissible control set and away from its boundaries providing enough control authority including the control of yaw. Clearly, enabling the bidirectional mode for all the remaining motors instead of a single one (the opposite of the failed one) increases the volume of the admissible set and thus the agility of the vehicle. However, in order to keep the motor direction inversions to the absolute minimum, the second option was preferred for the experiments presented in Section 4.8.2.

It is important to highlight that with our approach we can easily apply any of the two strategies discussed above. Specifically: (i) drop the yaw control when bidirectional motors are not available; (ii) enable the bidirectional mode for one or more motors. In the first case, flying with free heading is achieved by setting zero gains for the orientation term introduced in (4.3). In the latter case the bidirectional mode for one or more motors should be enabled by appropriately setting the upper and lower bounds in the optimisation defined in (4.7). An example of our MAV following setpoint commands (despite the loss of a motor) using the these two different modes is shown in Figure 4.9.

Another strategy for handling a motor failure was presented in [Schneider et al., 2012] with more experimental results in [Mazeh et al., 2018] and [Nguyen et al., 2019]. In these works, it is shown that a hexacopter with a non symmetric motor layout as shown in Figure 4.10 (right) can –in some cases– handle up to two lost motors without losing control of yaw. This approach has the clear advantage that bidirectional motors are not required. However, due to its asymmetric layout it can only handle a single failure of specific motors. Specifically, a failure of either motor or #1 or #6 results in an uncontrollable configuration. This is not the case for our chosen strategy since due to symmetry it can handle failure of any actuator. This is a very important property for a real system since the failure probability is similar for every actuator. Another disadvantage of any asymmetric layout is that the volume (and thus the overall MAV agility) of the admissible set $\mathcal{V}$ is less than the one corresponding to the symmetric one. This is why symmetric layouts are preferred in most multirotor platforms.

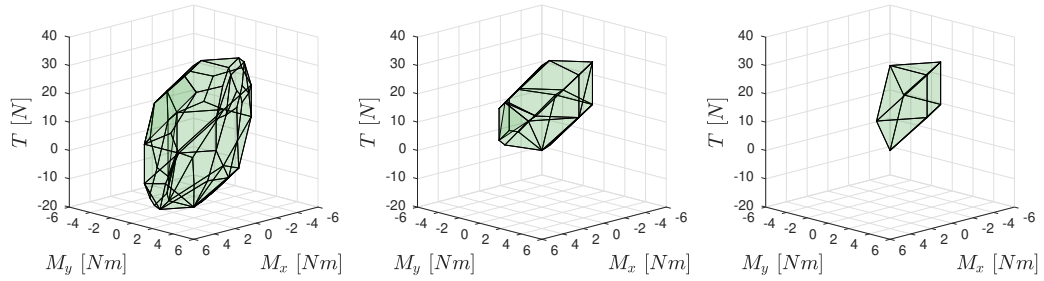


Figure 4.6: Visualisation of $\mathcal{V}$ defined in (4.9) cut at $M_z = 0$ for our hexacopter experiencing a failure of motor #1. Left: Bidirectional mode enabled for the remaining functioning motors. Middle: Bidirectional mode enabled for the opposite of the failed motor. Right: Bidirectional mode off. Volume of $\mathcal{V}$ (used as a metric of the MAV agility) is maximised when bidirectional mode is on (left plot).

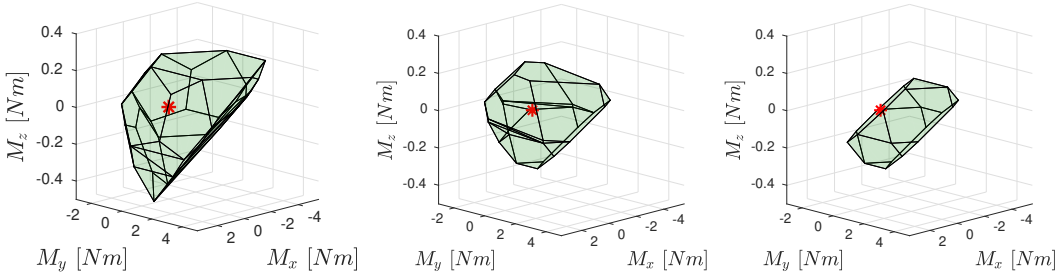


Figure 4.7: $\mathcal{V}$ cut at $T = mg$ for our hexacopter experiencing a failure of motor #1. Left: Bidirectional mode enabled for all the motors. Middle: Bidirectional mode on for the opposite of the failed motor. Right: Bidirectional mode off. The red asterisk, which corresponds to the nominal control input $[{}_B\mathbf{M}^\top, T]^\top = [\mathbf{0}, mg]^\top$ at hover, lies on the boundary of the set when the bidirectional mode is off (right plot).

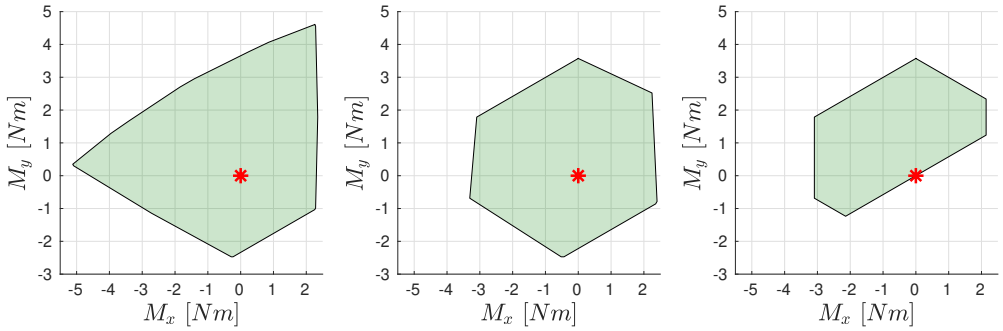


Figure 4.8: $\mathcal{V}$ cut at $M_z = 0, T = mg$, for our hexacopter experiencing a complete failure of motor #1. Left: Bidirectional mode enabled for all the functioning motors. Middle: Bidirectional mode enabled for the opposite of the failed motor. Right: Bidirectional mode off. The control input at hover (red asterisk) lies on the boundary of the admissible set when the bidirectional mode is off (right plot). Any deviation from that point will result in an infeasible control command which cannot be tracked. The same point lies inside $\mathcal{V}$ when the bidirectional mode is on, providing enough control authority for stable position and yaw control.

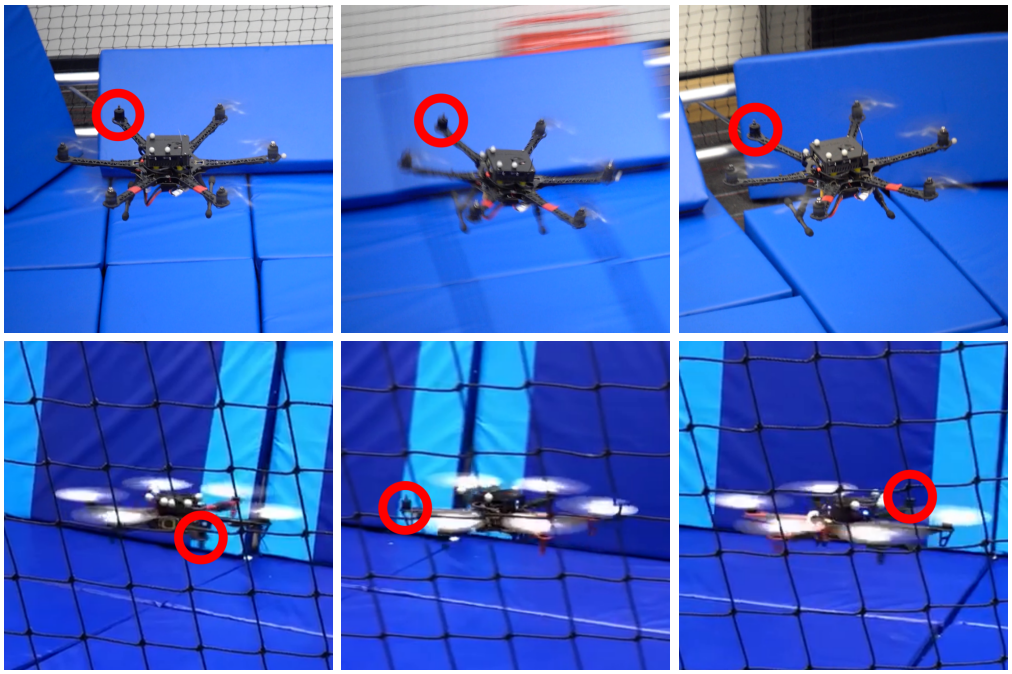


Figure 4.9: Waypoint following despite the loss of a single motor. When bidirectional motors are available (top) the MAV maintains control of yaw and position. Yaw control is lost (bottom) when the remaining motors can only produce positive thrust.

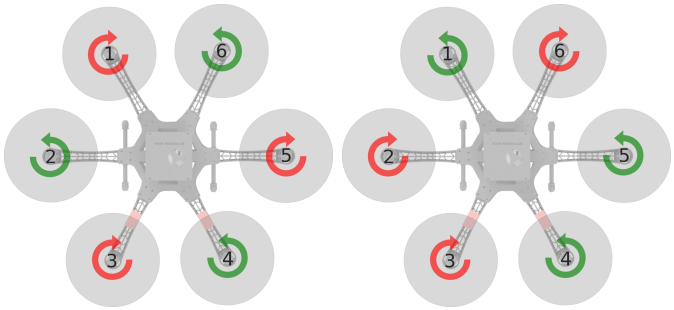


Figure 4.10: Left: The symmetric motor layout used in this work. Right: The asymmetric layout proposed in [Schneider et al., 2012] which can handle a single motor failure for four out of the six motors, without reverting the direction of rotation. In the event of no failure, the volume of the admissible set $\mathcal{V}$ is greater (resulting a more agile MAV) in the symmetric layout compared to the asymmetric one. This is why symmetric layouts are preferred in most multirotor platforms.

To sum up, we consider that our algorithms are generic enough and can be applied to any of the strategies of handling a single motor failure discussed above (e.g. enable bidirectional mode or fly with free heading when bidirectional hardware is not available). Additionally, they can be applied to a hexacopter with any motor layout (including non symmetrical motor rotation, or tilted motors) as long as the relationship between actuator forces $\mathbf{f}$ and control inputs $\mathbf{u}$ is given by $\mathbf{u} = \mathbf{A}\mathbf{f}$ with $\mathbf{A}$ the allocation matrix that depends on the vehicle's structure. Combination of the above (e.g. asymmetric motor layout with bidirectional motors) is also possible with the appropriate allocation matrix and actuator limits adaptation.

4.7 Fault identification

Our goal is to online estimate whether one or more motors have failed (consequently resulting in applied moments and collective thrust different from the ones commanded). To do so, we introduce the health variable $h_i \in \mathbb{R}$ for each individual motor i and assume that the effective force acting on the MAV generated from the i^{th} motor is $f_i^e = L(h_i)f_i$, where $L(h) = \frac{1.05}{1+e^{-h}}$ is the logistic function shown in Figure 4.11 and f_i corresponds to the respective motor thrust. Intuitively, we expect that $L(h_i) = 1$ for a healthy motor and $L(h_i) \to 0$ for a stopped one. We implemented an EKF that estimates the set of health variables h_i online for each individual motor. The effective body torques and collective thrust are now given by:

$$\begin{bmatrix} {}_B\mathbf{M} \\ T \end{bmatrix} = \mathbf{A} \begin{bmatrix} L(h_1)f_1 & \cdots & L(h_6)f_6 \end{bmatrix}^\top, \tag{4.10}$$

where $\mathbf{A}$ is the allocation matrix defined in (4.8), which depends on the moment coefficients and the motor direction of rotation.

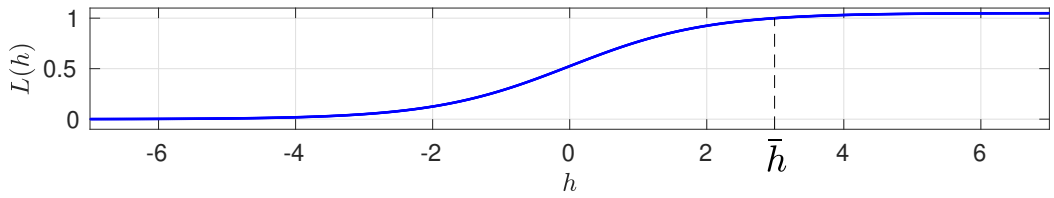


Figure 4.11: The logistic function used for the EKF. Notice, that it is appropriately scaled such that $\bar{h} = L^{-1}(1)$ has finite value.

In our EKF, we use the following prediction model :

$$ {}_B\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}({}_B\mathbf{M} - {}_B\boldsymbol{\omega} \times \mathbf{J}\,{}_B\boldsymbol{\omega}) + \boldsymbol{w}_\omega, \tag{4.11a} $$

$$ \dot{f}_i = \frac{1}{\tau_f}(f_i^r - f_i + w_f), \forall i = 1, \ldots, 6, \tag{4.11b} $$

$$ \dot{h}_i = \frac{1}{\tau_h}(\bar{h} - h_i) + w_h, \forall i = 1, \ldots, 6. \tag{4.11c} $$

The noises $\boldsymbol{w}_\omega$, w_h, and w_f are Gaussian white noise processes with densities σ_ω, σ_h, and σ_f, respectively. Equation (4.11a) corresponds to the MAV angular rate dynamics while (4.11b) to the first order motor thrust dynamics with f_i^r the per-motor reference thrust as given by the control allocation and τ_f the model time constant. The first order health variables dynamics (4.11c) with time constant τ_h, model how fast a failure can occur as well as ensure that $\bar{h}$ is an attraction point for the h_i estimates.

The measurement model is given by:

$$ \mathbf{z}_\omega = {}_B\boldsymbol{\omega} + \boldsymbol{v}_{\tilde{\omega}}, \tag{4.12a} $$

$$ z_T = T + v_T, \tag{4.12b} $$

with $\boldsymbol{v}_{\tilde{\omega}} \sim \mathcal{N}(0, \sigma_{\tilde{\omega}}^2\mathbf{I})$ and $v_T \sim \mathcal{N}(0, \sigma_{\tilde{T}}^2)$ modelling the observation noise. The measurements ${}_B\tilde{\boldsymbol{\omega}}$ and $\tilde{T}$ required for the EKF update are obtained using the on-board IMU. For ${}_B\tilde{\boldsymbol{\omega}}$ we use the bias corrected gyro measurements and for measured collective thrust $\tilde{T}$ we use $\tilde{T} \approx ma_z$ with m denoting the known mass of the MAV and a_z the accelerometer measurement along the z axis. Notice that our observation model for the collective thrust does not account for the ${}_B\boldsymbol{\omega} \times {}_B\mathbf{v}$ term which appears in the Body frame expressed linear acceleration dynamics. This was a design choice, allowing the EKF to rely exclusively on inertial measurements and thus able to be implemented as an algorithmic only update on any multirotor.

The values for the noise parameters and model constants are given in Table 4.1.

In order to avoid false positives due to e.g. inaccurate model, we use the estimated value of h_i and its estimated uncertainty. We consider a motor failed when $L(h_i + 3\sigma_i) < 0.5$ (with σ_i denoting the health state standard deviation obtained as a marginal from the state covariance matrix). When the above inequality is true we update the control allocation algorithm by setting $f_{\min} = f_{\max} = 0$ for the failed motor and enabling the bidirectional mode for the opposite. In the event of a detected failure, the number of functional motors is reduced by one. It makes

Description	Symbol	Value	Unit
Angular rate model noise density	σ_ω	3.16	$\mathrm{rad}/(\mathrm{s}\sqrt{\mathrm{Hz}})$
Thrust model noise density	σ_f	0.94	$\mathrm{N}/\sqrt{\mathrm{Hz}}$
Thrust model time constant	τ_f	0.01	s
Health variables model noise density	σ_h	0.31	$\sqrt{\mathrm{Hz}}^{-1}$
Health variables model time constant	τ_h	0.3	s
Health variables model constant	$\bar{h}$	2.99	-
Angular rate measurement noise	$\sigma_{\tilde{\omega}}$	0.01	rad/s
Collective thrust measurement noise	$\sigma_{\tilde{T}}$	0.1	N

Table 4.1: Numeric values of the fault detection EKF Parameters.

sense then to continue estimating the health variables for the remaining functional motors. This is done by introducing appropriate equality constraints $f_i^r = f_i = 0$ for the failed motor which can be incorporated (following the approach discussed in [Simon, 2010]) in the EKF through appropriate reduction of the model (4.11).

4.8 Experimental results

The experiments presented in this section were performed using our custom built hexacopter described in Section 2.2 with the bidirectional propulsion system #2 described in Section 2.2.3. The state estimates are provided by the EKF described in Section 2.4.1 with pose measurements obtained by a Vicon motion capture system. The parameters of the control model and the control allocation are provided in Table 4.2.

In the following we present two different types of experiments. In Section 4.8.1 we show how the controller exploits the non-linear MAV model to perform step commands in position and yaw. In Section 4.8.2, we further test the failure detection and recovery strategy discussed previously by manually switching off a single motor while flying.

4.8.1 Waypoint tracking

In both experiments presented, the MAV is first commanded to fly from a waypoint with reference position ${}_W\mathbf{r}_B^r = [1, 0, 3]^\top$ and orientation $\mathbf{q}_{WB}^r = [0, 0, 0, 1]^\top$ to a waypoint with reference position ${}_W\mathbf{r}_B^r = [-1, 0, 1]^\top$ and orientation $\mathbf{q}_{WB}^r = [0, 0, 1, 0]^\top$. This corresponds to a 2 m position jump in x and z and 180° in yaw.

Description	Symbol	Value	Unit
MAV mass	m	1.871	kg
MAV inertia in x axis	J_x	0.032	$\mathrm{kg/m^2}$
MAV inertia in y axis	J_y	0.034	$\mathrm{kg/m^2}$
MAV inertia in z axis	J_z	0.070	$\mathrm{kg/m^2}$
Normal rotation thrust coefficient	k_T^+	9.7408×10^{-6}	$\mathrm{N/(rad/s)^2}$
Normal rotation moment coefficient	k_M^+	1.4873×10^{-2}	Nm/N
Inverted rotation thrust coefficient	k_T^-	6.3309×10^{-6}	$\mathrm{N/(rad/s)^2}$
Inverted rotation moment coefficient	k_M^-	2.7928×10^{-2}	Nm/N
Min thrust in normal motor rotation	$f_{\min}^+$	0.0	N
Max thrust in normal motor rotation	$f_{\max}^+$	7.5	N
Min thrust in inverted motor rotation	$f_{\min}^-$	-5.0	N
Max thrust in inverted motor rotation	$f_{\max}^-$	0.0	N
Motor inversion hysteresis threshold	t_{hyst}	0.1	s

Table 4.2: Numeric values of control model parameters.

Figures 4.12 and 4.13 show the position and orientation response for the same experiment using low $\mathbf{Q}_q = 5\,\mathbf{I}_{3\times3}$ and high $\mathbf{Q}_q = 65\,\mathbf{I}_{3\times3}$ orientation gains respectively. Both maneuvers are executed in less than 2.5 s with linear accelerations exceeding $15\,\mathrm{m/s^2}$. In the high gain case, the yaw step is mainly performed by a half flip in roll and pitch as shown in Figure 4.14. This shows an advantage of the model based design over traditional approaches (e.g. PID) where a yaw maneuver is achieved by applying M_z moment and subsequently rotating around the z_B axis. Additionally, compared to approaches such as [Brescianini and D'Andrea, 2018] where the shortest rotation (between the current orientation and the reference one is preferred), our NMPC always chooses the fastest rotation (which does not always coincide with the shortest) based on the MAV dynamics.

Figures 4.12 and 4.13 additionally show the position predictions computed by the NMPC. The predictions are updated at every controller iteration and thus start from different initial states. In an ideal scenario, where the control and real model are identical, one would expect all the predicted and real trajectories to perfectly overlap. In reality, these two are close to its other but never coincide perfectly, as the control model is an approximation (e.g. control input set approximated as a 4D hyperrectangle, gyroscopic moments and motor dynamics ignored) of the unknown real one.

Another observation, is that regarding the predicted and actual response in the

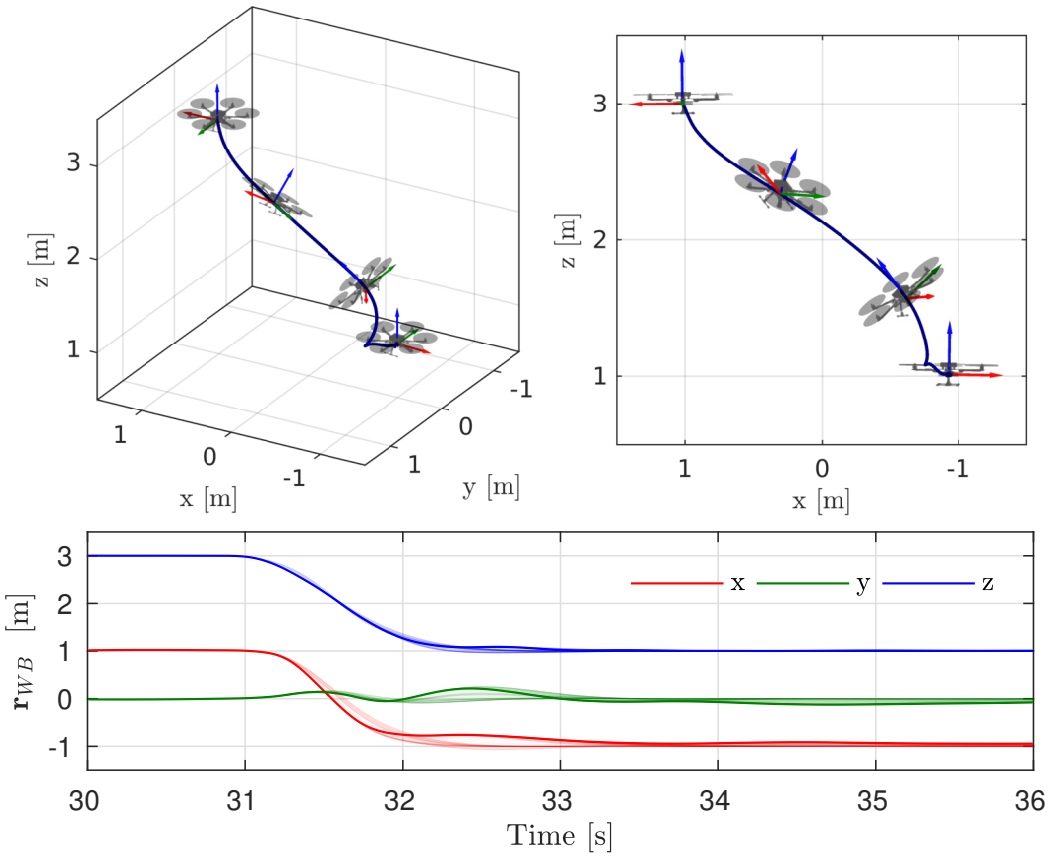


Figure 4.12: The MAV position while performing the 2 m and 180° jump in position and yaw with low orientation gains. Top: Two different views of the position and orientation response. Bottom: The per-axis position response (solid lines) alongside the NMPC predictions (faint lines). For visualisation clarity, here we plot the NMPC predictions at 20 Hz compared to their actual update rate of 100 Hz.

y axis. It can be seen that position in y temporarily deviates from its zero reference. This behaviour is also reflected in the corresponding predictions and is the result of the existence of coupling between the vehicle's position and attitude. Even though not tested in this work, this shows the potential use of a similar optimisation framework as a discrete time dynamically feasible planner.

4.8.2 Fault detection and recovery

We tested the failure detection and autonomous recovery in two different scenarios where one motor was switched off (i) while hovering and another (ii) while the MAV was following setpoint commands. In the first case motor #1 was manually deactivated by overwriting the thrust command produced by the control allocation

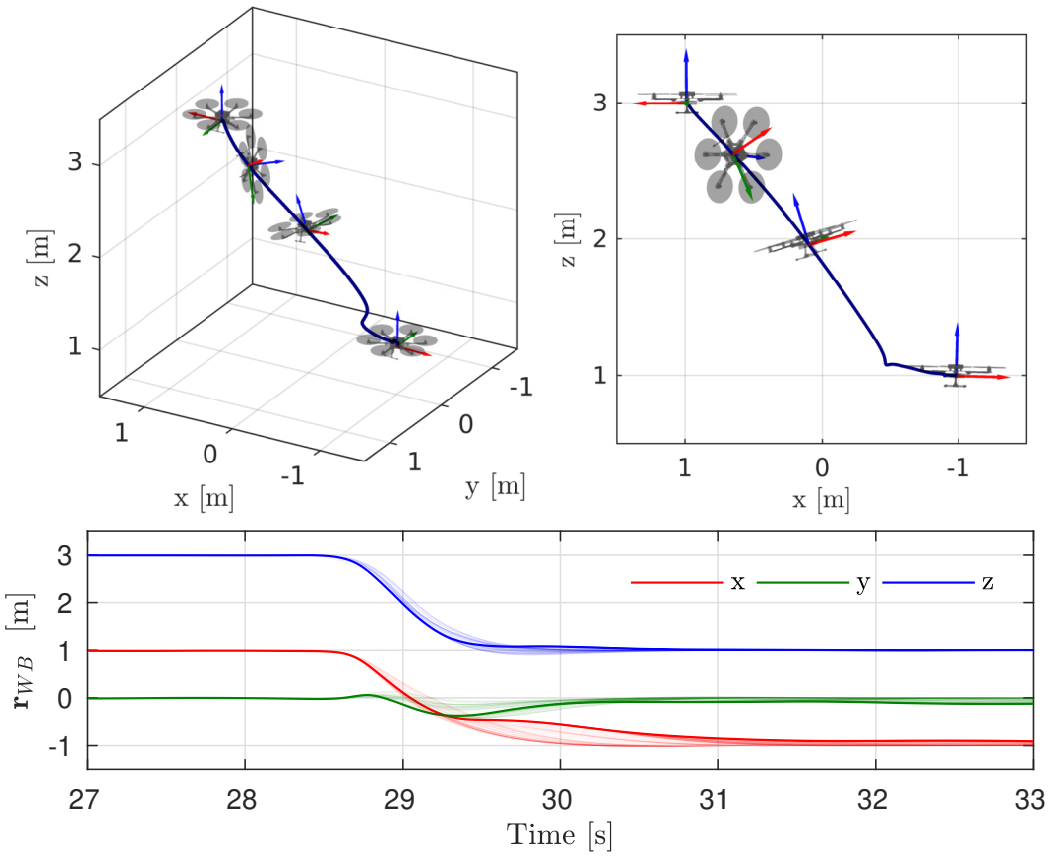


Figure 4.13: The MAV position while performing the 2 m and 180° jump in position and yaw with high orientation gains. Top: Two different views of the position and orientation response. Bottom: The per-axis position response (solid lines) alongside the NMPC predictions (faint lines). For visualisation clarity, here we plot the NMPC predictions at 20 Hz compared to their actual update rate of 100 Hz.

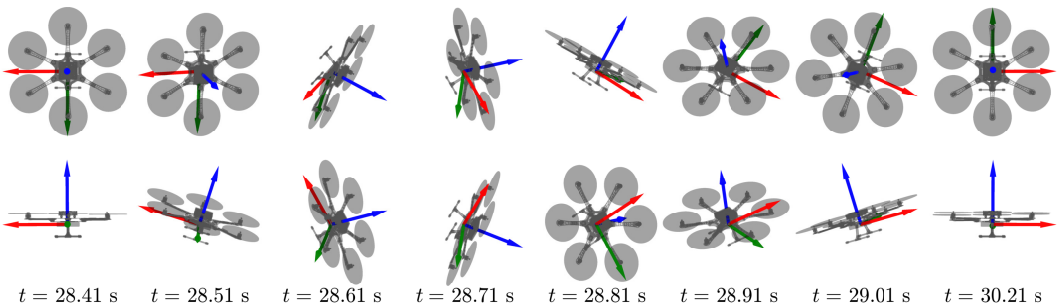


Figure 4.14: The MAV orientation while performing the position-yaw step with high orientation gains. Top: View from above (projection on the x-y plane). Bottom: View from the side (projection on the x-z plane). The yaw step is mainly achieved by a half flip in roll and pitch.

and sending a zero one instead. For the setpoint case, motor #3 was automatically deactivated once the MAV was at a specific point in space. Due to the symmetric motor layout we claim that deactivating any other motor would result in very similar behavior. The failure was detected automatically and the failsafe activated based on the methods described in Sections 4.6 and 4.7.

Figures 4.15 and 4.16 show the MAV position, the absolute yaw error and the online estimated health status of each motor for the failure while hovering experiment. The failure was injected at $t = 23.68\,\text{s}$ and identified at $t = 23.85\,\text{s}$ resulting in an overall height loss of 0.3 m. We claim that even this minimal height loss is mainly because of the slow motor dynamics due to the direction inversion command. This can be justified as in Figure 4.15 there is no visible red line (indicating operation with a motor failure that has not been identified yet). The fast detection can be also verified by the video[1] of the conducted experiments where the opposite motor switches direction of rotation before the failed motor comes to a full stop. We repeated the same experiment twice and provide the 3D position data in Figure 4.17. For these two additional experiments, the worst case detection delay and altitude loss was 0.18 s and 0.4 m respectively.

For the waypoint experiments, we used four distinct waypoint commands (forming a 2.0 m wide square) which were passed consecutively to the controller. Yaw and height reference were kept constant to 0° and 2 m respectively. We define a sphere around every waypoint as a tolerance to signal a waypoint as complete. That is, when the MAV is inside the corresponding sphere, the next waypoint is passed to the controller. Figures 4.18 and 4.19 show the MAV position, the absolute yaw error and the per-motor estimated health status. Similarly, to the hovering experiments the failure was identified within 0.16 s and resulted in a 0.5 m height loss. Two additional experiments are illustrated in Figure 4.20 with a worst case detection delay and height loss of 0.18 s and 0.52 m.

In both cases tracking of position and yaw is maintained and the MAV can recover to the hover point for the first case and fly to the next waypoints for the second. Not surprisingly, the 5-motor asymmetric configuration results in degraded tracking performance especially for yaw.

Regarding the health status variables of the functioning motors, these always

[1] See https://youtu.be/cAQeSZ3tIqY

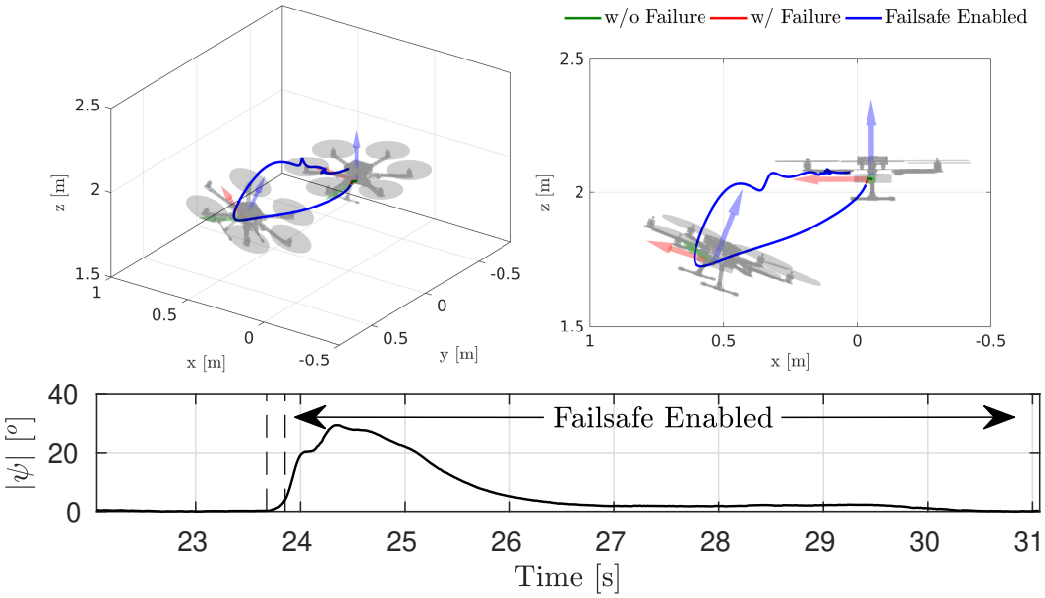


Figure 4.15: Autonomous fault detection and recovery during hover. The failure is identified within 0.17 s after the manual deactivation of motor #1. Position and yaw tracking is maintained and the MAV recovers to the hovering spot.

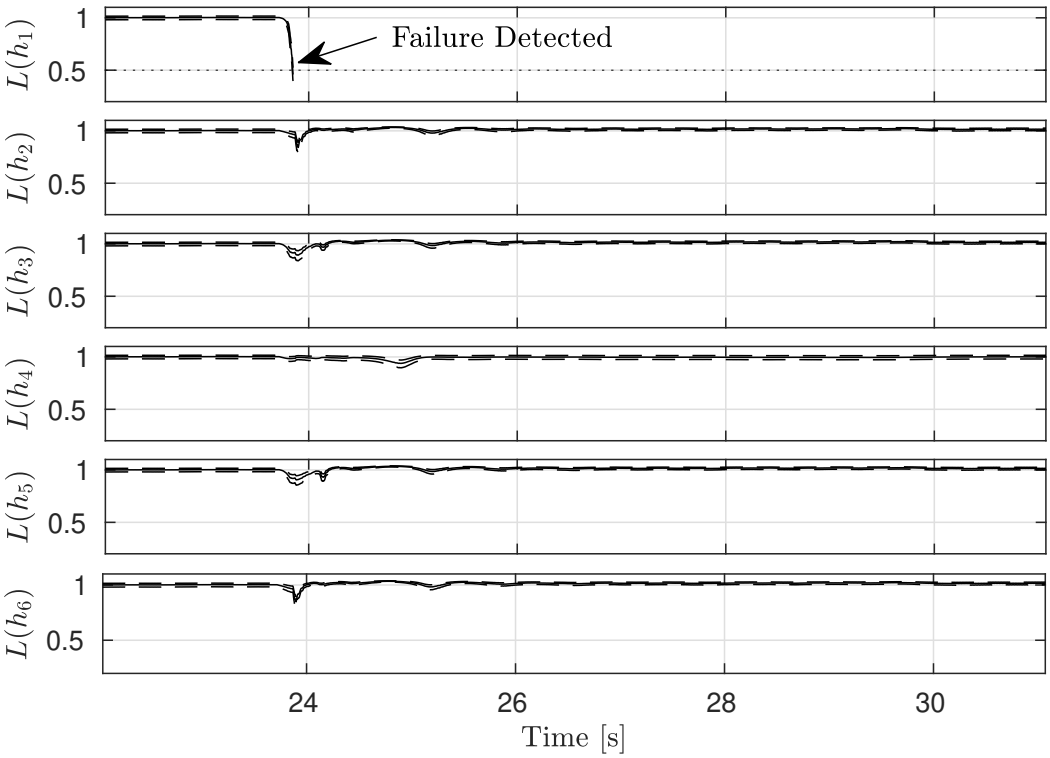


Figure 4.16: The online estimates of the health status $L(h_i)$ and their corresponding 3σ upper and lower bounds for the hovering experiment. The upper bound $L(h_1 + 3\sigma_1)$ for motor #1 drops below the 0.5 threshold within 0.17 s after the motor deactivation at $t = 23.68$ s

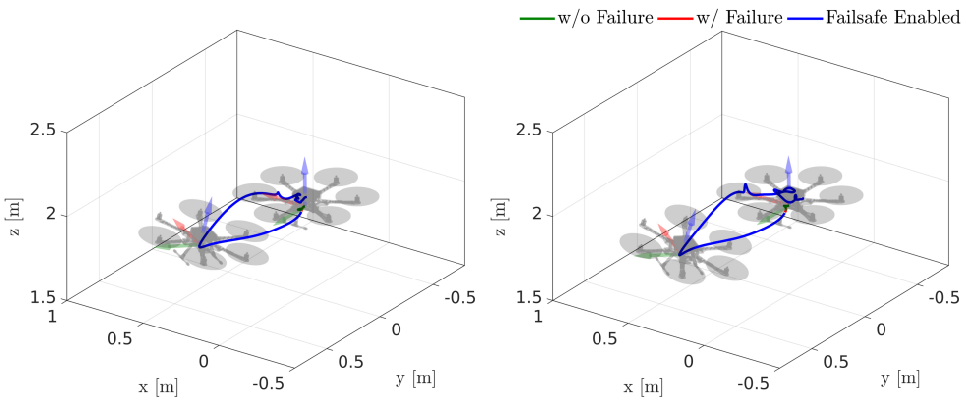


Figure 4.17: Autonomous fault detection and recovery for two additional hovering experiments. The detection delay was 0.16 s for the experiment on the left and 0.18 s for the right while the respective altitude loss was 0.35 m and 0.4 m respectively.

remain close to 1 for both types of experiments. However, in the setpoint case, there exist some short-in-duration deviations from 1. These spikes correspond to time instants when large angular accelerations were executed. We consider the main reason for this behaviour to be the mismatch between the EKF prediction model (which does not take into account less significant phenomena, such as the gyroscopic moments) and the measurement model (which does not take into account the ${}_B\boldsymbol{\omega} \times {}_B\mathbf{v}$) and the real ones. Despite these temporary inaccuracies, the 3σ upper bound was always greater than 0.8 and thus unable to trigger a false positive.

4.9 Discussion

In this Chapter we presented a nonlinear model based controller with the body torques and collective thrust as the control input. We showed how this can be used for aggressive maneuvers that fully exploit the dynamic capabilities of the platform. Compared to other approaches such as [Kamel et al., 2017a, Falanga et al., 2018], we drop the assumption regarding the lack of coupling between different DoF. Consequently, there is no need for a separate attitude or rate controller and the MAV position, velocity and orientation are controlled by a single control block. This further enables the use of the same algorithm without any modifications as an attitude, rate or mixed mode (e.g. linear velocity and yaw rate) controller by setting non-zero gains $\mathbf{Q}$ to the terms in 4.5 which contain the states we are interested in and zero for the rest. This property was particularly handy during the execution of autonomous experiments, where the same algorithm was used as a safety backup

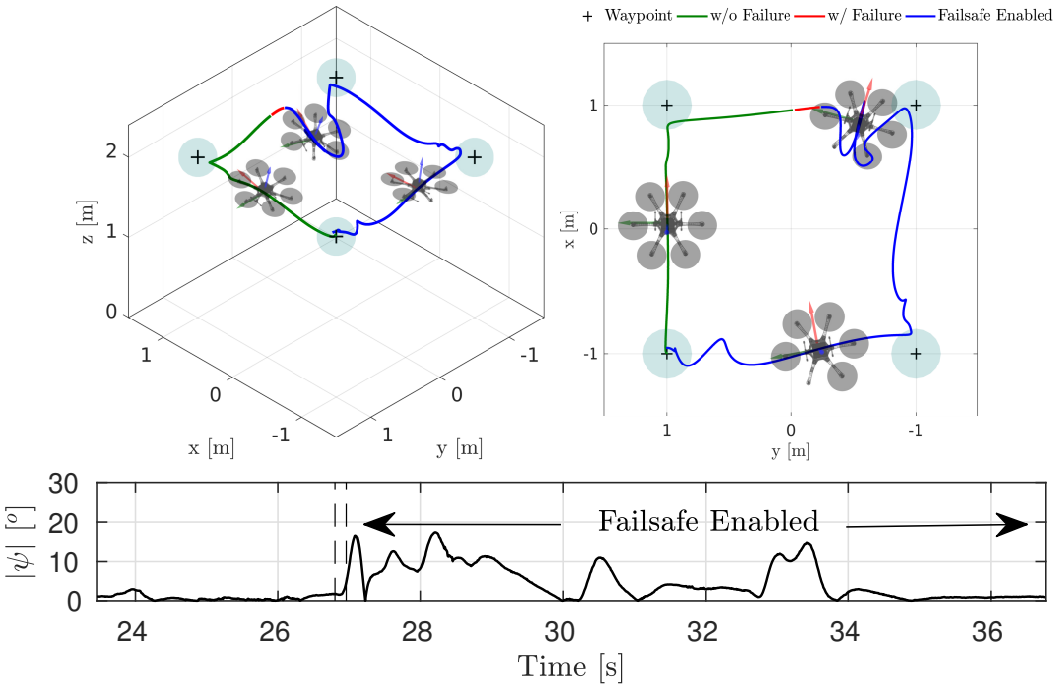


Figure 4.18: Autonomous fault detection and recovery while following setpoint commands. The failure is identified within 0.16 s after the manual deactivation of motor #3. Position and yaw tracking is maintained and the MAV continues to follow the setpoint commands.

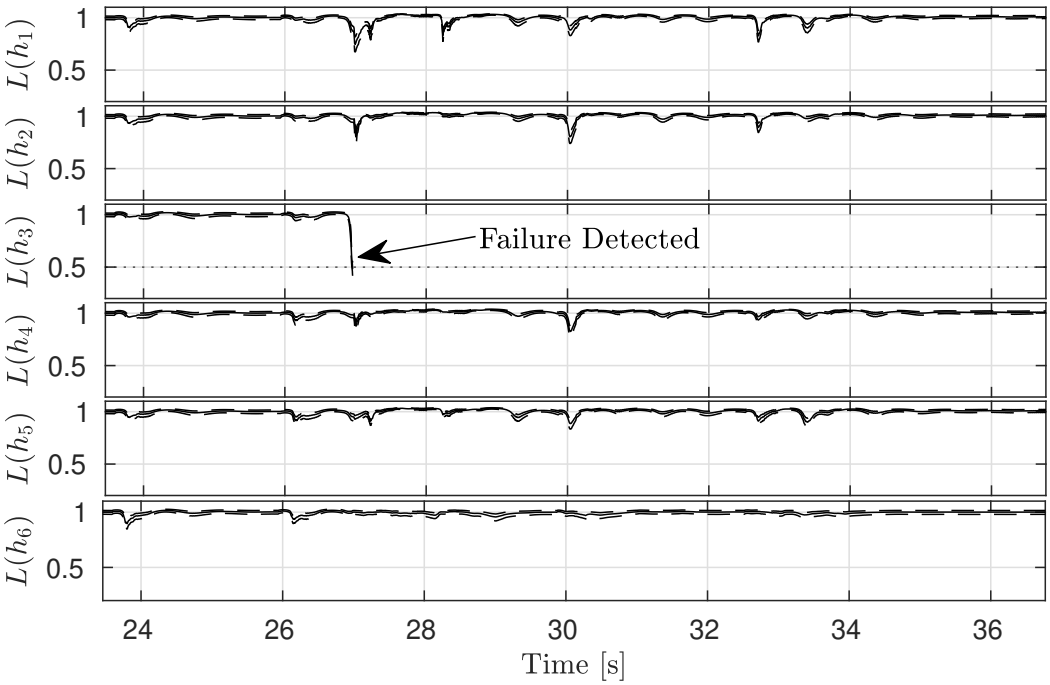


Figure 4.19: The online estimates of the health status $L(h_i)$ and their corresponding 3σ upper and lower bounds for the setpoint experiment. The upper bound $L(h_3 + 3\sigma_3)$ for motor #3 drops below the 0.5 threshold within 0.16 s after the motor deactivation at $t = 26.80$ s

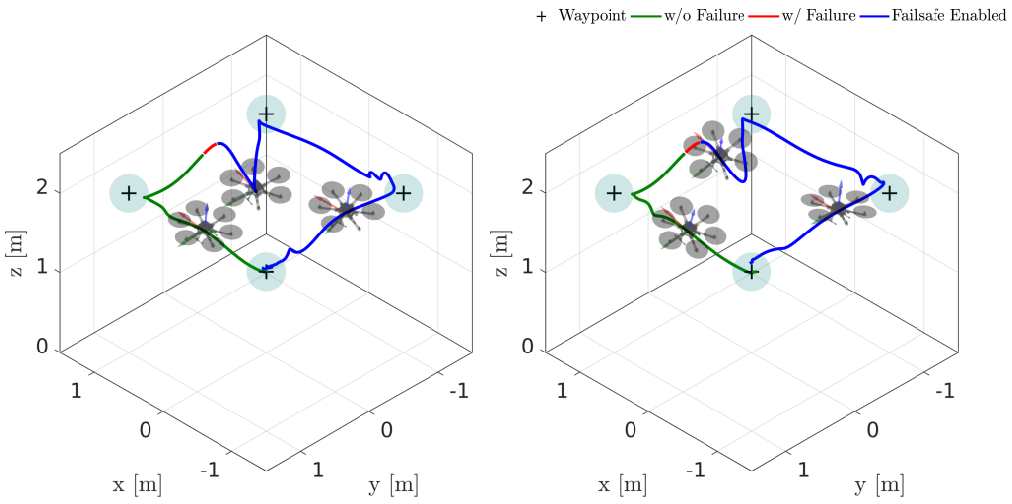


Figure 4.20: Autonomous fault detection and recovery for two additional setpoint experiments. The detection delay was 0.17 s for the experiment on the left and 0.18 s for the right while the attitude loss was 0.52 m and 0.5 m respectively.

attitude controller receiving orientation commands by a human operator.

We further proposed an optimisation based control allocation scheme that maps the control inputs into feasible actuator commands. We showed how this can be further extended for a platform capable of generating both positive and negative thrust. We used this capability in order to stabilise the yaw of our hexacopter experiencing a motor failure.

Finally, we developed an EKF that can be used for fast motor failure detection. This solely relies on inertial measurements and can be implemented as an algorithmic only update on any MAV and can be accordingly extended with motor speed and current measurements.

We argue that all the algorithms presented in this section can be seamlessly implemented on multirotors with different number or types of actuators. In that case the allocation matrix $\mathbf{A}$ (used in the control allocation and the failure detection EKF) has to be adapted based on the MAV geometry. Additionally, the strategy in the event of an actuator failure has to be changed accordingly (e.g. no need for direction inversion in the case of an octacopter or fly without yaw control in the case of a quadrotor). An effective strategy on the MAV type, can be adopted by examining the admissible set as discussed in Section 4.6. The NMPC block however, remains the same irrespectively of the type of the underactuated MAV at hand. This has the

clear advantage, that the computational complexity of the most resource demanding part remains unaffected by the number of MAV actuators.

As part of future work, the extension of the model presented in this Chapter with the effect of unmodeled – in the current implementation – phenomena (e.g. gyroscopic moments, rotor drag) would be of high interest. Additionally, including the motor dynamics in a control model with the individual motor thrust as control input, will eliminate the need of the allocation algorithm. This will also eliminate the case of not feasible control inputs introduced by the approximation of the feasible control input set defined in (4.9) as a 4D hyperrectangle.

Chapter 5

Nonlinear Model Predictive Control for aerial manipulation

Contents

Parts of this Chapter appear in: Tzoumanikas, D., Graule, F., Yan, Q., Shah, D., Popovic, M., and Leutenegger, S. (2020). Aerial Manipulation Using Hybrid Force and Position NMPC Applied to Aerial Writing. *In Proceedings of Robotics: Science and Systems (RSS).* [Tzoumanikas et al., 2020a]

5.1 Introduction

In Chapter 4 we presented a collection of algorithms required for safe and aggressive flight of an MAV for tasks which do not require physical interaction with the environment. In a more complex scenario, an MAV apart from being able to safely navigate in space, it should be able to physically interact with its environment by carrying appropriate mechanisms. Example applications include inspection of infrastructure like bridges or manufacturing plants [Ikeda et al., 2017, Ángel Trujillo et al., 2019, Bodie et al., 2019], physical interaction through MAV-attached tools like grinding, welding, drilling and other maintenance work in hard-to-reach places [Papachristos et al., 2014b, Bodie et al., 2019] and autonomous pick-up and transport of objects [Kessens et al., 2016, Augugliaro et al., 2014]. Another possible application includes the AABM scenario introduced in Chapter 3 where multiple MAVs were used to 3D print a structure.

However, combining the maneuverability of aerial vehicles with the manipulation capabilities of robotic arms comes at the cost of additional control complexity due to the coupling of the dynamics of the two systems. The requirements for high precision in real world aerial manipulation applications further increases the difficulty of the task.

In this Chapter we extend the NMPC presented in the previous one for a combined MAV-arm system performing an aerial manipulation task. We formulate a hybrid control model for the combined system which incorporates interaction forces acting on the end effector. We explain the practical implementation of our algorithm and show extensive experimental results of our custom built system performing multiple 'aerial-writing' tasks on a whiteboard revealing millimetre-level accuracy. An instance of the conducted experiments is shown in Figure 5.1.

5.2 Related work

Aerial manipulation systems can be broadly distinguished based on the MAV type (as being omnidirectional or underactuated) and the end effector (as being fixed or moving). In general, using an omnidirectional MAV to fulfill complex aerial tasks does not require a moving end effector as the necessary 6 DoFs are provided by the MAV itself. Examples include the works presented by [Brescianini and D'Andrea, 2016, Brescianini and D'Andrea, 2018a, Ryll et al., 2019]. [Brescianini

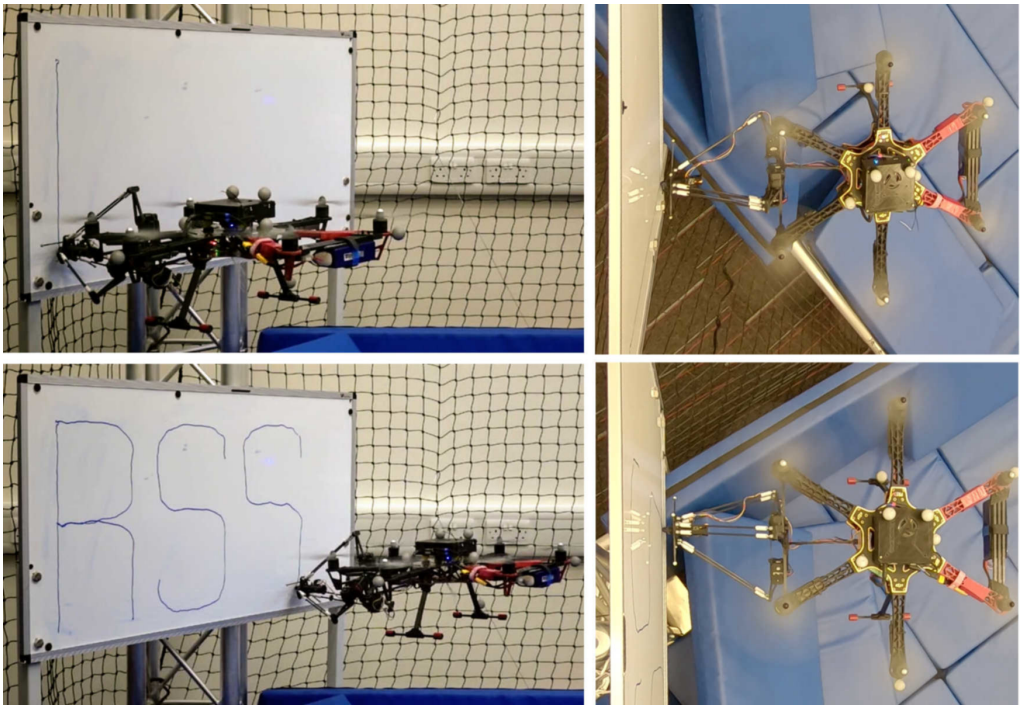


Figure 5.1: Our MAV-arm system performing an 'aerial-writing' task. Motion of the MAV and arm is jointly controlled by a single control block.

and D'Andrea, 2016] show an omnidirectional MAV called OmniCopter that achieves 6-DoF motion by using eight fixed rotors in a non co-planar configuration. In a subsequent study [Brescianini and D'Andrea, 2018a], this platform is used with a fixed end effector to fetch moving objects. Using a similar approach with a fixed configuration of tilted rotors, [Ryll et al., 2019] propose a novel paradigm to control all 6 DoF of the MAV while using a rigid end effector to exert forces and torques independently. The system is demonstrated in numerous experimental tasks, e.g. surface sliding.

Following a different approach, [Kamel et al., 2018] develop a setup of six rotors which tilt individually to control the direction of their thrust vector. [Bodie et al., 2019] leverage this system, named Voliro, to solve a variety of aerial manipulation tasks with a rigidly mounted, low complexity end effector. The authors further show precise force control when in contact with unstructured environments while running online visual-inertial state estimation. While this platform allows for accurate 6-DoF flight and longitudinal force exertion with a relatively simplistic control method, it is mechanically more complex and thus more costly compared to classical multirotor

platforms. Recently, a similar approach was followed by [Ángel Trujillo et al., 2019], who introduce AeroX, an omnidirectional octacopter for contact-based inspection. Their end effector design minimises the torque caused by contact forces and features wheels on its base to allow moving along a surface while remaining in contact. In [Papachristos et al., 2014b, Papachristos et al., 2014a] the authors show a less complex but highly capable tri-tilt-rotor MAV for surface grinding and obstacle manipulation. The control model consists of two disjoint modes: one for free-flight and another for physical interaction. The authors further discuss different force exertion principles for under- and fully-actuated MAVs.

Employing an underactuated MAVs to perform aerial manipulation typically increases the complexity of the end effector since it has to provide the additional DoFs. To investigate this, many different end effector designs have been proposed over the last years. We can categorise these works by the increasing complexity of the end effector: [Darivianakis et al., 2014] use a fixed end effector on an under-actuated MAV to perform contact-based tasks. The authors use a hybrid MPC method and mode switching between a free-flight and in-contact model. Similarly to our work, they benchmark their approach by performing "aerial-writing". In [Mellinger et al., 2011], the authors use different light-weight, low complexity grippers to perch, pick up, and transport payload. Meanwhile,[Kessens et al., 2016] use a self-sealing suction mechanism to pick up and carry objects. Moving up in terms of complexity,[Kim et al., 2013] suggest mounting a 2-DoF robotic arm on a MAV to allow grasping and transporting of objects. The authors propose an adaptive sliding mode controller for the combined system. In [Orsag et al., 2014] the authors present an aerial manipulator with two robotic 2-DoF arms to open a valve. The MAV and arms are controlled as a coupled system which is modeled as a switched nonlinear system during valve turning. In a more recent work,[Suarez et al., 2018] propose a light-weight, human-sized dual arm system designed to minimise the inertia transferred to the MAV. Each of the two arms add 5 DoFs to the system and the applied arm control law takes into account that low-cost servo motors do not allow torque control but require position commands. Further, a torque estimator is used to predict the torques produced by the servos and inform the MAV control algorithm accordingly. In order to minimise such disturbances coming from the end effector, [Nayak et al., 2018] propose a light-weight design which can produce longitudinal forces for contact-based inspection using a switched system MPC method incorporating the contact dynamics. While attaching a serial robotic arm on an MAV increases the

number of tasks it can perform, they only provide limited precision when using low-cost and light-weight actuators. Some previous efforts [Kamel et al., 2016a, Kamel et al., 2016b] try to mitigate this by mounting a parallel delta arm on an MAV instead.

As far as the control method is concerned, we distinguish three different approaches for controlling the motion of the MAV-arm system. The first one is decoupled control, with a few illustrative examples of an MAV equipped with a serial link arm in [Orsag et al., 2013, Jimenez-Cano et al., 2013] and a parallel arm in [Chermprayong et al., 2019]. In the decoupled control approach the MAV and arm motion are independently controlled, and the effects of the arm on the MAV dynamics are treated as unknown external disturbances. In the second approach, the control remains decoupled but some of the effects introduced by the arm, are compensated in a feed forward way. For example in [Mellinger et al., 2011] a mechanically simple quadrotor-gripper system is used for aerial pick up and transport with the payload mass and the CoM displacement being estimated by a batch least squares estimator running online. Similarly in [Fresk et al., 2017] an EKF is used for online estimating the CoM displacement caused by the arm motion of a hexacopter-arm system. In both the aforementioned approaches the estimates are used in a feed forward fashion by the MAV controller to improve tracking. In alternative work [Tognon et al., 2017] the coupling effect is incorporated into feed forward terms computed exploiting the differential flatness property of the system. The third approach includes forming a unified model and controlling the system in a coupled way which overcomes the limitations of the two methods described above. Examples include the work of [Mersha et al., 2014, Yang et al., 2014] which employ feedback linearisation and LQR controllers respectively and the works of [Garimella and Kobilarov, 2015, Lunni et al., 2017] which –similar to our method– rely on an NMPC for free flight operation.

5.3 Contribution

In this work, we address the problem of precise aerial manipulation and employ our underactuated MAV equipped with a 3DoF delta arm to showcase the capabilities of our approach. In relation to the relevant work presented above, we show the following contributions:

- The formulation of a hybrid model which captures the non-linear dynamics

of the MAV and considers the quasi-static forces, including contact with the environment, introduced by the attached manipulator. The control model is derived using the Newton-Euler equations and can thus be easily extended to any type of multirotor (underactuated/omnidirectional) and attached manipulator.

- Use of this generic model in a NMPC which jointly controls the MAV and arm motion. This natively handles the effects of the arm motion to the MAV dynamics (e.g. displacement of CoM) as well as provides an easy way to control the force (which is a function of the system states) exerted to the environment during contact.

- Experimental evaluation of our method in "aerial-writing" tasks using our custom built system. Our results demonstrate high repeatability and millimeter-level accuracy across multiple trajectories of varying difficulty.

5.4 System overview

The software components of the proposed system are outlined in Figure 5.2. The state estimation block that fuses pose data from the motion capture system and the onboard IMU is identical to the one presented in Section 2.4.1. Same applies to the control allocation block described in Section 4.5.3 that transforms the control inputs into actuator commands. As our focus is on precision rather than failure detection and handling, we have deactivated the fault detection EKF presented in Chapter 4. The NMPC is given full state trajectory commands for the MAV and the end effector corresponding to a given aerial manipulation task. Based on these, it produces the desired MAV body moments, collective thrust, and end effector position. The desired end effector position is then transformed into joint angle commands by solving the inverse kinematics problem for the delta arm described in Section 2.3.1. All algorithms run onboard at a rate of 100 Hz.

In the free flight scenarios presented in previous chapters, equations of motion were expressed using the MAV Body fixed frame $\underrightarrow{\mathcal{F}}_B$ and the World fixed frame $\underrightarrow{\mathcal{F}}_W$. For the setup presented in this Chapter we further introduce: (i) an arm frame $\underrightarrow{\mathcal{F}}_A$, an end-effector frame $\underrightarrow{\mathcal{F}}_E$ and the contact surface frame $\underrightarrow{\mathcal{F}}_T$. An overview of the different coordinate frames is given in Figure 5.3.

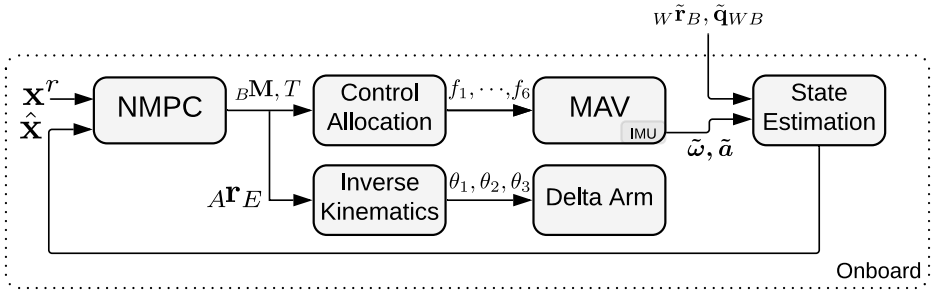


Figure 5.2: An overview of the software running onboard our MAV in an aerial manipulation task. The NMPC jointly controls the MAV and arm motion. The desired moments and thrust are transformed into motor commands using the control allocation described in Section 4.5.3 while the delta arm joint angles are computed using the inverse kinematics described in Section 2.3.2.

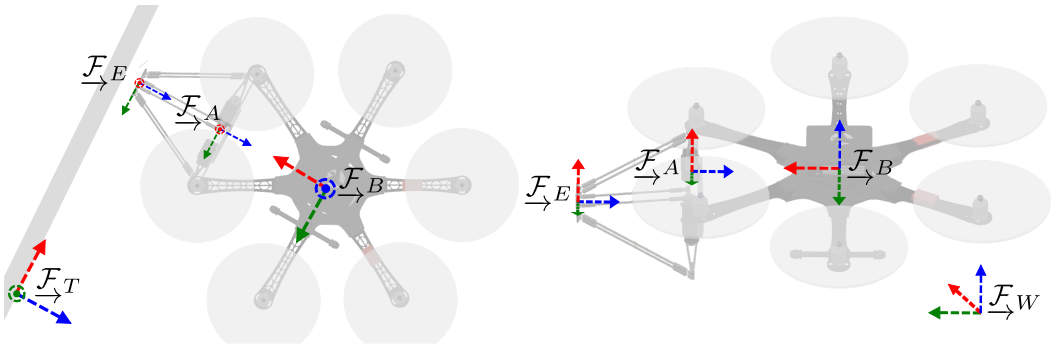


Figure 5.3: The different coordinate frames used in the aerial manipulation task. Specifically $\underrightarrow{\mathcal{F}}_W$, $\underrightarrow{\mathcal{F}}_B$, $\underrightarrow{\mathcal{F}}_A$, $\underrightarrow{\mathcal{F}}_E$, and $\underrightarrow{\mathcal{F}}_T$ stand for the World, MAV body, arm, end effector, and contact frame, respectively.

5.5 Hybrid modelling

The derivation of the hybrid model used for control largely follows the one presented in Section 4.5.2. It is appropriately augmented with the quasi-static forces introduced by the arm dynamics and its interaction with the environment. Overall, the

combined dynamics take the following form:

$$ {}_W\dot{\mathbf{r}}_B = {}_W\mathbf{v}_B\,, \tag{5.1a} $$

$$ \dot{\mathbf{q}}_{WB} = \frac{1}{2}\mathbf{\Omega}({}_B\boldsymbol{\omega}\,)\mathbf{q}_{WB}\,, \tag{5.1b} $$

$$ {}_W\dot{\mathbf{v}}_B = \frac{1}{m_c}\mathbf{C}_{WB}\,({}_B\mathbf{F}_r + {}_B\mathbf{F}_e) + {}_W\mathbf{g}, \tag{5.1c} $$

$$ {}_B\dot{\boldsymbol{\omega}} = \mathbf{J_c}^{-1}({}_B\mathbf{M}_r + {}_B\mathbf{M}_e - {}_B\boldsymbol{\omega} \times \mathbf{J_c}\,{}_B\boldsymbol{\omega}\,), \tag{5.1d} $$

$$ \mathbf{\Omega}({}_B\boldsymbol{\omega}\,) = \begin{bmatrix} {}_B\boldsymbol{\omega}^{\times} & {}_B\boldsymbol{\omega} \\ -{}_B\boldsymbol{\omega}^{\top} & 0 \end{bmatrix} \tag{5.1e} $$

where m_c, $\mathbf{J_c}$ are the combined MAV-arm mass and inertia tensors, respectively. Regarding the forces and moments ${}_B\mathbf{F}_i, {}_B\mathbf{M}_i$, we use the subscript $i \in \{r, e\}$ to distinguish the ones generated by the MAV motors r from the ones caused by the end effector movement e and its potential contact with the environment. In our system, the MAV motor-generated forces and moments are given by:

$$ {}_B\mathbf{F}_r := \begin{bmatrix} 0, 0, T \end{bmatrix}^{\top}\,,\; T = \sum_{i=1}^{6} f_i, \tag{5.2a} $$

$$ {}_B\mathbf{M}_r := \sum_{i=1}^{6} \Big(f_i {}_B\mathbf{r}_i \times {}_B\mathbf{e}_z + (-1)^{i+1} k_m f_i {}_B\mathbf{e}_z \Big), \tag{5.2b} $$

with $f_i \in \mathbb{R}$ the thrust produced by the i^{th} motor, ${}_B\mathbf{r}_i$ its position with respect to the MAV body frame, k_M the known thrust to moment constant (as defined in the motor model (2.8)) and ${}_B\mathbf{e}_z = [0, 0, 1]^{\top}$. As shown in Section 4.5.3, Equation (5.2) can be summarised as $\begin{bmatrix} {}_B\mathbf{M}_r, & T \end{bmatrix}^{\top} = \mathbf{A} \begin{bmatrix} f_1 & f_2 & \dots & f_6 \end{bmatrix}^{\top}$ with $\mathbf{A} \in \mathbb{R}^{4\times 6}$ the constant allocation matrix obtained from (4.8) when all the available motors can produce positive only thrust. The forces and moments ${}_B\mathbf{F}_e$, ${}_B\mathbf{M}_e$ introduced by the end-effector are given by:

$$ {}_B\mathbf{F}_e := \mathbf{C}_{BE}\,{}_E\mathbf{F}_c, \tag{5.3a} $$

$$ {}_B\mathbf{M}_e := {}_B\mathbf{r}_E \times {}_B\mathbf{F}_e + ({}_B\mathbf{r}_E - {}_B\mathbf{r}_{E_0}) \times (\mathbf{C}_{BW}\, m_e\, {}_W\mathbf{g}), \tag{5.3b} $$

where ${}_E\mathbf{F}_c$ is the contact force acting on the end effector expressed in its frame $\underrightarrow{\mathcal{F}}_E$ and ${}_B\mathbf{r}_{E_0} \in \mathbb{R}^3$ the nominal end effector position which results in no CoM displacement. The two terms in (5.3b) represent the moments due to contact and due to the displacement of the CoM respectively. The combined mass $m_c = m + m_e$ is the sum of the MAV and end effector, respectively, while the combined rotational inertia can be computed as $\mathbf{J_c} = \mathbf{J} + m_e \text{diag}({}_B\mathbf{r}_E - {}_B\mathbf{r}_{E_0})^2$ with m_e being the mass

of the end effector, $\mathbf{J} = \text{diag}(J_x, J_y, J_z)$ the inertia tensor of the MAV (including the arm mass in nominal position) and $\text{diag}(\cdot)$ the corresponding diagonal matrix.

Regarding the contact force, we assume that this can be approximated with a linear spring model as:

$$ {}_E\mathbf{F}_c = \mathbf{C}_{ET}\,(k_{sT}\mathbf{r}_{E_z}), \tag{5.4} $$

where k_s is a known spring coefficient and ${}_T\mathbf{r}_{E_z}$ is the normal component of the contact surface penetration. This way, the controller can anticipate contact before it even happens and there is no need for a switching mode controller (one for free flight and another one for contact dynamics).

5.6 NMPC for aerial manipulation

For the control formulation we define the following control state and input:

$$ \mathbf{x} := \left[{}_W\mathbf{r}_B^\top, {}_W\mathbf{v}_B^\top, \mathbf{q}_{WB}^\top, {}_B\boldsymbol{\omega}^\top \right]^\top \in \mathbb{R}^6 \times S^3 \times \mathbb{R}^3, \tag{5.5a} $$

$$ \mathbf{u} := \left[{}_B\mathbf{M}_r^\top, T, {}_A\mathbf{r}_E^\top \right]^\top \in \mathbb{R}^7. \tag{5.5b} $$

The definition of the control state coincides with one in Section 4.5.1, whereas the control input is augmented with the position of the end-effector ${}_A\mathbf{r}_E$. Note that we use ${}_B\mathbf{r}_E$ for the formulation of the control model, while ${}_A\mathbf{r}_E$ is used in the control input. We use the constant and known homogeneous transformation $\boldsymbol{T}_{BA}$ to change the coordinate representation of these position vectors.

We use the following error functions for the position of the MAV, the position of the end effector, the MAV linear and angular velocity, the orientation, the contact force and the control input, respectively:

$$ \mathbf{e}_{rB} = {}_W\mathbf{r}_B - {}_W\mathbf{r}_B^r, \tag{5.6a} $$

$$ \mathbf{e}_{rE} = {}_W\mathbf{r}_E - {}_W\mathbf{r}_E^r, \tag{5.6b} $$

$$ \mathbf{e}_v = {}_W\mathbf{v}_B - {}_W\mathbf{v}_B^r, \tag{5.6c} $$

$$ \mathbf{e}_\omega = {}_B\boldsymbol{\omega} - \mathbf{C}_{BB^r}\,{}_{B^r}\boldsymbol{\omega}^r, \tag{5.6d} $$

$$ \mathbf{e}_q = [\mathbf{q}_{WB}^{-1} \otimes \mathbf{q}_{WB}^r]_{1:3}, \tag{5.6e} $$

$$ e_f = f_c - f_c^r, \tag{5.6f} $$

$$ \mathbf{e}_u = \mathbf{u} - \mathbf{u}^r, \tag{5.6g} $$

with $f_c := {}_E\mathbf{F}_{c_z}$ and the superscript r used to denote the time-varying reference quantities. The optimal input sequence $\mathbf{u}^*$ is obtained by the online solution of the following constrained optimisation problem:

$$\mathbf{u}^* = \underset{\mathbf{u}_0,\dots,\mathbf{u}_{N_f}}{\operatorname{argmin}} \Big\{ \Phi(\mathbf{x}_{N_f}, \mathbf{x}^r_{N_f}) + \sum_{n=0}^{N_f-1} L(\mathbf{x}_n, \mathbf{x}^r_n, \mathbf{u}_n) \Big\}, \tag{5.7a}$$

$$\text{s.t. : } \mathbf{x}_{n+1} = \mathbf{f}_d(\mathbf{x}_n, \mathbf{u}_n), \tag{5.7b}$$

$$\mathbf{x}_0 = \hat{\mathbf{x}}, \tag{5.7c}$$

$$u_{\text{lb}} \leq u_i \leq u_{\text{ub}}, \quad i = 1, \dots, 7, \tag{5.7d}$$

where N_f is the discrete horizon length, $\mathbf{f}_d$ is the discrete version of the dynamics given in Equations 5.1-5.3, $\hat{\mathbf{x}}$ is the current state estimate and $u_{\text{lb}}, u_{\text{ub}}$ are the appropriate lower and upper bounds of the control input defined in (5.5). For the intermediate L and final terms Φ we use quadratic costs of the form $\mathbf{e}_i^\top \mathbf{Q}_i \mathbf{e}_i$ $\forall \mathbf{e}_i \in \{\mathbf{e}_{rB}, \mathbf{e}_{rE}, \mathbf{e}_v, \mathbf{e}_\omega, \mathbf{e}_q, e_f, \mathbf{e}_u\}$ as defined in (5.6) where the gain matrices $\mathbf{Q}_i \succcurlyeq 0$ were experimentally tuned.

As in Chapter 4, the optimal control problem is implemented using a modified version of the CT toolbox [Giftthaler et al., 2018] with a 10 ms discretisation step and a 2 s constant prediction horizon. We use a Runge-Kutta 4 integration scheme followed by a re-normalisation for the quaternion. As common in receding horizon control, the first input $\mathbf{u}_0^*$ is applied to the system and the whole process is repeated once a new state estimate $\hat{\mathbf{x}}$ becomes available.

Regarding the computation of the MAV motor commands, we use the control allocation algorithm described in Section 4.5.3 with the bidirectional mode disabled. Briefly, the motor commands $\mathbf{f} = \begin{bmatrix} f_1 & f_2 & \dots & f_6 \end{bmatrix}^\top$ for the MAV are obtained by solving the following QP:

$$\mathbf{f}^* = \underset{\mathbf{f}}{\operatorname{argmin}} \left(\left\| \mathbf{A}\mathbf{f} - \mathbf{u}^*_{0_{1:4}} \right\|^2_{\mathbf{W}} + \lambda \left\| \mathbf{f} \right\|_2^2 \right), \tag{5.8a}$$

$$\text{s.t. : } f_{\min} \leq f_i \leq f_{\max}, \quad i = 1, \dots, 6, \tag{5.8b}$$

where $f_{\min}, f_{\max} \in \mathbb{R}$ are the minimum and the maximum attainable thrusts, $\mathbf{W} \in \mathbb{R}^{4\times4}$ is a gain matrix and $\lambda = 10^{-7}$ is the regularisation parameter.

The end effector position commands $\mathbf{u}^*_{0_{5:7}}$ are mapped into servo angle commands $\theta_1, \theta_2, \theta_3$ by solving the inverse kinematics problem for the delta arm explained in Section 2.3.2. In the case of an infeasible (e.g. outside the arm's workspace) or

unsafe end effector position command (e.g. one that results collision between the MAV propellers and the arm's links), the position command is reprojected onto the boundary of the feasible and safe to operate workspace. In practice, this was rarely the case as the MAV and end effector reference trajectories are designed (see Section 5.8) so that the end effector operates close to its nominal position. In this way the usable workspace is maximised while the effect of the CoM displacement (which is captured by our control model) is minimised.

Our C++ implementation of the above, requires 6.7 ms with a standard deviation of 0.57 ms per iteration. On average, 98% of the computation time is spent on the optimisation problems defined in (5.7) and (5.8).

We would like to highlight that our method is generic enough and can be easily applied on similar MAV-arm systems. In the case of an underactuated MAV (similar to this work) the only part that has to be changed is the control allocation method whcih depends on the vehicle type and its motor layout. In the case of an overactuated MAV, the Equation 5.2a as well as the control input (5.5) should be further augmented with the lateral forces. Regarding the arm, our method is arm type agnostic as it only considers the end effector position. Additionally, the model can easily be extended to capture aerodynamic friction, gyroscopic moments, handle multiple contact points or use more sophisticated contact models (e.g. ones that include combination of linear springs and dampers). Similarly, the 'aerial-writing' task which we use for the experimental evaluation of our framework, is just an example application that requires precision. We believe that our algorithms are adaptable to other tasks such as inspection through contact.

5.7 Extrinsics calibration

Apart from the various MAV related parameters (e.g. mass, inertia, motor coefficients), the control model depends on the arm $\underrightarrow{\mathcal{F}}_A$ to body $\underrightarrow{\mathcal{F}}_B$ frame transformation $\boldsymbol{T}_{BA}$ as well as the transformation $\boldsymbol{T}_{WT}$ between the contact $\underrightarrow{\mathcal{F}}_T$ and world frame $\underrightarrow{\mathcal{F}}_W$. We experimentally observed that the system performance and mainly the end-effector tracking accuracy are significantly affected by wrong estimates of these transformations. We thus adopt an optimisation based calibration process.

5.7.1 Arm to body frame transformation

Theoretically, getting an accurate estimate of the constant $\boldsymbol{T}_{BA}$ can be done through the detailed Computer Aided Design (CAD) of the system. In practice, precise manufacturing of the aerial system with respect to the reference CAD model is challenging. An alternative approach would be to separately track the $\underrightarrow{\mathcal{F}}_A$ and $\underrightarrow{\mathcal{F}}_B$ pose using the motion capture system and compute their relative transform. Unfortunately, the sideways mounted arm resulted in poor marker visibility for $\underrightarrow{\mathcal{F}}_A$ and thus inaccurate results. To overcome this we follow an optimisation based approach where the calibration process consists of the following steps:

1. We command the NMPC a constant position reference ${}_W\mathbf{r}_E^r$ for the end effector.
2. With the MAV motors disabled, we manually move the MAV in various different positions and orientations. As the NMPC tries to keep the end effector error as small as possible, the arm is constantly moving in order to counteract the MAV movement.
3. We record the MAV position and orientation ${}_W\mathbf{r}_B$, $\mathbf{q}_{WB}$ as well as the end effector position ${}_W\mathbf{r}_E$ using the motion capture system.
4. We compute the end effector position ${}_A\mathbf{r}_E$ expressed in $\underrightarrow{\mathcal{F}}_A$ using the encoder measurements from the delta arm servos and the forward kinematics described in Section 2.3.1.

Once the data collection is completed, we offline solve the following optimisation:

$$J = \underset{\mathbf{q}_{BA}\,,{}_B\mathbf{r}_A}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{T}_{WB_i}\, \boldsymbol{T}_{BA}\, {}_A\boldsymbol{r}_{E_i} - {}_W\boldsymbol{r}_{E_i} \right\|^2 \right), \tag{5.9}$$

where the subscript i denotes the index of the N discrete observations. Since the motion capture and encoder measurements are collected at different time instants, we obtained time synchronised measurements ${}_W\mathbf{r}_{B_i}, \mathbf{q}_{WB_i}, {}_W\mathbf{r}_{E_i}, {}_A\mathbf{r}_{E_i}$ by evaluating (using linear interpolation) the encoder data at the timestamps of the motion capture data.

A side effect of estimating the $\boldsymbol{T}_{BA}$ using the optimisation problem in (5.9) is that it ensures that tracking accuracy of the end effector position commands is maximised. This allows small position and orientation offsets originating from imperfect

forward kinematics to be –in average– cancelled out by appropriate estimation of the position and orientation parts of $\boldsymbol{T}_{BA}$. On average the optimum value for J was in the order of 4 mm which should be considered as the maximum position accuracy of the end-effector.

5.7.2 Contact to world frame transformation

For the $\boldsymbol{T}_{WT}$ calibration we start with an initial estimate $\boldsymbol{T}_{WT_0}$ and aim to estimate a transformation refinement $\boldsymbol{T}_{T_0T}$ given by:

$$\boldsymbol{T}_{T_0T} := \begin{bmatrix} & & 0 \\ \mathbf{C}_x(\phi)\mathbf{C}_y(\theta) & & 0 \\ & & z \\ \mathbf{0}_{1\times 3} & & 1 \end{bmatrix}, \tag{5.10}$$

with $\mathbf{C}_x$, $\mathbf{C}_y$ the rotation matrices around the x and y axes respectively and θ, ϕ, z the unknown rotation angles and offset we want to estimate. We do so by recording multiple position measurements ${}_W\boldsymbol{r}_T$ of points which lie on the contact surface and by solving the following optimisation problem:

$$J = \underset{\theta,\phi,z}{\operatorname{argmin}} \Big(\frac{1}{N} \sum_{i=1}^{N} \Big\| \big((\boldsymbol{T}_{WT_0}\, \boldsymbol{T}_{T_0T}\,)^{-1}\, {}_W\boldsymbol{r}_{T_i} \big)_{3:3} \Big\|^2 \Big), \tag{5.11}$$

with the subscript i denoting the index of the N discrete observations. Solution of (5.11) yields the best estimate $\boldsymbol{T}_{WT} := \boldsymbol{T}_{WT_0}\, \boldsymbol{T}_{T_0T}$ which ensures that the distance between the observations ${}_W\boldsymbol{r}_{T_i}$ and the contact surface along its normal direction is minimised.

The definition of $\boldsymbol{T}_{T_0T}$ in (5.10) ensures that the unobservable rotation of $\boldsymbol{T}_{WT}$ around the normal direction is identical to the initial estimate $\boldsymbol{T}_{WT_0}$. For the computation of the initial estimate $\boldsymbol{T}_{WT_0}$, we average the pose measurements provided by the motion capture system by tracking an object attached on the contact surface. For the pose averaging we use the method described in [Markley et al., 2007] where the average orientation quaternion is computed by minimising the squared Frobenius norm of the difference between rotation matrices.

Compared to the calibration process of $\boldsymbol{T}_{BA}$ which is only done once, calibration of $\boldsymbol{T}_{WT}$ is required after every motion capture system calibration (which includes realignment of the $\underrightarrow{\mathcal{F}}_W$). In all the tested calibration datasets, the optimal value for J was less than 0.5 mm.

5.8 Reference computation

We use a trajectory generator to map arbitrary sets of characters to end effector trajectories. We use a constant acceleration motion model to generate trajectories with a smooth velocity profile. This is of special importance when the reference path contains sharp edges. Through our software we can adjust the velocity and acceleration profile by changing the maximum $\|{}_W\mathbf{v}_E^r\|$ and $\|{}_W\mathbf{a}_E^r\|$. Once the trajectory for the end effector has been computed, we proceed with the computation of the reference position ${}_W\mathbf{r}_B^r$ and velocity ${}_W\mathbf{v}_B^r$ for the MAV as follows:

$$ {}_W\mathbf{r}_B^r = {}_W\mathbf{r}_E^r - \mathbf{C}_{WB}\,{}_B\mathbf{r}_{E_0}, \tag{5.12a} $$

$$ {}_W\mathbf{v}_B^r = {}_W\mathbf{v}_E^r - \mathbf{C}_{WB}\,{}_B\boldsymbol{\omega}^r \times {}_B\mathbf{r}_{E_0}. \tag{5.12b} $$

The reference MAV orientation $\mathbf{q}_{WB}^r$ is chosen such that the end effector position is always perpendicular to the contact surface, assuming perfect position tracking. Computation of the MAV reference position using the above equations ensures that the end effector operates close to its nominal position ${}_B\mathbf{r}_{E_0}$. Given a dynamically feasible trajectory for the end effector, the desired acceleration, jerk and snap for the MAV can be computed by further differentiating (5.12a) and using $\dot{\mathbf{C}} = \mathbf{C}\,[\boldsymbol{\omega}]^\times$ for the rotation matrix derivative.

In our framework, each trajectory is accompanied by an appropriate flag which disables or enables the position tracking for the end effector. This is achieved by setting the appropriate gains to zero. In that case the NMPC may decide to move the arm to assist the reference tracking of the MAV due to the CoM displacement. This – depending on the application – potentially unwanted behaviour can be avoided by further penalising (i.e. by increasing the input gains) the arm displacement from its nominal position. However, it is an interesting capability enabled by our hybrid modeling.

5.9 Experimental results

The experiments presented in this section were performed using the custom built hexacopter equipped with the sideways mounted delta arm manipulator described in Sections 2.2 and 2.3 respectively. In order to cope with the increased payload we used the propulsion system #1 described in Section 2.2.3. The end effector of the delta arm holds the pen which is mounted on a spring to provide additional

compliance. We set the coefficient of the contact model in Equation (5.4) to match the used spring. The applied force is measured by a SingleTact[1] force sensor, which provides measurements at a resolution of 9×10^{-3} N, mounted at the end of the spring. We estimate the spring coefficient k_s by measuring the applied force for known tip displacements. We measured the inertia of the MAV $\mathbf{J}$ by checking its angular response to constant input torque while it is hanging to freely rotate. A table with all the numeric values of the system parameters, used in the control model, is given in Table 5.1 while a photo showing the platform and its different components is shown in Figure 5.4.

Description	**Symbol**	**Value**	**Unit**
MAV mass	m	2.6	kg
End effector mass	m_e	0.058	kg
MAV inertia in x axis	J_x	0.042	kg/m^2
MAV inertia in y axis	J_y	0.054	kg/m^2
MAV inertia in z axis	J_z	0.110	kg/m^2
Moment coefficient	k_M	1.5864×10^{-2}	Nm/N
Spring coefficient	k_s	42.95	N/m

Table 5.1: Numeric values of control model parameters.

Full state estimates are obtained by the EKF described in Section 2.4.1 which fuses the onboard IMU measurements and the pose estimates provided by a Vicon motion capture system. Each experiment consisted of the following different trajectory stages: (i) approach, (ii) write, and (iii) return home. The end effector tracking was enabled, using the appropriate flags as mentioned in Section 5.8, for the writing trajectory and disabled for the rest. Our analysis mainly focuses on the trajectory writing which includes contact whereas for the other two parts (approach/return) the MAV performs simple position tracking. We evaluate the accuracy of our system by comparing the reference trajectories to those estimated by the Vicon motion capture system. In addition, we use a vision-based error as a performance metric. This is because we observed inaccuracies in the Vicon measurements stemming from either bad calibration, poor object visibility, or marker reflections on the whiteboard surface. The visual error is computed by running a 2D Iterative Closest Point (ICP) method [Besl and McKay, 1992] on a filtered and rectified photo of the final writing and a rendering of the planned path. After registration of the two point sets, we

[1] See https://www.singletact.com/.

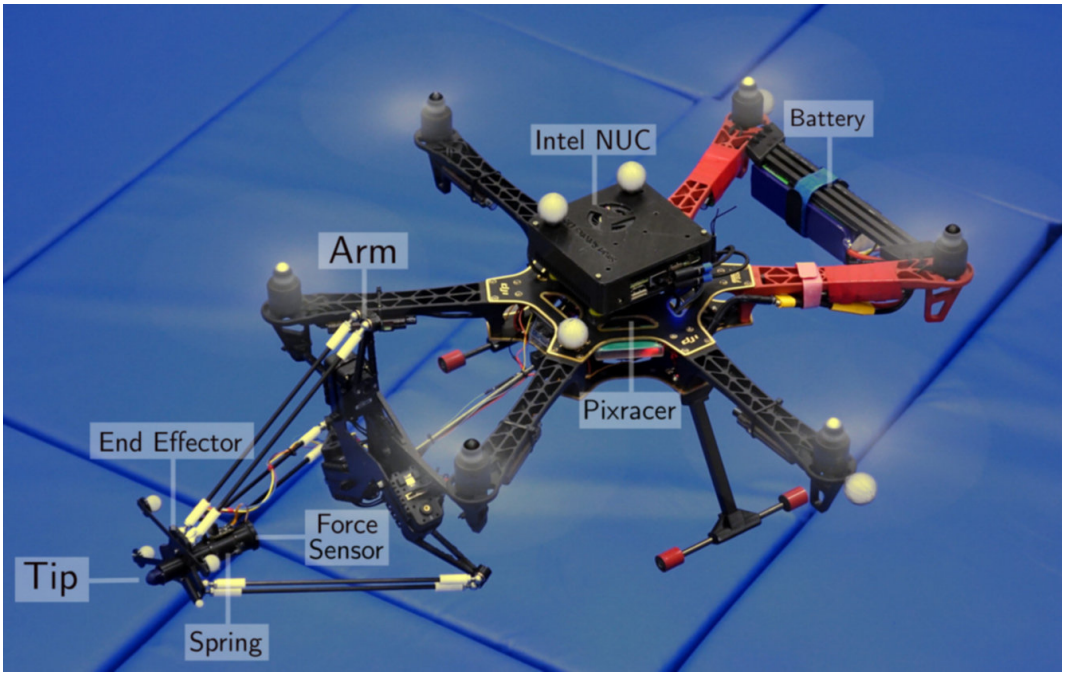


Figure 5.4: The aerial manipulation platform used in the 'aerial writing' experiments in Section 5.9 with labels for its individual components.

use the nearest neighbour distance to evaluate the accuracy.

In the following we show four experiments: in Section 5.9.1 we present detailed and repeatable results for two different trajectories, namely `RSS` and $E = mc^2$. We then show consistent tracking performance across varying MAV velocities and text sizes in Sections 5.9.2 and 5.9.3, respectively.

5.9.1 Trajectory tracking

Figure 5.6 shows the tracking of the `RSS` trajectory visualised in the contact frame $\underrightarrow{\mathcal{F}}_T$ for the end effector and the MAV. The maximum reference velocity was set to 7.5 cm/s and the maximum acceleration to 2.5 cm/s^2. The trajectory consists of four contact segments with a combined duration of 65 s. Based on the Vicon estimates the tracking error of the end effector remains in the $[-10, 10]$ mm range during the contact segments while the MAV position error is within the $[-40, 40]$ mm range. This highlights the efficacy of using a manipulator with faster dynamics than the MAV's for precision tasks such as 'aerial-writing'.

Similarly to the above, Figure 5.8 shows the trajectory tracking for the more chal-

lenging $E = mc^2$ experiment which contains ten contact segments with a combined duration of 63 s. Tracking accuracy is similar as before with the end effector and MAV tracking error in the $[-10, 10]$ and $[-50, 50]$ mm range. The accuracy can be visually verified since the overlapping segments of the 'R' and 'm' coincide almost perfectly. Additionally, the consistent approaching and retracting from the contact surface leads to identical starting points of individual letter segments, e.g. the three horizontal lines of the letter 'E'. In both cases, the maximum error based on the visual error analysis is 10 mm mostly originating from temporary loss of contact. Possible reasons for this are bad estimation of the orientation part of the contact frame transformation $\boldsymbol{T}_{WT}$, the assumption of a perfectly flat contact surface being wrong and most importantly the finite accuracy of the delta arm. The imperfect tracking along the contact frame normal direction (shown in blue in Figures 5.6, 5.8) is also reflected in the reference force tracking.

In order to prove the repeatability of our approach, we conducted each experiment thrice. We give the relevant tracking statistics for the MAV and arm separately in Figure 5.9, in which the textured box plots correspond to MAV data and the plain ones to that of the end effector. The median and upper values for the end effector are significantly lower than the ones for the MAV further showing the need of an aerial manipulator for precise tasks including contact. The MAV tracking accuracy along the z axis was the lowest amongst all axes, as this was most affected by the interaction forces and unmodeled torque disturbances due to the movement of the servos.

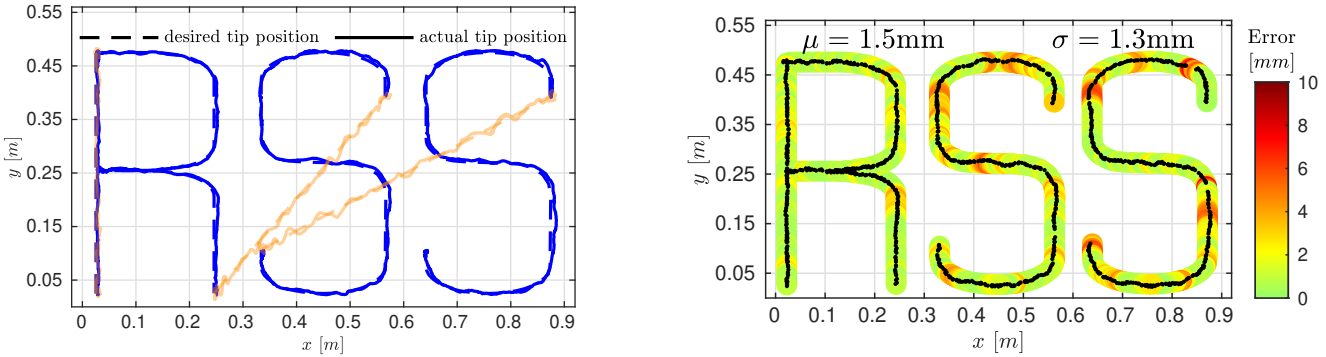


Figure 5.5: Reference and actual tip position (left) as estimated by Vicon. Blue corresponds to contact segments while orange refers to free flight. Visual error (right) between reference and actual tip position. The maximum estimated error is lower than 10 mm and is located at discontinuous segments as expected.

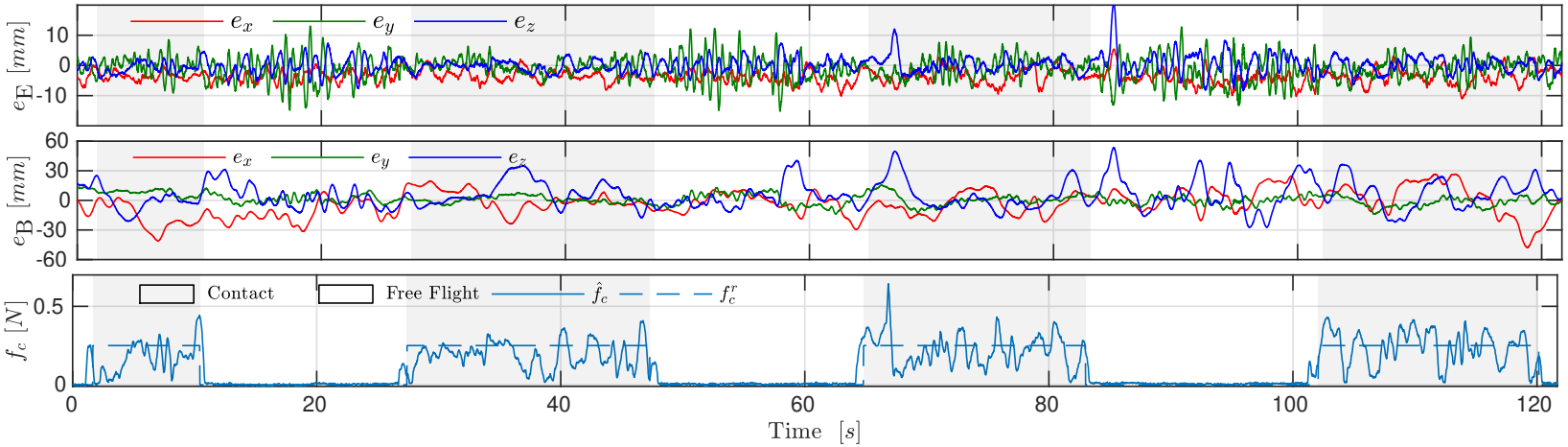


Figure 5.6: Reference tracking error of the tip position (top), MAV (middle), and measured contact force (bottom). The tracking error is plotted in the contact frame $\mathcal{F}_T$. The tracking accuracy of the end effector is significantly greater than that of the MAV, given that they remain in the $[-10, 10]$ mm and $[-40, 40]$ mm ranges, respectively.

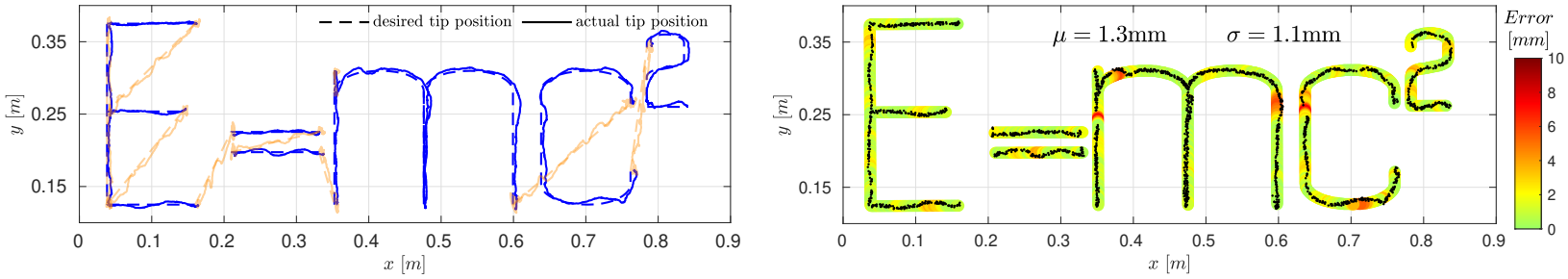


Figure 5.7: Reference and actual tip position (left) as estimated by Vicon. Blue corresponds to contact segments while orange refers to free flight. Visual error (right) between reference and actual tip position. Similarly as in the RSS experiment shown in Figure 5.6, maximum error does not exceed 1cm.

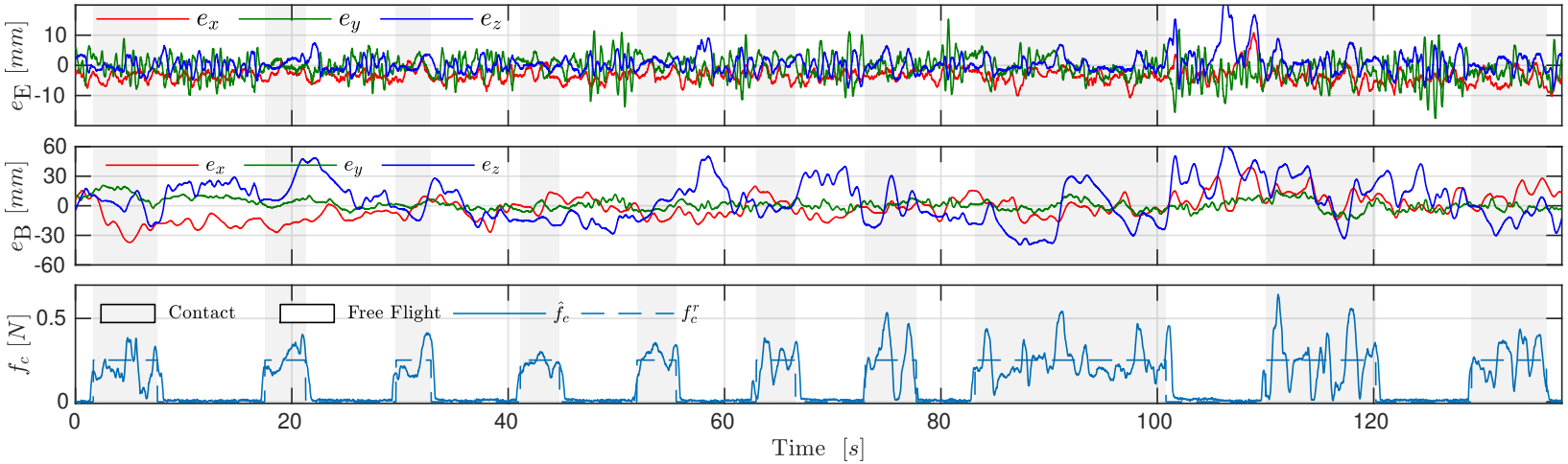


Figure 5.8: Reference tracking error of the tip position (top), MAV (middle) and measured contact force (bottom). The tracking accuracy of the end effector is significantly greater than that of the MAV, given that they remain in the $[-10, 10]$ mm and $[-50, 50]$ mm ranges, respectively

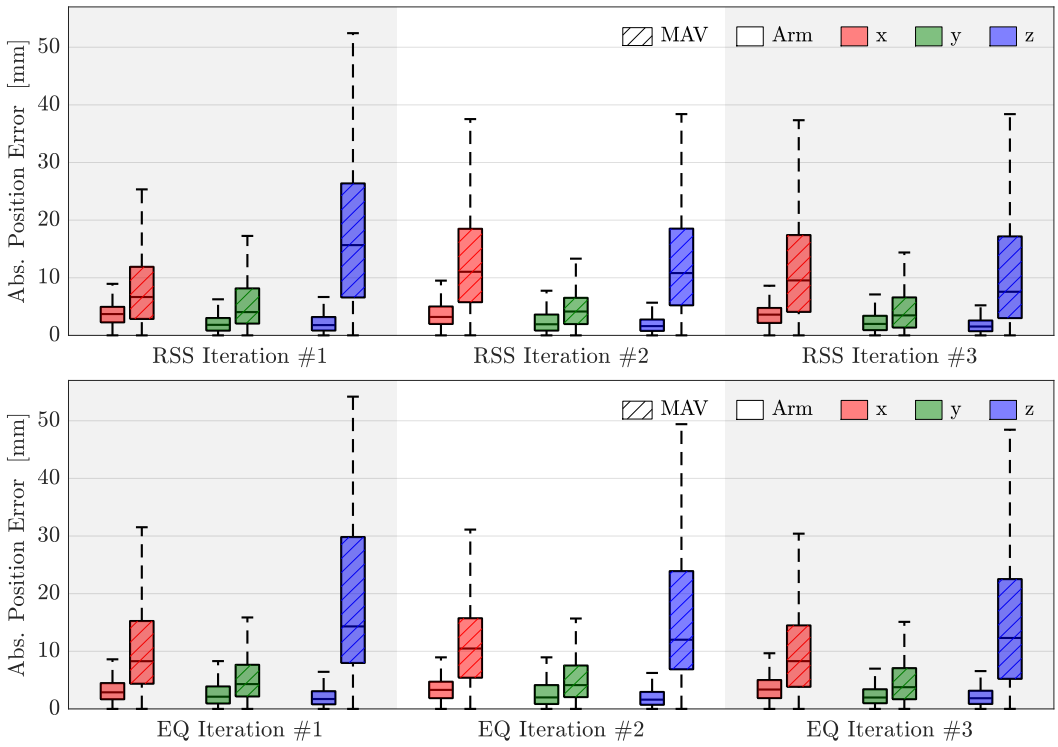


Figure 5.9: MAV and end effector box plots of the contact segments for 3 iterations of the `RSS` trajectory experiment (top) and the more challenging $E = mc^2$ trajectory experiment (bottom).

5.9.2 Velocity Sweep

The aim of the next experiments is to demonstrate the effects of the input velocity and acceleration on the writing accuracy. We performed five iterations of the same `Hello` trajectory experiment with different velocity and acceleration profiles. These correspond to maximum velocity $\|{}_W\mathbf{v}_E^r\| \in \{7.5, 12.5, 17.5, 22.5, 27.5\}$ cm/s and maximum acceleration $\|{}_W\mathbf{a}_E^r\| \in \{3.75, 6.25, 8.75, 11.25, 13.75\}$ cm/s^2.

Figure 5.10 shows the box plots for the MAV and end effector tracking accuracy based on the Vicon measurements. The plots show that consistent tracking results are obtained in all the different velocity and acceleration settings tested. The numeric values of the tracking error are similar to the ones previously presented with the end effector achieving sub centimetre accuracy (per axis) while the MAV error is consistently less than 50mm. However, by observing the visual error, we verify that as the reference velocity increases, the system struggles more with the trajectory segments containing curvature e.g. 'e' and 'o'. In contrast, the performance on the

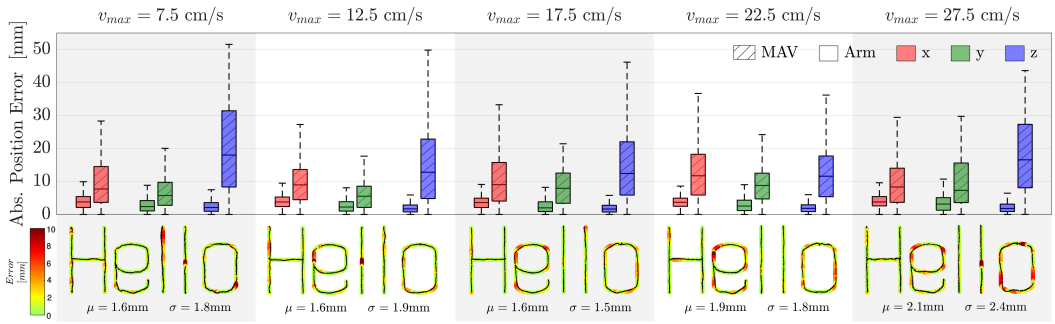


Figure 5.10: MAV and end effector box plots(top) and visual errors(bottom) for 5 iterations of the `Hello` trajectory. Different iterations correspond to different velocity and acceleration profiles.

straight line segments remains similar.

We believe that the tracking error of the MAV can be further reduced if the NMPC was given dynamically feasible trajectories not only for position and velocity but also acceleration, jerk, and snap. Regarding the end effector tracking error, we generally expect this to increase for reference velocities beyond the ones tested here. This is because our control model assumes that the position of the end effector can be controlled infinitely fast which is not the case for a real system.

5.9.3 Trajectory size sweep

In Figure 5.11 we show the visual error of our system for the same trajectory in four different text sizes ranging from 10 to 40 cm. The consistent accuracy observed shows that the system can handle the fast direction changes imposed by the small scale.

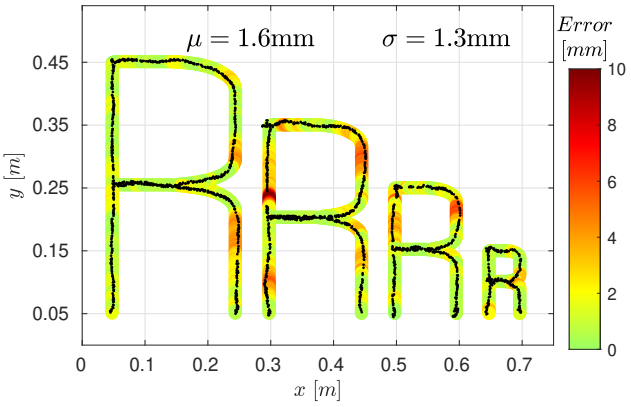


Figure 5.11: Visual error plot showing consistent results for varying text sizes

5.10 Discussion

Overall, our system achieves accurate and consistent results over a series of different trajectories. The end effector tracking error consistently remains in the $[-10, 10]$ mm range for trajectories with maximum velocities ranging from 7.5 to 27.5 cm/s, maximum acceleration from 3.75 to 13.75. The tracking error of the end effector is significantly lower than the one for the MAV highlighting the accuracy boost due to the utilisation of the arm. We would like to mention that our system was built using relatively cheap off-the-shelf components and 3D printed parts. This leads to errors in the manufacturing of the aerial system with respect to the reference model e.g. errors in the true inverse kinematics of the arm due to non-identical dimensions of its links.

Another important issue that we faced during our experiments, was the reliance on the motion capture system for localisation. Apart from issues related to WiFi delays which resulted in temporary loss of tracking, we faced issues with poor object visibility resulting in unreliable estimates of both the static objects, such as the contact frame, and moving ones such as the MAV. In fact, during our data analysis we realised that there are segments where Vicon returned mechanically impossible configurations for our system e.g. end effector position below the surface of the contact frame. Despite these problems which further propagate into tracking errors, our system was able to handle well multiple transitions to contact during the same experiment.

We experimentally verified that for contact tasks, where the main objective is accuracy instead of speed, using a planner respecting full state dynamic feasibility is not an absolute necessity. Despite our simplified motion planner, our system achieves sub-centimeter accuracy. However, we argue that for more aggressive maneuvers, a full state dynamically feasible planer would be required.

Regarding future work, we consider that online estimation of the contact frame transformation $\boldsymbol{T}_{WT}$ would be beneficial as our system was very sensitive to its wrong estimates. This would be very useful in a real life scenario (e.g. inspection through contact) where the contact surface might not be reachable and thus its position and orientation not directly measurable. Furthermore, it will provide a natural way of handling contact with curved or generally non flat surfaces. Another possible extension is that of using visual feedback to close the (currently open) loop

of the aerial manipulation task. This requires the formulation of a visual-based error metric based on the reference aerial manipulation task and the achieved one (e.g. reference end effector trajectory and observed one). Closing this feedback loop would result a major improvement for errors stemming from bad calibration (e.g. $\boldsymbol{T}_{WT}$, $\boldsymbol{T}_{BA}$), state estimate delays or inaccuracies of the arm. Finally, using a better model for the combined MAV-arm system, would result better tracking performance. As a first step, we can approximate the arm response using a first order model (the current method further assumes an instantaneous end effector position response) or ultimately, formulate the full multi-body dynamics model. The last would be beneficial for agile combined maneuvers not tested in this work.

Chapter 6

Conclusions

Contents

6.1 Summary of results

In this section we present a summary of the contributions and results which have been presented in detail in the previous chapters.

In Chapter 3 a linear MPC incorporating soft state constraints as well as a generic framework for vision based tracking were presented. A cascaded control approach was adopted with the position and attitude dynamics being independently controlled. The position control problem was formulated as a canonical form QP enabling online computation using any generic QP solver. The experiments conducted for the MBZIRC competition showed that the designed controller can be used for tracking and landing on a target moving with velocities not exceeding 5.0 m/s while the experiments related to the AABM project revealed that the system is extremely accurate (maximum per-axis error of 1.5 cm) for slow reference trajectories. Main benefit of this control approach is its easy deployment on any platform which is equipped with a closed loop attitude controller. This can be further justified since the experimental results presented in Chapter 3 were obtained using three different platforms. These two advantages (accuracy for slow maneuvers, ease of use) made this controller our preferred choice for the majority of our experiments and live demos which did not require agile maneuvers.

The linear MPC was extended to an NMPC in Chapter 4. By using a quaternion based parameterisation and the non linear model of the system we overcame the main limitations of the initial approach (i.e. inaccurate system model when operating away from the linearisation point, gimbal lock due to the use of Euler angles). Through a simple position-yaw step maneuver we showed how the controller can use the non-linear dynamics –in an unconventional way– so as to perform the reference task as fast as possible. In order to enable robust operation despite the loss of a single actuator, we examined the controllability of our MAV and proposed a stabilisation technique (based on reversing the motor direction during flight) which ensured stable position and yaw response. To address the more realistic scenario, of a mid flight motor failure, we developed an EKF which relies on inertial measurements and monitors the health status of all the motors. This was used for autonomous detection and recovery in experiments where one motor was abruptly switched off. Apart from the advantages presented above (i.e. valid model irrespectively of the MAV state, robustness in mechanical failures) the developed controller does not rely on the existence of an attitude or rate controller. This property can be used for tracking of full state dynamically feasible trajectories and natively enables the use of the same controller as e.g. an attitude or rate controller by nullifying the cost function terms that are not required.

A rushed comparison between the linear controller (Chapter 3) and the non linear one (Chapter 4) would always result in favor of the latter. However, in practice this is not always the case. The unified control of both the fast attitude and rate as well as the slow position dynamics of the MAV from a single controller requires state estimates to be provided at a high update rate. We experimentally identified that the NMPC controlled system can experience unstable behavior for update rates lower than 70 Hz while to avoid this, the controller was always run at rates equal or greater than 100 Hz. To eliminate potential instability issues related to delays between perception and actuation, the NMPC was only tested on relative powerful CPUs while special effort was made to phase out potential sources of delays in the software pipeline. In contrast, the position and attitude dynamics separation of the linear MPC allows the position controller to run at much lower rates. Specifically, in the presented experiments the position loop rate was 20 Hz (while the system was still stable when the rate was reduced to 10 Hz) which shows that the same controller can be used in conjunction with a low rate state estimation scheme e.g. a SLAM system that runs at camera rate. As a consequence of the low rate updates and

the simpler optimisation problem, we have successfully run the linear MPC using less powerful onboard computers such as an Odroid XU4[1]. When available, the extra computational power can be devoted to increasing the length of the prediction horizon of the linear MPC which given valid model predictions would result a performance improvement compared to implementations with a shorter horizon. Given the above, the choice between the two controllers on a real system, should not solely depend on their theoretical principles but should be a function of the platform hardware, the available state estimation and more importantly the desired MAV application.

The final contribution presented in Chapter 5 was an extension of the NMPC for tasks requiring interaction with the environment. There, the motion of an MAV-arm system was jointly controlled while also considering the quasi static forces introduced by the arm and its interaction with the environment. This approach compensates for the CoM displacement due to the end effector motion as well as the interaction forces acting on the end effector which are further integrated into forces and moments acting on the MAV base. We consider the method generic enough to be extended for multiple contact points as well as to be applied on similar systems since the control model is platform and arm-type agnostic. Overall, our custom built system achieves sub centimetre accuracy in "aerial writing" which was used as an example scenario that requires accuracy.

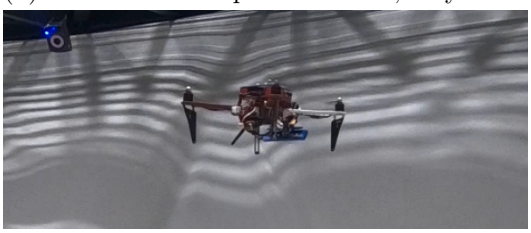
All in all, the experimental verification presented in this work would not be possible without the: (i) the hardware platforms built, (ii) the state estimation software, (iii) the parameter identification all presented in Chapter 2 and (iv) the complete software framework that enables easy MAV operation. The latter includes the controllers and estimators described in this work, provides an interface for flight tasks, implements multiple control modes for redundancy as well as contains the state machines which monitor the health status of all vital components (e.g. IMU and state estimation rate). So far, our framework has been used in many indoor and outdoor experiments including more than 100 public demonstrations with some examples shown in Figure 6.1.

[1]See https://www.odroid.co.uk/hardkernel-odroid-xu4. Accessed April 2020.

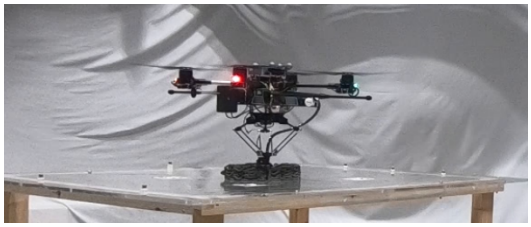
(a) Initial OKVIS enabled flights, April 2016.

(b) Live demo at Imperial Festival, May 2016.

(c) Live demo at Imperial Festival, May 2017.

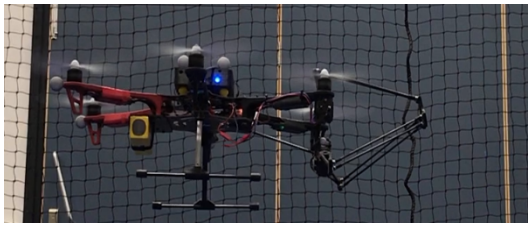


(d) AABM integration week, December 2017.



(e) AABM integration week, July 2018.

(f) Initial NMPC tests, September 2018.



(g) Initial MAV-arm tests, May 2019.

(h) GPS enabled flights, June 2019.

Figure 6.1: Examples of experiments and live demos accomplished with our software framework.

6.2 Future work

Regarding the linear MPC presented in Chapter 3, its natural extension to an NMPC was presented in Chapter 4. Therefore, future extensions should be focused on the engineering implementation such as including the effect of external disturbance in the control model as in [Kamel et al., 2017a] and use the fastest QP solver [Frison and Diehl, 2020] to date.

Our analysis regarding future work focuses on the two NMPC algorithms presented in Chapters 4 and 5 respectively.

More accurate system modelling. In both NMPC formulations we ignored less significant phenomena such as the gyroscopic moments, the motor dynamics as well as aerodynamic drag. Incorporation of the first two into the control model, requires the rotor angular velocity to be considered as the control input which further eliminates the need of the control allocation algorithm as the actuator commands will be directly computed by the NMPC optimisation. Compared to the existing approach, this comes at the cost of increased computational complexity for MAVs with more than four motors. However, it ensures that the motor dynamics are taken into account (e.g. in practice the motors cannot accelerate or decelerate infinitely fast as assumed by our approach) and most importantly yields no approximation for the feasible control input set (e.g. feasible control input set presented in Section 4.6 approximated as a 4D hyperrectangle in the current approach). Regarding the aerodynamic effects, recent work [Faessler et al., 2017] showed that tracking performance (especially when the MAV has to follow long in duration aggressive maneuvers) can be further improved when these are taken into account by the control model. The main challenge however is the tedious identification of the drag coefficients which requires experiments to be done in controlled wind tunnel environments.

Online model refinement. In this work, the model parameters (e.g. MAV mass, inertia and motor constants) were identified offline and the control model remained unchanged while flying. We can make the controller adaptive to model changes by online estimating these by e.g. adopting the approaches discussed in [Bähnemann et al., 2017] and [Burri et al., 2018] and updating the model online. By doing so, we do not only take into account for parameters which change over time (e.g. the mass of the MAV in the AABM experiments), but also correct for model or parameter estimation errors which affect the flight performance. An example of the latter includes the online estimation of the position of CoM which is affected even by a small misplacement of the MAV battery. Online model refinement will eliminate the need of using integral terms (as done in the experiments presented in Section 3.7) in order to compensate steady state offsets arising from mismatches between the real and control model.

Quantitative comparison between the MPC and NMPC. The controller described in Chapter 4 was designed in order to overcome the limitations of its linear counterpart described in Chapter 3. While the two methods were presented in detail, a quantitative comparison of the tracking performance achieved was not performed.

As also shown in similar work [Kamel et al., 2017b], we anticipate the NMPC to dominate when fast maneuvers are required while both the MPC and the NMPC should perform similar for slow or static trajectories. However, it would be beneficial to further quantify the performance improvement due to the use of a nonlinear model approach as a function of e.g. the reference velocity, acceleration or the number of DoFs excited at the same time. Such an analysis can provide an objective criterion as to when switching to a NMPC is beneficial.

Use the NMPC in a machine learning framework. In recent learning-based control works such as [Zhang et al., 2016, Müller et al., 2018b, Kaufmann et al., 2020] the MAV control policy is trained entirely in simulation in a supervised fashion using training data provided by either human demonstrations or a preexisting closed loop controller. Such approaches have the potential to outperform classic MPC approaches in terms of computational cost while, depending on the network structure, may not require an explicit state estimation scheme during test time. We believe that our NMPC framework can be utilised to provide demonstrations to be used during the training phase of the neural network while it can be easily adapted to different MAV models used in the network design (e.g. consider the MAV rates as the control input). The proven capability of running in real-time enables the NMPC to run as a safety backup or, as an attempt to handle the simulation-to-reality gap, provide online training data in the event of e.g. an MAV state or reference on which the trained policy does not perform well.

Investigate robustness of fault detection EKF. The EKF used for motor failure detection was only tested in indoor flights in the absence of external disturbances. Theoretically, a large in magnitude disturbance (e.g. a wind gust) can disturb the MAV in a way to "trick" the EKF and trigger a false positive. This has to be further investigated prior to the deployment on an MAV which operates in outdoor environments. A potential solution to this problem would require the system state to be augmented with wind states which should also be estimated online. An alternative solution, which does not require any software modification, includes lowering the threshold for the health status upper bound $L(h_i + 3\sigma_i)$ used for signalling a fault of the i^{th} motor. Lowering this threshold will decrease the probability of a false positive in the expense of an increased detection delay. Finally, a more robust solution would be to further augment the EKF with per-motor encoder and current measurements and their corresponding models. However, this option requires additional hardware

which is not available to every MAV and therefore defeats the main advantage of the original method which is the algorithmic only implementation.

Multi-body dynamic modelling. The hybrid MAV-arm model only considers the quasi static forces introduced by the arm. In order to utilise the full potential of the model based approach, a multi-body dynamic model has to be formulated. This will allow for agile combined maneuvers, as well as capture the effect of the servo torque (required for the arm motion) on the MAV base. From a hardware perspective this approach would require the manipulator motors to support direct torque control which is not possible with the cheap and lightweight servos used in this work. Most of the commercially available motors with such a capability are designed for ground robots and are -due to weight limitations- unsuitable for use in aerial vehicles of similar scale as ours. A potential solution includes the design of proprietary motor-controller hardware setups as in [Kamel et al., 2016b] that take into account the maximum required torque for controlling the arm's motion as well as the limited payload capacity of MAVs.

Closed loop control of the manipulation task. As mentioned in Chapter 5 the overall performance is sensitive to wrong estimates of the contact frame. This is because the contact forces are part of the model and the NMPC anticipates them to occur at a known position and time instant. To overcome this, one can use the force sensor measurements and the MAV state in order to estimate the contact frame normal direction online. This will not only account for small calibration errors but will enable aerial interaction with e.g surfaces which are not perfectly flat. Another possible extension which will improve the overall performance of the system is to close the loop of the interaction with the environment task with visual feedback. This will compensate for errors arising from either small MAV model inaccuracies or arm kinematics mismatches and will handle the existence of drift in the MAV position estimates which is usually the case for vision based localisation.

Tests with omnidirectional MAVs. A main advantage of the NMPC for aerial manipulation is that a reference contact force can be provided and followed by the system. Since the used underactuated MAV cannot produce lateral forces in the body frame, the desired force was produced through the inertial coupling between the arm and the MAV. Also, in our experimental setup the maximum contact force was further constrained by the low stall torque of the delta arm servos. In order to showcase our algorithm in scenarios requiring exertion of significant lateral forces,

we would like to test our NMPC with hardware platforms such as the ones presented in [Bodie et al., 2019] and [Ryll et al., 2019] which have much higher capabilities in terms of lateral force control.

Collision free navigation. In this work we presented algorithms that can run in realtime on commercially available hardware and form the basis for more complex/realistic experimental scenarios. However, there are a few missing components which are crucial for successful deployment in the unstructured real world. The most important one is the generation of collision free trajectories since all presented experiments relied on the assumption of an obstacle-free space. This is usually achieved by online mapping of the environment e.g. [Oleynikova et al., 2017, Vespa et al., 2018] using onboard sensors such as RGBD cameras and subsequently using the built map for generation of obstacle free paths e.g [Zucker et al., 2013, Richter et al., 2016, Oleynikova et al., 2016].

Bibliography

[Achtelik et al., 2013] Achtelik, M. W., Lynen, S., Chli, M., and Siegwart, R. (2013). Inversion based direct position control and trajectory following for micro aerial vehicles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2933–2939.

[Alexis et al., 2013] Alexis, K., Huerzeler, C., and Siegwart, R. (2013). Hybrid modeling and control of a coaxial unmanned rotorcraft interacting with its environment through contact. In *2013 IEEE International Conference on Robotics and Automation*, pages 5417–5424.

[Alexis et al., 2011] Alexis, K., Nikolakopoulos, G., and Tzes, A. (2011). Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances. *Control Engineering Practice*, 19(10):1195 – 1207.

[Alexis et al., 2012] Alexis, K., Nikolakopoulos, G., and Tzes, A. (2012). Model predictive quadrotor control: attitude, altitude and position experimental studies. *IET Control Theory & Applications*, 6:1812–1827(15).

[Araar et al., 2017] Araar, O., Aouf, N., and Vitanov, I. (2017). Vision based autonomous landing of multirotor uav on moving platform. *Journal of Intelligent & Robotic Systems*, 85(2):369–384.

[Augugliaro et al., 2014] Augugliaro, F., Lupashin, S., Hamer, M., Male, C., Hehn, M., Mueller, M. W., Willmann, J. S., Gramazio, F., Kohler, M., and D'Andrea, R. (2014). The flight assembled architecture installation: Cooperative construction with flying machines. *IEEE Control Systems Magazine*, 34(4):46–64.

[Baca et al., 2016] Baca, T., Loianno, G., and Saska, M. (2016). Embedded model predictive control of unmanned micro aerial vehicles. In *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 992–997.

[Bähnemann et al., 2017] Bähnemann, R., Pantic, M., Popovic, M., Schindler, D., Tranzatto, M., Kamel, M., Grimm, M., Widauer, J., Siegwart, R., and Nieto, J. I. (2017). THE ETH-MAV team in the MBZ international robotics challenge. *CoRR*, abs/1710.08275.

[Bangura and Mahony, 2014] Bangura, M. and Mahony, R. (2014). Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773 – 11780. 19th IFAC World Congress.

[Besl and McKay, 1992] Besl, P. and McKay, N. (1992). A method for Registration of 3D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256.

[Beul et al., 2017] Beul, M., Houben, S., Nieuwenhuisen, M., and Behnke, S. (2017). Fast autonomous landing on a moving target at mbzirc. In *2017 European Conference on Mobile Robots (ECMR)*.

[Bhargavapuri et al., 2019] Bhargavapuri, M., Shastry, A. K., Sinha, H., Sahoo, S. R., and Kothari, M. (2019). Vision-based autonomous tracking and landing of a fully-actuated rotorcraft. *Control Engineering Practice*, 89:113 – 129.

[Bodie et al., 2019] Bodie, K., Brunner, M., Pantic, M., Walser, S., Pfändler, P., Angst, U., Siegwart, R., and Nieto, J. (2019). An Omnidirectional Aerial Manipulation Platform for Contact-Based Inspection. In *Robotics: Science and Systems*.

[Bodie et al., 2020] Bodie, K., Brunner, M., Pantic, M., Walser, S., Pfändler, P., Angst, U., Siegwart, R., and Nieto, J. (2020). Active interaction force control for omnidirectional aerial contact-based inspection.

[Bodie et al., 2018] Bodie, K., Taylor, Z., Kamel, M., and Siegwart, R. (2018). Towards efficient full pose omnidirectionality with overactuated mavs. *CoRR*, abs/1810.06258.

[Bouabdallah, 2007] Bouabdallah, S. (2007). Design and control of quadrotors with application to autonomous flying.

[Bouabdallah et al., 2004] Bouabdallah, S., Murrieri, P., and Siegwart, R. (2004). Design and control of an indoor micro quadrotor. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4393–4398. IEEE.

[Brescianini and D'Andrea, 2016] Brescianini, D. and D'Andrea, R. (2016). Design, modeling and control of an omni-directional aerial vehicle. In *IEEE International Conference on Robotics and Automation*, pages 3261–3266.

[Brescianini and D'Andrea, 2018] Brescianini, D. and D'Andrea, R. (2018). Tilt-prioritized quadrocopter attitude control. *IEEE Transactions on Control Systems Technology*, pages 1–12.

[Brescianini and D'Andrea, 2018a] Brescianini, D. and D'Andrea, R. (2018a). Computationally efficient trajectory generation for fully actuated multirotor vehicles. *IEEE Transactions on Robotics*, 34:555–571.

[Brescianini and D'Andrea, 2018b] Brescianini, D. and D'Andrea, R. (2018b). An omni-directional multirotor vehicle. *Mechatronics*, 55:76 – 93.

[Bucki and Mueller, 2019] Bucki, N. and Mueller, M. W. (2019). Design and control of a passively morphing quadcopter. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9116–9122.

[Burri et al., 2018] Burri, M., Bloesch, M., Taylor, Z., Siegwart, R., and Nieto, J. (2018). A framework for maximum likelihood parameter identification applied on MAVs. *Journal of Field Robotics*, 35(1):5–22.

[Bähnemann et al., 2017] Bähnemann, R., Burri, M., Galceran, E., Siegwart, R., and Nieto, J. (2017). Sampling-based motion planning for active multirotor system identification. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3931–3938.

[Castillo et al., 2003] Castillo, P., Dzul, A., and Lozano, R. (2003). Real-time stabilization and tracking of a four rotor mini-rotorcraft. In *2003 European Control Conference (ECC)*, pages 3123–3128.

[Cataldi et al., 2016] Cataldi, E., Muscio, G., Trujillo, M. A., Rodriguez, Y., Pierri, F., Antonelli, G., Caccavale, F., Viguria, A., Chiaverini, S., and Ollero, A. (2016). Impedance control of an aerial-manipulator: Preliminary results. In *2016*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3848–3853.

[Chermprayong et al., 2019] Chermprayong, P., Zhang, K., Xiao, F., and Kovac, M. (2019). An integrated delta manipulator for aerial repair: A new aerial robotic system. *IEEE Robotics Automation Magazine*, 26(1):54–66.

[Clark et al., 2017] Clark, R., Wang, S., Wen, H., Markham, A., and Trigoni, N. (2017). Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem.

[Dai et al., 2020] Dai, A., Papatheodorou, S., Funk, N., Tzoumanikas, D., and S., L. (2020). Fast Frontier-based Information-driven Autonomous Exploration with an MAV. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Darivianakis et al., 2014] Darivianakis, G., Alexis, K., Burri, M., and Siegwart, R. (2014). Hybrid predictive control for aerial robotic physical interaction towards inspection operations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 53–58.

[de Crousaz et al., 2015] de Crousaz, C., Farshidian, F., Neunert, M., and Buchli, J. (2015). Unified motion control for dynamic quadrotor maneuvers demonstrated on slung load and rotor failure tasks. In *IEEE International Conference on Robotics and Automation*, pages 2223–2229.

[Doherty and Rudol, 2007] Doherty, P. and Rudol, P. (2007). A uav search and rescue scenario with human body detection and geolocalization. In Orgun, M. A. and Thornton, J., editors, *AI 2007: Advances in Artificial Intelligence*, pages 1–13, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Dorling et al., 2017] Dorling, K., Heinrichs, J., Messier, G. G., and Magierowski, S. (2017). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85.

[Faessler et al., 2017] Faessler, M., Falanga, D., and Scaramuzza, D. (2017). Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight. *IEEE Robotics and Automation Letters*, 2(2):476–482.

[Falanga et al., 2018] Falanga, D., Foehn, P., Lu, P., and Scaramuzza, D. (2018). Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8.

[Falanga et al., 2019] Falanga, D., Kleber, K., Mintchev, S., Floreano, D., and Scaramuzza, D. (2019). The foldable drone: A morphing quadrotor that can squeeze and fly. *IEEE Robotics and Automation Letters*, 4(2):209–216.

[Ferreau et al., 2014] Ferreau, H., Kirches, C., Potschka, A., Bock, H., and Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363.

[Findeisen and Allgöwer, 2002] Findeisen, R. and Allgöwer, F. (2002). An introduction to nonlinear model predictive control. In *21ST BENELUX MEETING ON SYSTEMS AND CONTROL, EIDHOVEN*, pages 1–23.

[Foehn and Scaramuzza, 2018] Foehn, P. and Scaramuzza, D. (2018). Onboard State Dependent LQR for Agile Quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6566–6572.

[Freddi et al., 2011] Freddi, A., Lanzon, A., and Longhi, S. (2011). A feedback linearization approach to fault tolerance in quadrotor vehicles. *IFAC Proceedings Volumes*, 44(1):5413 – 5418. 18th IFAC World Congress.

[Fresk et al., 2017] Fresk, E., Wuthier, D., and Nikolakopoulos, G. (2017). Generalized center of gravity compensation for multirotors with application to aerial manipulation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4424–4429.

[Frison and Diehl, 2020] Frison, G. and Diehl, M. (2020). Hpipm: a high-performance quadratic programming framework for model predictive control.

[Furgale et al., 2013] Furgale, P., Rehder, J., and Siegwart, R. (2013). Unified temporal and spatial calibration for multi-sensor systems. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1280–1286. IEEE.

[Furrer et al., 2016] Furrer, F., Burri, M., Achtelik, M., and Siegwart, R. (2016). *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter

RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham.

[Gandhi et al., 2017] Gandhi, D., Pinto, L., and Gupta, A. (2017). Learning to fly by crashing.

[Garcia et al., 1989] Garcia, C. E., Prett, D. M., and Morari, M. (1989). Model predictive control: Theory and practice - a survey. *Autom.*, 25(3):335–348.

[Garimella and Kobilarov, 2015] Garimella, G. and Kobilarov, M. (2015). Towards model-predictive control for aerial pick-and-place. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4692–4697.

[Giftthaler et al., 2018] Giftthaler, M., Neunert, M., Stäuble, M., and Buchli, J. (2018). The Control Toolbox - an open-source C++ library for robotics, optimal and model predictive control. In *IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 123–129.

[Giftthaler et al., 2017] Giftthaler, M., Neunert, M., Stäuble, M., Buchli, J., and Diehl, M. (2017). A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control. *CoRR*, abs/1711.11006.

[Giribet et al., 2017] Giribet, J. I., Pose, C. D., and Mas, I. (2017). Fault tolerant control of an hexacopter with a tilted-rotor configuration. In *2017 XVII Workshop on Information Processing and Control (RPIC)*, pages 1–6.

[Giusti et al., 2016] Giusti, A., Guzzi, J., Ciresan, D., He, F.-L., Rodriguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., Scaramuzza, D., and Gambardella, L. (2016). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*.

[Ham, 2018] Ham, A. M. (2018). Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies*, 91:1 – 14.

[Herceg et al., 2013] Herceg, M., Kvasnica, M., Jones, C. N., and Morari, M. (2013). Multi-parametric toolbox 3.0. In *2013 European Control Conference (ECC)*, pages 502–510.

[Houska et al., 2011] Houska, B., Ferreau, H., and Diehl, M. (2011). ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312.

[Hwangbo et al., 2017] Hwangbo, J., Sa, I., Siegwart, R., and Hutter, M. (2017). Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103.

[Ikeda et al., 2017] Ikeda, T., Yasui, S., Fujihara, M., Ohara, K., Ashizawa, S., Ichikawa, A., Okino, A., Oomichi, T., and Fukuda, T. (2017). Wall contact by octo-rotor uav with one dof manipulator for bridge inspection. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5122–5127.

[Invernizzi and Lovera, 2017] Invernizzi, D. and Lovera, M. (2017). Geometric tracking control of a quadcopter tiltrotor uav. *IFAC-PapersOnLine*, 50(1):11565 – 11570. 20th IFAC World Congress.

[Izadi et al., 2011] Izadi, H. A., Zhang, Y., and Gordon, B. W. (2011). Fault tolerant model predictive control of quad-rotor helicopters with actuator fault estimation. *IFAC Proceedings Volumes*, 44(1):6343 – 6348. 18th IFAC World Congress.

[Jimenez-Cano et al., 2013] Jimenez-Cano, A. E., Martin, J., Heredia, G., Ollero, A., and Cano, R. (2013). Control of an aerial robot with multi-link arm for assembly tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4916–4921.

[Jung et al., 2018] Jung, S., Hwang, S., Shin, H., and Shim, D. H. (2018). Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, 3(3):2539–2544.

[Kaess, 2013] Kaess, M. (2013). Apriltags c++ library.

[Kahn et al., 2017] Kahn, G., Zhang, T., Levine, S., and Abbeel, P. (2017). Plato: Policy learning using adaptive trajectory optimization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3342–3349.

[Kamel et al., 2015] Kamel, M., Alexis, K., Achtelik, M., and Siegwart, R. (2015). Fast nonlinear model predictive control for multicopter attitude tracking on so(3). In *2015 IEEE Conference on Control Applications (CCA)*, pages 1160–1166.

[Kamel et al., 2016a] Kamel, M., Alexis, K., and Siegwart, R. (2016a). Design and modeling of dexterous aerial manipulator. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4870–4876.

[Kamel et al., 2017] Kamel, M., Alonso-Mora, J., Siegwart, R., and Nieto, J. (2017). Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 236–243.

[Kamel et al., 2017a] Kamel, M., Burri, M., and Siegwart, R. (2017a). Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles. *IFAC-PapersOnLine*, 50(1):3463 – 3469. 20th IFAC World Congress.

[Kamel et al., 2016b] Kamel, M., Comari, S., and Siegwart, R. (2016b). Full-body multi-objective controller for aerial manipulation. In *2016 24th Mediterranean Conference on Control and Automation (MED)*, pages 659–664.

[Kamel et al., 2017b] Kamel, M., Stastny, T., Alexis, K., and Siegwart, R. (2017b). *Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System*, pages 3–39. Springer International Publishing, Cham.

[Kamel et al., 2018] Kamel, M., Verling, S., Elkhatib, O., Sprecher, C., Wulkop, P., Taylor, Z., Siegwart, R., and Gilitschenski, I. (2018). Voliro: An Omnidirectional Hexacopter With Tiltable Rotors. *arXiv preprint arXiv:1801.04581*.

[Kamel et al., 2018] Kamel, M., Verling, S., Elkhatib, O., Sprecher, C., Wulkop, P., Taylor, Z., Siegwart, R., and Gilitschenski, I. (2018). The voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle. *IEEE Robotics Automation Magazine*, 25(4):34–44.

[Kaufmann et al., 2018] Kaufmann, E., Loquercio, A., Ranftl, R., Dosovitskiy, A., Koltun, V., and Scaramuzza, D. (2018). Deep drone racing: Learning agile flight in dynamic environments.

[Kaufmann et al., 2020] Kaufmann, E., Loquercio, A., Ranftl, R., Müller, M., Koltun, V., and Scaramuzza, D. (2020). Deep drone acrobatics.

[Kerrigan and Maciejowski, 2000] Kerrigan, E. C. and Maciejowski, J. M. (2000). Soft constraints and exact penalty functions in model predictive control. In *Proc. UKACC International Conference (Control*.

[Kessens et al., 2016] Kessens, C. C., Thomas, J., Desai, J. P., and Kumar, V. (2016). Versatile aerial grasping using self-sealing suction. In *IEEE International Conference on Robotics and Automation*, pages 3249–3254.

[Kim et al., 2013] Kim, S., Choi, S., and Kim, H. J. (2013). Aerial manipulation using a quadrotor with a two DOF robotic arm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4990–4995.

[Koch et al., 2019] Koch, W., Mancuso, R., West, R., and Bestavros, A. (2019). Reinforcement learning for uav attitude control. 3(2).

[Kouris and Bouganis, 2018] Kouris, A. and Bouganis, C. (2018). Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9.

[Kvasnica et al., 2004] Kvasnica, M., Grieder, P., Baotić, M., and Morari, M. (2004). Multi-parametric toolbox (mpt). In Alur, R. and Pappas, G. J., editors, *Hybrid Systems: Computation and Control*, pages 448–462, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Lee et al., 2012] Lee, D., Ryan, T., and Kim, H. J. (2012). Autonomous landing of a vtol uav on a moving platform using image-based visual servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 971–976. IEEE.

[Lee et al., 2010] Lee, T., Leok, M., and McClamroch, N. H. (2010). Geometric tracking control of a quadrotor UAV on SE(3). In *IEEE Conference on Decision and Control*, pages 5420–5425.

[Leutenegger et al., 2014] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2014). Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, page 0278364914554813.

[Leutenegger et al., 2014] Leutenegger, S., Melzer, A., Alexis, K., and Siegwart, R. (2014). Robust state estimation for small unmanned airplanes. In *2014 IEEE Conference on Control Applications (CCA)*, pages 1003–1010.

[Li et al., 2018] Li, W., Saeedi, S., McCormac, J., Clark, R., Tzoumanikas, D., Ye, Q., Huang, Y., Tang, R., and Leutenegger, S. (2018). Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *British Machine Vision Conference (BMVC)*.

[Lippiello and Ruggiero, 2012] Lippiello, V. and Ruggiero, F. (2012). Cartesian impedance control of a uav with a robotic arm. *IFAC Proceedings Volumes*, 45(22):704 – 709. 10th IFAC Symposium on Robot Control.

[Liu et al., 2014] Liu, P. C. K., Chen, A. Y., Huang, Y.-N., Han, J.-Y., Lai, J.-S., Kang, S.-C. J., Wu, T.-H., Wen, M.-C., and Tsai, M.-H. (2014). A review of rotorcraft unmanned aerial vehicle (uav)developments and applications in civil engineering.

[Loquercio et al., 2020] Loquercio, A., Kaufmann, E., Ranftl, R., Dosovitskiy, A., Koltun, V., and Scaramuzza, D. (2020). Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 36(1):1–14.

[Loquercio et al., 2018] Loquercio, A., Maqueda, A. I., del-Blanco, C. R., and Scaramuzza, D. (2018). Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095.

[Lu and van Kampen, 2015] Lu, P. and van Kampen, E. (2015). Active fault-tolerant control for quadrotors subjected to a complete rotor failure. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4698–4703.

[Lunni et al., 2017] Lunni, D., Santamaria-Navarro, A., Rossi, R., Rocco, P., Bascetta, L., and Andrade-Cetto, J. (2017). Nonlinear model predictive control for aerial manipulation. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 87–93.

[Lynen et al., 2013] Lynen, S., Achtelik, M., Weiss, S., Chli, M., and Siegwart, R. (2013). A robust and modular multi-sensor fusion approach applied to mav navigation. In *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*.

[Maciejowski, 2002] Maciejowski, J. (2002). *Predictive Control: With Constraints*. Pearson Education. Prentice Hall.

[Markley et al., 2007] Markley, L., Cheng, Y., Crassidis, J., and Oshman, Y. (2007). Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30:1193–1196.

[Marks et al., 2012] Marks, A., Whidborne, J. F., and Yamamoto, I. (2012). Control allocation for fault tolerant control of a vtol octorotor. *Proceedings of 2012 UKACC International Conference on Control*, pages 357–362.

[Mattingley and Boyd, 2012] Mattingley, J. and Boyd, S. (2012). Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27.

[Mazeh et al., 2018] Mazeh, H., Saied, M., Shraim, H., and Francis, C. (2018). Fault-tolerant control of an hexarotor unmanned aerial vehicle applying outdoor tests and experiments. *IFAC-PapersOnLine*, 51(22):312 – 317. 12th IFAC Symposium on Robot Control SYROCO 2018.

[Mellinger and Kumar, 2011] Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525.

[Mellinger et al., 2011] Mellinger, D., Lindsey, Q., Shomin, M., and Kumar, V. (2011). Design, modeling, estimation and control for aerial grasping and manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2668–2673.

[Mersha et al., 2014] Mersha, A. Y., Stramigioli, S., and Carloni, R. (2014). Exploiting the dynamics of a robotic manipulator for control of uavs. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1741–1746.

[Michieletto et al., 2017] Michieletto, G., Ryll, M., and Franchi, A. (2017). Control of statically hoverable multi-rotor aerial vehicles and application to rotor-failure robustness for hexarotors. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2747–2752.

[Michieletto et al., 2018] Michieletto, G., Ryll, M., and Franchi, A. (2018). Fundamental actuation properties of multirotors: Force–moment decoupling and fail–safe robustness. *IEEE Transactions on Robotics*, 34(3):702–715.

[Morari and Lee, 1999] Morari, M. and Lee, J. H. (1999). Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667 – 682.

[Mueller and D'Andrea, 2014] Mueller, M. W. and D'Andrea, R. (2014). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 45–52.

[Mur-Artal and Tardós, 2017] Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*.

[Murray and Chu, 2015] Murray, C. C. and Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86 – 109.

[Muskardin et al., 2016] Muskardin, T., Balmer, G., Wlach, S., Kondak, K., Laiacker, M., and Ollero, A. (2016). Landing of a fixed-wing uav on a mobile ground vehicle. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1237–1242.

[Müller et al., 2018a] Müller, M., Casser, V., Smith, N., Michels, D. L., and Ghanem, B. (2018a). Teaching uavs to race: End-to-end regression of agile controls in simulation.

[Müller et al., 2018b] Müller, M., Casser, V., Smith, N., Michels, D. L., and Ghanem, B. (2018b). Teaching uavs to race: End-to-end regression of agile controls in simulation.

[Nayak et al., 2018] Nayak, V., Papachristos, C., and Alexis, K. (2018). Design and control of an aerial manipulator for contact-based inspection. *ArXiv*, abs/1804.03756.

[Neunert et al., 2016] Neunert, M., de Crousaz, C., Furrer, F., Kamel, M., Farshidian, F., Siegwart, R., and Buchli, J. (2016). Fast nonlinear model predictive control for unified trajectory optimization and tracking. In *IEEE International Conference on Robotics and Automation*, pages 1398–1404.

[Nguyen and Lee, 2013] Nguyen, H. and Lee, D. (2013). Hybrid force/motion control and internal dynamics of quadrotors for tool operation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3458–3464.

[Nguyen and Hong, 2019] Nguyen, N. and Hong, S. (2019). Fault diagnosis and fault-tolerant control scheme for quadcopter uavs with a total loss of actuator. *Energies*, 12(6):1139.

[Nguyen and Hong, 2018] Nguyen, N. P. and Hong, S. K. (2018). Fault-tolerant control of quadcopter uavs using robust adaptive sliding mode approach. *Energies*, 12(1):95.

[Nguyen et al., 2019] Nguyen, N. P., Xuan Mung, N., and Hong, S. K. (2019). Actuator fault detection and fault-tolerant control for hexacopter. *Sensors*, 19(21).

[Oettershagen et al., 2016] Oettershagen, P., Melzer, A., Mantel, T., Rudin, K., Stastny, T., Wawrzacz, B., Hinzmann, T., Leutenegger, S., Alexis, K., and Siegwart, R. (2016). Design of small hand-launched solar-powered uavs: From concept study to a multi-day world endurance record flight. *Journal of Field Robotics*.

[Ohnishi et al., 2017] Ohnishi, Y., Takaki, T., Aoyama, T., and Ishii, I. (2017). Development of a 4-joint 3-dof robotic arm with anti-reaction force mechanism for a multicopter. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 985–991.

[Oleynikova et al., 2016] Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., and Galceran, E. (2016). Continuous-time trajectory optimization for online uav replanning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339.

[Oleynikova et al., 2017] Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., and Nieto, J. (2017). Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373.

[Orsag et al., 2014] Orsag, M., Korpela, C., Bogdan, S., and Oh, P. (2014). Valve turning using a dual-arm aerial manipulator. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 836–841.

[Orsag et al., 2013] Orsag, M., Korpela, C., and Oh, P. (2013). Modeling and control of mm-uav: Mobile manipulating unmanned aerial vehicle. *Journal of Intelligent and Robotic Systems*, 69(1):227–240.

[Ozaslan et al., 2017] Ozaslan, T., Loianno, G., Keller, J., Taylor, C. J., Kumar, V., Wozencraft, J. M., and Hood, T. (2017). Autonomous navigation and mapping for inspection of penstocks and tunnels with mavs. *IEEE Robotics and Automation Letters*, 2(3):1740–1747.

[Papachristos et al., 2012] Papachristos, C., Alexis, K., and Tzes, A. (2012). Towards a high-end unmanned tri-tiltrotor: design, modeling and hover control. In *2012 20th Mediterranean Conference on Control Automation (MED)*, pages 1579–1584.

[Papachristos et al., 2014a] Papachristos, C., Alexis, K., and Tzes, A. (2014a). Efficient force exertion for aerial robotic manipulation: Exploiting the thrust-vectoring authority of a tri-tiltrotor uav. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4500–4505.

[Papachristos et al., 2014b] Papachristos, C., Alexis, K., and Tzes, A. (2014b). Technical activities execution with a tiltrotor uas employing explicit model predictive control. *IFAC Proceedings Volumes*, 47(3):11036 – 11042. 19th IFAC World Congress.

[Papachristos et al., 2016] Papachristos, C., Alexis, K., and Tzes, A. (2016). Dual-authority thrust-vectoring of a tri-tiltrotor employing model predictive control. *Journal of Intelligent & Robotic Systems*, 81(3-4):471.

[Pierrot et al., 1990] Pierrot, F., Reynaud, C., and Fournier, A. (1990). Delta: a simple and efficient parallel robot. *Robotica*, 8(2):105–109.

[Rajappa et al., 2015] Rajappa, S., Ryll, M., Bülthoff, H. H., and Franchi, A. (2015). Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4006–4013.

[Ranjbaran and Khorasani, 2010] Ranjbaran, M. and Khorasani, K. (2010). Fault recovery of an under-actuated quadrotor aerial vehicle. In *49th IEEE Conference on Decision and Control (CDC)*, pages 4385–4392.

[Richter et al., 2016] Richter, C., Bry, A., and Roy, N. (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Springer Tracts in Advanced Robotics*, pages 649–666. Springer International Publishing.

[Riviere et al., 2018] Riviere, V., Manecy, A., and Viollet, S. (2018). Agile robotic fliers: A morphing-based approach. *Soft Robotics*, 5(5):541–553. PMID: 29846133.

[Rose, 2013] Rose, C. (2013). Amazon unveils futuristic plan: Delivery by drone. URL: https://www.cbsnews.com/news/amazon-unveils-futuristic-plan-delivery-by-drone/.

[Ruggiero et al., 2015] Ruggiero, F., Trujillo, M. A., Cano, R., Ascorbe, H., Viguria, A., Peréz, C., Lippiello, V., Ollero, A., and Siciliano, B. (2015). A multilayer control for multirotor uavs equipped with a servo robot arm. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4014–4020.

[Ryll et al., 2016] Ryll, M., Bicego, D., and Franchi, A. (2016). Modeling and control of fast-hex: A fully-actuated by synchronized-tilting hexarotor. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1689–1694.

[Ryll et al., 2019] Ryll, M., Muscio, G., Pierri, F., Cataldi, E., Antonelli, G., Caccavale, F., Bicego, D., and Franchi, A. (2019). 6D interaction control with aerial robots: The flying end-effector paradigm. *The International Journal of Robotics Research*, 38(9):1045–1062.

[Sa et al., 2018a] Sa, I., Kamel, M., Khanna, R., Popović, M., Nieto, J., and Siegwart, R. (2018a). Dynamic system identification, and control for a cost-effective and open-source multi-rotor mav. In Hutter, M. and Siegwart, R., editors, *Field and Service Robotics*, pages 605–620, Cham. Springer International Publishing.

[Sa et al., 2018b] Sa, I., Popović, M., Khanna, R., Chen, Z., Lottes, P., Liebisch, F., Nieto, J., Stachniss, C., Walter, A., and Siegwart, R. (2018b). Weedmap: A large-scale semantic weed mapping framework using aerial multispectral imaging and deep neural network for precision farming. *Remote Sensing*, 10(9):1423.

[Saeedi et al., 2019] Saeedi, S., Carvalho, E. D. C., Li, W., Tzoumanikas, D., Leutenegger, S., Kelly, P. H. J., and Davison, A. J. (2019). Characterizing visual localization and mapping datasets. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6699–6705.

[Saied et al., 2015] Saied, M., Lussier, B., Fantoni, I., Francis, C., Shraim, H., and Sanahuja, G. (2015). Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor. In *IEEE International Conference on Robotics and Automation*, pages 5266–5271.

[Saied et al., 2017] Saied, M., Lussier, B., Fantoni, I., Shraim, H., and Francis, C. (2017). Fault diagnosis and fault-tolerant control of an octorotor uav using motors speeds measurements. *IFAC-PapersOnLine*, 50(1):5263 – 5268. 20th IFAC World Congress.

[Schneider et al., 2012] Schneider, T., Ducard, G., Konrad, R., and Pascal, S. (2012). Fault-tolerant Control Allocation for Multirotor Helicopters Using Parametric Programming. In *International Micro Air Vehicle Conference and Flight Competition*, Braunschweig, Germany.

[Shuo Yang et al., 2015] Shuo Yang, Jiahang Ying, Yang Lu, and Zexiang Li (2015). Precise quadrotor autonomous landing with srukf vision perception. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2196–2201.

[Simon, 2010] Simon, D. (2010). Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory & Applications*, 4(8):1303–1318.

[Smolyanskiy et al., 2017] Smolyanskiy, N., Kamenev, A., Smith, J., and Birchfield, S. (2017). Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness.

[Suarez et al., 2018] Suarez, A., Jimenez-Cano, A. E., Vega, V. M., Heredia, G., Rodriguez-Castaño, A., and Ollero, A. (2018). Design of a lightweight dual arm system for aerial manipulation. *Mechatronics*, 50:30 – 44.

[Sun et al., 2020] Sun, S., Baert, M., van Schijndel, B. S., and de Visser, C. (2020). Upset recovery control for quadrotors subjected to a complete rotor failure from large initial disturbances. *2020 IEEE International Conference on Robotics and Automation (ICRA)*.

[Thomas et al., 2017] Thomas, J., Welde, J., Loianno, G., Daniilidis, K., and Kumar, V. (2017). Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor. *IEEE Robotics and Automation Letters*, 2(3):1762–1769.

[Tognon et al., 2019] Tognon, M., Chávez, H. A. T., Gasparin, E., Sablé, Q., Bicego, D., Mallet, A., Lany, M., Santi, G., Revaz, B., Cortés, J., and Franchi, A. (2019). A truly-redundant aerial manipulator system with application to push-and-slide inspection in industrial plants. *IEEE Robotics and Automation Letters*, 4(2):1846–1851.

[Tognon et al., 2017] Tognon, M., Yüksel, B., Buondonno, G., and Franchi, A. (2017). Dynamic decentralized control for protocentric aerial manipulators. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6375–6380.

[Tzoumanikas et al., 2020a] Tzoumanikas, D., Graule, F., Yan, Q., Shah, D., Popovic, M., and Leutenegger, S. (2020a). Aerial Manipulation Using Hybrid Force and Position NMPC Applied to Aerial Writing. In *Proceedings of Robotics: Science and Systems (RSS)*.

[Tzoumanikas et al., 2019] Tzoumanikas, D., Li, W., Grimm, M., Zhang, K., Kovac, M., and Leutenegger, S. (2019). Fully autonomous micro air vehicle flight and landing on a moving target using visual-inertial estimation and model-predictive control. *Journal of Field Robotics*, 36(1):49–77.

[Tzoumanikas et al., 2020b] Tzoumanikas, D., Yan, Q., and Leutenegger, S. (2020b). Nonlinear MPC with Motor Failure Identification and Recovery for Safe and Aggressive Multicopter Flight. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Vespa et al., 2018] Vespa, E., Nikolov, N., Grimm, M., Nardi, L., Kelly, P. H. J., and Leutenegger, S. (2018). Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151.

[Vlantis et al., 2015] Vlantis, P., Marantos, P., Bechlioulis, C. P., and Kyriakopoulos, K. J. (2015). Quadrotor landing on an inclined platform of a moving ground vehicle. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2202–2207. IEEE.

[Wang et al., 2018] Wang, S., Clark, R., Wen, H., and Trigoni, N. (2018). End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research*, 37(4-5):513–542.

[Weiss et al., 2012] Weiss, S., Achtelik, M. W., Lynen, S., Chli, M., and Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 957–964. IEEE.

[Wu et al., 2019] Wu, Y., Hu, K., Sun, X., and Ma, Y. (2019). Nonlinear control of quadrotor for fault tolerance: A total failure of one actuator. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–11.

[Xu et al., 2019] Xu, B., Li, W., Tzoumanikas, D., Bloesch, M., Davison, A., and Leutenegger, S. (2019). Mid-fusion: Octree-based object-level multi-instance dynamic slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5231–5237.

[Yang et al., 2014] Yang, B., He, Y., Han, J., and Liu, G. (2014). Rotor-flying manipulator: Modeling, analysis, and control. *Mathematical Problems in Engineering*, 2014:492965.

[Zhan et al., 2018] Zhan, H., Garg, R., Weerasekera, C. S., Li, K., Agarwal, H., and Reid, I. (2018). Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Zhang and Kovacs, 2012] Zhang, C. and Kovacs, J. M. (2012). The application of small unmanned aerial systems for precision agriculture: a review. *Precision Agriculture*, 13:693–712.

[Zhang et al., 2019] Zhang, K., Chermprayong, P., Tzoumanikas, D., Li, W., Grimm, M., Smentoch, M., Leutenegger, S., and Kovac, M. (2019). Bioinspired design of a landing system with soft shock absorbers for autonomous aerial robots. *Journal of Field Robotics*, 36(1):230–251.

[Zhang et al., 2016] Zhang, T., Kahn, G., Levine, S., and Abbeel, P. (2016). Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search.

[Zucker et al., 2013] Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., and Srinivasa, S. S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193.

[Ángel Trujillo et al., 2019] Ángel Trujillo, M., de Dios, J. R. M., Martín, C., Viguria, A., and Ollero, A. (2019). Novel aerial manipulator for accurate and robust industrial ndt contact inspection: A new tool for the oil and gas inspection industry. *Sensors*, 19(6).