# Physical Interaction and Manipulation of the Environment using Aerial Robots

Azarakhsh Keipour

CMU-RI-TR-22-17
May 2nd, 2022


The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

**Thesis Committee:**
Sebastian Scherer, *chair*
Oliver Kroemer
Wennie Tabib
Kostas Alexis, *NTNU*

*Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.*

*To my beloved wife Juwairia and my most wonderful parents Mitra and Iradge.*

# Abstract

The physical interaction of aerial robots with their environment has countless potential applications and is an emerging area with many open challenges. Fully-actuated multirotors have been introduced to tackle some of these challenges. They provide complete control over position and orientation and eliminate the need for attaching a multi-DoF manipulation arm to the robot. However, there are still several open problems before they can be used in real-world applications.

Researchers have introduced some methods for the physical interaction of fully-actuated multirotors in limited settings. Their experiments primarily use prototype-level software without an efficient path to integrating these methods into real-world applications. This thesis describes a new controller design that provides a cost-effective solution for integrating these robots with the existing software and hardware flight systems for real-world applications. It further expands the controller to physical interaction applications to show its flexibility and effectiveness.

On the other hand, the existing control approaches for fully-actuated robots assume conservative limits for the thrusts and moments available to the robot. Using conservative assumptions for these already-inefficient robots makes their interactions even less optimal and may even result in many feasible physical interaction applications becoming infeasible. This work proposes a real-time method for estimating the complete set of instantaneously available forces and moments that robots can use to optimize their physical interaction performance.

Finally, many real-world applications where aerial robots can improve the existing manual solutions deal with deformable objects. However, the perception of deformable objects and planning for their manipulation is still challenging. Additionally, no studies have been

performed to analyze the requirements of aerial tasks that involve deformable objects. This research explores how aerial physical interaction can be extended to deformable objects. It provides a detection method suitable for manipulating deformable one-dimensional objects and introduces a new perspective on planning the manipulation of these objects. It further studies the viability of working with such objects for aerial manipulators.

# Acknowledgments

I would like to thank my advisor Sebastian Scherer for his support, advice, and patience during the ups and downs of my graduate journey. His vision and passion for the field have always inspired me to push the boundaries of what robots can do in the real world. His willingness to support his students and his relentless enthusiasm and positive attitude make it really hard to say goodbye to this chapter of my life.

I would also like to thank my committee members for their feedback and suggestions on improving the work along the way up until the end. I would like to thank Kostas Alexis for finding the best ways to improve the work from scientific and storyline perspectives. His invaluable feedback has helped connect the dots of my research together before the proposal and the dissertation. His excellent understanding of details and the value of the work and his eagle-eye perspective have been irreplaceable for me. Oliver Kroemer has always helped with pushing for better and higher-quality work, and his suggestions on how to relate my different parts of work have been highly beneficial. I would like to thank Wennie Tabib for accepting the responsibility to support me at the last minute and for providing excellent feedback on how to make my work better, given the existing time constraints. Thank you to Changliu Liu, who supported me along the way, and with her great control and robotics perspective, I improved my work further. A special thanks to Guilherme A.S. Pereira, who helped start my journey in robotics on my first project and has been a great friend and mentor since. I also received very helpful advice and support during my qualifiers from George Kantor and Maxim Likhachev and am very grateful for all I have learned from them.

During the summer of 2021, in my internship at *X, the moonshot factory* (a.k.a. *Google X*), I had the opportunity to be advised by Maryam Bandari and Stefan Schaal. I would like to thank Stefan for his regular feedback and encouragement on my summer work, his support for publishing the work, and for allowing me to use it in

my Ph.D. research. Special thanks go to Maryam, who believed in me and fought hard to make my internship happen, and then gave me space to explore my ideas while providing great feedback and support during and after the summer research.

I would also like to thank Mohammadreza Mousaei for assisting with tests, publications, and lengthy discussions on the concepts, and Junyi (Jenny) Geng for her helpful feedback and helping with the final experiments. Andrew Saba and Greg Armstrong helped in building the UAVs and performing the experiments, Andrew Ashley and John Keller assisted with the flight tests. Thank you to Rogerio Bonatti, Cherie Ho, Jay Patrikar, Weikun Zhen, Kevin Pluckter, Ratnesh Madaan, and Mohammadreza Mousaei for being the most fantastic office mates in the universe. And thank you to all the other former and current AirLab and RI members for their help and support through this work and for making the graduate school a happy time, including but certainly not limited to: Milad Moghassem Hamidi, Sankalp Arora, Rohit Garg, Lucas Nogueira, Brady Moon, Yaoyu Hu, Anish Bhattacharya, Vaibhav Viswanathan, Vishal Dugar, Puru Rastogi, Dong-Ki Kim, Silvio Maeta, Guilherme Pereira, and Geetesh Dubey. No acknowledgment can be complete without thanking Nora Kazour, who was the most supportive friend and lab coordinator who could make magical things happen to facilitate our lives.

Finally, the most special thanks go to my wonderful wife, Juwairia Mulla, and my amazing parents, Iradge and Mitra, for their unconditional support and patience through the whole academic endeavor.

# Funding

This project became possible due to the support and the funding provided by the following institutions:

– The Robotics Institute Ph.D. program

– National Aeronautics and Space Administration (NASA): Grant Number 80NSSC19C010401

– X, the moonshot factory (a.k.a., Google X): AI Residency program

– Intrinsic AI, an Alphabet Company

– Near Earth Autonomy (NEA)

# Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

Our work aims to improve the state-of-the-art in physical interaction and manipulation of the environment using aerial robots and extend it to real-world applications involving rigid and deformable objects.

This chapter explains the motivation behind choosing the problems we tackled (Section 1.1), introduces the current challenges in the field (Section 1.2), describes the contributions of our work (Section 1.3), and presents the outline of this dissertation (Section 1.4).

## 1.1 Motivation

The last two decades have seen rapid growth in applications for aerial robots, ranging from hobby racing and artistic aerial shows to professional cinematography, security surveillance, and commercial infrastructure inspection. Most of these applications are performed during the flight without any physical contact of the robot with the objects and surfaces around it.

The unstable nature of the aerial robots makes their controlled physical interaction very challenging. On the other hand, traditional multirotors have a planar arrangement for their rotors, making them underactuated. These

multirotors need to tilt in the direction of the generated thrust, affecting the position of their end-effector, making the control of the end-effector position and force even more challenging.

Despite these challenges, the research on applications involving aerial robots' physical interaction with their environment has grown steadily in the last decade. To work around the difficulty of the end-effector control, many researchers attached manipulator arms with multiple degrees of freedom to their aerial robots. This approach increases the payload weight, limits the remaining forces of the robot, and requires a very complex controller to control the end effector reliably. Even with all these challenges, the result is generally a limited reachable workspace for the end-effector, which depends on the arm's properties and attachment location on the robot.

A more novel approach to tackle the challenges of physical interaction is to utilize the recently-introduced fully-actuated multirotors. These robots have a non-planar configuration for their rotors, allowing them to tilt independently from their generated thrust direction. Such an advantage eliminates the need for a heavy and complex manipulation arm attached to the multirotor. Instead, it allows the robot's attitude to be used for controlling the end-effector's position and applied forces. However, there are several drawbacks to this approach: the design of the multirotor is more complex, the robot is less efficient in generating thrusts than the planar designs, the lateral thrust is limited in most common fully-actuated designs, and the integration of the controllers of these new robots into the existing software and hardware flight stacks is not straightforward.

Even with the challenges mentioned above, researchers have found ways to develop and test methods for the physical interaction of aerial robots with their environment in limited settings. The fully-actuated multirotors have been shown to perform tasks such as peg-in-a-hole and contact inspection. However, these experiments have primarily used prototype-level software (such as MATLAB-generated code) without any time- and cost-efficient path to integrating them into an autonomy stack for real-world applications.

On the other hand, all the work has been performed with conservative

assumptions for the possible thrusts that the robot can generate. Since these robots are already inefficient and have limited thrusts, using conservative assumptions for the available thrusts makes the fully-actuated robot interactions even less optimal. It may result in many feasible physical interaction tasks deemed infeasible while resulting in sub-optimal solutions for many other applications. However, this problem is not limited to aerial robots. Estimation of a manipulator's available forces and moments has been used for four decades since the introduction of dynamic manipulability ellipsoids. Ellipsoids can be computed in real-time but are very conservative estimations of the available forces and moments. Estimating the complete set of the available forces and moments can be done using dynamic manipulability polytopes, which currently are computationally expensive and unsuitable for most real-time applications. A real-time solution for estimating the manipulability polytopes can significantly improve the performance of the physical interaction tasks not only for fully-actuated aerial robots but also for the ground manipulators.

The extent of the physical interaction research for aerial robots has been manipulating rigid (or almost-rigid) objects and contact with rigid surfaces, leaving out a major class of objects called deformable objects. Many real-world applications where the multirotors can improve the existing manual solutions include deformable objects. For example, the inspection, maintenance, and manipulation of the cables and wires at the top of the utility pole require working with deformable one-dimensional (a.k.a., deformable linear) objects.

Several new challenges arise in dealing with these objects, including the perception of their state and planning for their manipulation. Researchers in other robotics communities have been working on these problems for decades and have proposed many methods for perceiving deformable objects. However, many areas, such as detecting the state of the deformable one-dimensional objects (such as wires and cables), lack viable solutions that can be used in fully-automated settings. On the other hand, no studies have been performed to analyze the requirements of the aerial manipulation tasks involving deformable objects.

In this document, we explain our solutions for tackling the challenges mentioned above regarding the physical interaction of aerial robots with their environment. We propose a novel controller design for fully-actuated multirotors that provides a cost-effective solution for integration with the existing software and hardware flight systems for real-world applications. We then expand it to physical interaction applications to show the design's flexibility and effectiveness.

We propose a set of real-time methods for estimating the instantaneously available forces and moments that an aerial robot can use in its flight and interactions. The methods are proposed to compute the dynamic manipulability polytopes for aerial robots. However, they can be adopted for all kinds of manipulators to replace the conservative estimations made by the dynamic manipulability ellipsoids in real-time applications. We further extend the method to work in physical interactions and illustrate the power of the wrench set estimation methods by improving several aspects of real-world applications.

Finally, We explore how the physical interaction can be extended to deformable objects. We provide a deformable one-dimensional objects detection method suitable for manipulation tasks and introduce a new perspective on planning for manipulating these objects in specific settings such as utility cable boxes. We further study the viability of aerial manipulation of such objects from the end-effector precision perspective and use the proposed real-time dynamic manipulability polytope computation methods to analyze the feasibility of the wire manipulation task.

*This goal of this work is to improve the state-of-the-art in physical interaction and manipulation using aerial robots by providing faster integration for fully-actuated robots, better estimation of available wrenches and new solutions to allow real-world physical interaction with rigid and deformable objects.*

## 1.2 Challenges and Insights

Aerial robots' controlled physical interaction with their environment is still an ongoing research problem. Employing fully-actuated robots has been shown to improve the robustness of the interaction in the presence of real-world uncertainties. However, fully-actuated UAVs create new challenges that need to be addressed before they can effectively be integrated into real-world applications. On the other hand, the autonomous inspection and manipulation of cables and wires using UAVs face additional challenges due to the limited existing research on deformable objects and the lack of solutions for seemingly everyday tasks such as cable detection.

The major challenges that are currently preventing the physical interaction of fully-actuated UAVs include:

**Challenge 1. Complete flight stack for fully-actuated UAVs:**

The deployment of an aerial robot in an autonomous mission requires a complete system (a.k.a., flight stack) in addition to the flight control subsystem. The flight stack includes software such as a ground control station, planning system, communication protocols, and hardware components such as the safety pilot's remote controller. Due to the nature of fully-actuated UAVs requiring 6-D commands to control as compared to the traditional 4-D commands required for underactuated vehicles, the existing flight stack components cannot be directly used in conjunction with the existing control solutions for fully-actuated UAVs and new solutions should be found to provide a complete flight stack for these vehicles.

**Challenge 2. Wrench limits for most fully-actuated UAV designs:**

Most common fully-actuated UAV designs suffer from limited horizontal thrust compared to their overall thrust. On the other hand, controlled physical interaction tasks directly deal with the wrenches generated by the robot and can benefit from the knowledge about these limits and instantaneously avail-

able wrenches. However, the existing methods for estimating such limits for UAVs are slow and not suitable for real-time use during flight. Additionally, these methods have almost exclusively been used for UAV architecture design optimization and do not account for the relation between the applied wrenches during flight and the remaining wrenches, making them even less desirable to estimate the available wrenches during the physical interaction.

**Challenge 3. Lack of a complete pipeline for aerial manipulation of deformable objects:**

A crucial step in achieving a fully-autonomous mission involving physical interaction with objects such as wires and cables is the perception of these objects. Several segmentation methods exist that can segment these objects, which can be used for visual inspection and obstacle avoidance. On the other hand, many methods have been proposed in the industrial robotics community to track these deformable objects once their initial state is given. However, no method exists to fill the gap between the segmentation and tracking to provide the required initial state for the tracking methods in non-trivial conditions (e.g., a straight line). In order to achieve a fully autonomous mission, this gap needs to be filled.

## 1.3 Contributions

Applications involving the physical interaction of UAVs with structures involving deformable objects, such as cables and wires, have not been directly addressed in academia or industry yet. There have been efforts on some underlying challenges, especially in recent years. However, several unsolved issues were left out or needed significant improvements to see these types of applications a reality someday.

We tackle the problem of the physical interaction of UAVs by identifying the most critical challenges requiring solutions or improvements to existing solutions.

We first enable the integration of fully-actuated UAVs into existing flight stacks for real-world applications, cutting the required effort and time for deploying them into the real world.

Then we address the aerial robots' limited thrust and moment availability to alleviate the issues resulting from the wrench constraints. We propose a method to predict the limits in real-time and use this method to improve different aspects of physical interaction applications, ranging from planning and flight optimization to hardware design and control allocation.

Finally, the lack of perception methods suitable for physical interaction with wires and cables drove us to propose a detection solution to enable fully-autonomous physical interaction with such objects. Along the way, we also addressed the routing problem for wires and cables from a new angle, improving the performance compared to the existing planning methods, and studied the feasibility of aerial manipulation of these objects from the end-effector's precision and required wrenches perspectives.

A summary of the contributions is as follows:

1. A novel controller design is proposed that can extend existing flight controllers to work with the fully-actuated robots. It requires no change to the systems working with the controller and thus provides a short integration timeline with the existing flight stack. A novel set of attitude and thrust strategies and an extension to motion-force control are provided to allow the full actuation of UAVs for indoor and outdoor environments and free flight and physical interaction tasks. See Chapter 3.
2. A set of *real-time* wrench set (dynamic manipulability polytopes) estimation methods for calculating the available instantaneous wrenches of robots are proposed. A novel method is proposed to estimate the available wrenches during controlled physical interaction for robotics manipulators. See Chapter 4.
3. The benefits of the proposed real-time wrench-set estimation methods are showcased by several applications, including: improving the performance

and flexibility of the UAV control allocation module; computing the optimal UAV attitude when an external force is exerted during the physical interaction or free flight; improving the planning of physical interaction tasks. See Chapter 5.

4. A novel method for detecting deformable one-dimensional objects (e.g., wires and cables) is proposed to enable autonomous physical interaction and manipulation of such objects. Additionally, a novel approach to spatial representation and routing of deformable one-dimensional objects is proposed that allows significant performance improvement for tasks involving routing on component boards, such as cable boxes being inspected on the utility poles. The first analysis to enable the physical interaction of aerial robots with these deformable objects is performed. See Chapters 6 and 7.

## 1.4 Outline

The outline of this dissertation is as follows:

Chapter 2 introduces the relevant concepts and the background for the rest of this document. It provides an introduction to fully-actuated multirotors and derives a controller system from the equations of motion of these robots.

Chapter 3 presents our control design for fully-actuated multirotors to address the current challenges facing the integration of these robots into real-world applications. Then the design is further extended to allow controlled physical interaction with the environment.

Chapter 4 introduces the real-time wrench set estimation methods for fully-actuated multirotors and discusses the different extensions of the method to controlled interaction with the physical world.

Chapter 5 illustrates various applications of the real-time wrench set estimation methods to enhance further the use of the fully-actuated multirotors in real-world applications.

Chapter 6 presents the novel detection method for deformable one-dimensional objects capable of working with occlusions and an imperfect segmentation, which enables fully-autonomous physical interaction with objects such as cables and wires using ground or aerial manipulators.

Chapter 7 proposes a novel approach to spatial representation and routing for deformable one-dimensional objects for manipulation in workspaces such as cable boxes in autonomous tasks such as utility pole manipulation and inspection. It further explores the feasibility of such manipulation tasks from different perspectives for aerial robots.

Finally, Chapter 8 provides a summary of the work and outlines the possible further research.

## 1.5 Bibliographical Remarks

This thesis only contains the work and research where this author is the primary contributor:

- Chapter 3 is the work done with Sebastian Scherer, Mohammadreza Mousaei, and Junyi Geng, with contributions from John Keller, Andrew Saba, Andrew Ashley, Greg Armstrong, Dongwei (Saeed) Bai, and Near Earth Autonomy. Some of the work has appeared in [82].
- Chapters 4 and 5 are based on the work with Sebastian Scherer with contributions from Mohammadreza Mousaei and helpful insights from Oliver Kroemer.
- Chapters 6 and 7 are based on the work with Maryam Bandari and Stefan Schaal with helpful insights from Kostas Alexis. The work has appeared in [87], [86] and [88].

## 1.6 Excluded Research

The author has excluded a significant portion of his graduate work and publications to keep this thesis focused. The excluded works are listed below:

1. Automatic Real-time Anomaly Detection for Autonomous Aerial Vehicles, that appeared in [81].
2. ALFA: A Dataset for UAV Fault and Anomaly Detection, that appeared in [83].
3. Path Planning for Unmanned Fixed-Wing Aircraft in Uncertain Wind Conditions Using Trochoids, that appeared in [167].
4. Real-Time Ellipse Detection for Robotics Applications, that appeared in [85].
5. Visual Servoing Approach for Autonomous UAV Landing on a Moving Vehicle, that appeared in [84].
6. Design, Modeling and Control for a Tilt-rotor VTOL UAV in the Presence of Actuator Failure, that appeared in [124].
7. VTOL Failure Detection and Recovery by Utilizing Redundancy, that appeared in [125].
8. Trajectory Planning for a UAV Wrench Task Considering Vehicle Dynamics and Force Output Capabilities, that appeared in [6].

Chapter 2

# Background: Fully-Actuated Multirotors and Controllers

This chapter introduces the preliminary definitions and terminology that will set up the infrastructure to discuss the content in the rest of this document.

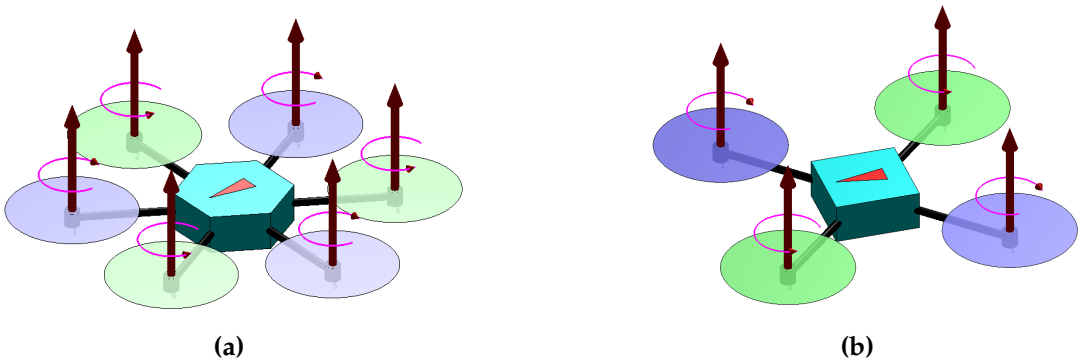
Section 2.1 discusses the related work and provides an introduction to the rest of the chapter. Section 2.2 explains the fully-actuated multirotor model and the assumptions used in this work. Sections 2.3, 2.4 and 2.5 describe the kinematics and dynamics of fully-actuated multirotors. Section 2.6 converts the kinematics and dynamics models into a control-affine model. Finally, Section 2.7 uses the control-affine model to design a controller for fully-actuated multirotors.

## 2.1 Introduction and Related Work

In traditional multirotor designs (e.g., quadrotors), the thrusts generated from all rotors are in the same direction, which is the opposite of the gravity vector in the default attitude (see Figure 2.1). Therefore, the total generated thrust in these robots (called *fixed co-planar* multirotors) is the scalar sum of all the

individual rotor thrusts (ignoring some minor aerodynamic effects). This co-planar rotor arrangement makes this design very efficient in compensating for gravity and optimizing energy consumption to generate motion.

However, fixed co-planar multirotor designs can only generate thrust in a single direction with respect to their body, requiring them to tilt the whole body towards the direction of the desired motion to generate the thrust for the desired acceleration and compensate for gravity. Therefore, these designs cannot fly at a given desired attitude, and their attitude depends on their motion, making them underactuated.



(a) (b)

**Figure 2.1:** In multirotors with fixed co-planar rotors, all the rotors generate thrust in the same direction, and the whole multirotor needs to tilt to generate the desired thrust vector and counteract gravity. (a) A planar hexarotor design. (b) A quadrotor design.

The fixed co-planar multirotors are suitable for many real-world applications, as the precise control of the attitude is not required. Many other applications, such as mapping and cinematography, have found workarounds for the under-actuation by adding payloads such as gimbals to allow controlling the camera's attitude separately from the multirotor's attitude [21]. However, good gimbals are expensive and heavy. Moreover, the devices attached to the gimbal have to be controlled to some degree separately from the robot itself [84].

As another solution, for physical interactions with the world around the robot, many researchers have been adding manipulator arms with at least two degrees of freedom to achieve the end-effector's complete position and

attitude control [22, 157]. The same issues with the gimbal also apply here, i.e., the manipulator arm adds the payload weight, increases the robot's cost, and reduces the system's stability. Besides, the control solutions to physical interaction problems are generally very complex and sub-optimal [208].

On the other hand, there are applications where controlling the attitude of the whole robot is required, and there are no good solutions for the problems caused by underactuation. A practical example is a multirotor trying to navigate a disaster site with a very cluttered environment, and some passes narrower than the robot's width.

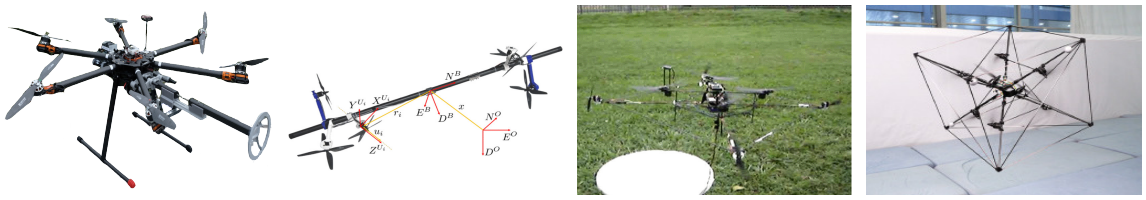
In recent years, many new designs have been proposed to eliminate the underactuation in multirotors. These designs either control the direction of rotors during the flight (known as *variable-pitch designs*) or set fixed non-zero rotations for the rotors in a way that they are not collinear anymore (known as *fixed-pitch designs*) [46, 151].

The fixed-pitch designs can be made from the existing co-planar multirotors by tilting their rotor arms in different directions, making them easy to construct. To achieve fully independent control over the orientation and position, a robot needs to have at least six rotors, and the arrangement of the rotors should make the allocation matrix full-rank. These designs are classified as *fully-actuated multirotors* as they have full actuation around the hovering point.

An issue with fixed-pitch designs is the inefficiency due to the different rotor thrusts' directions, causing some portion of the generated forces by different rotors to be used for counteracting each other. Besides, the aerodynamic effects caused by the rotors slightly pointing towards each other further reduce the system's overall efficiency. Finally, these designs can only provide a limited amount of thrust parallel to the ground plane, limiting the maximum horizontal acceleration and the maximum feasible attitude.

Figure 2.2 shows some fixed-pitch architectures proposed in the literature.

The variable-pitch designs add additional servo motors to control the rotors' direction during the flight. Depending on the design, each servo is responsible



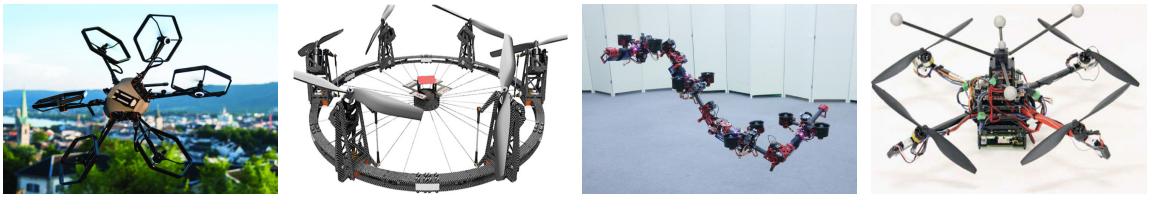
(a) (b) (c) (d)

**Figure 2.2:** Some examples of fixed-pitch fully-actuated multirotor designs in the literature. (a) [82] (b) [140] (c) [154] (d) [24].

for controlling the direction of all rotors together, some of them or just a single rotor. The total number of rotors and servos needs to be at least six for these multirotors to achieve total control over both the orientation and the position. Some of these designs keep the rotors collinear to maximize the efficiency of the generated thrust and simplify the design.

While most of the variable-pitch designs are fully-actuated, some are *omni-directional*, meaning that they can achieve any desired position and orientation. In addition to the extra servo motors causing a lower flight time, the main issue with the variable-pitch designs is the complexity of both the hardware and the controller for these multirotors, making them expensive and requiring extensive hardware and controller debugging and tuning efforts.

Figure 2.3 shows some sample variable-pitch multirotor designs proposed in the literature.



(a) (b) (c) (d)

**Figure 2.3:** Some examples of variable-pitch fully-actuated multirotor designs in the literature. (a) [77] (b) [159] (c) [211] (d) [158].

## 2.2 Modeling and Structure

We use the fixed-pitch fully-actuated designs in the current dissertation work, where the rotors can have fixed arbitrary orientations for different tasks. Figure 2.4 shows a hexarotor design we have been using for some of our experiments. The fixed-pitch design allows us to develop faster and focus better on the physical interaction challenges. Furthermore, all the methods in this work can also be extended to most of the other fully-actuated and omni-directional robot designs. In contrast, the opposite may not be possible, meaning that the methods developed for more complex omni-directional robots may not be able to get easily adapted to the other types of fully-actuated multirotors. Finally, in many variable-pitch designs, changing the rotor directions is too slow for a suitable reaction to the external disturbances in the applications targeted in this work, and the airflow between the lateral and vertical rotors creates nonlinear dynamics, which is hard to deal with [145].

(a) (b)

**Figure 2.4:** A fully-actuated fixed-pitch hexarotor design used in this work. The shown robot has all its rotors tilted 30 degrees to the side, whereas every other rotor is tilted to the opposite side.

The fixed-pitch model described in this chapter uses a set of assumptions described below. In deriving our fixed-pitch model, we have got our inspiration from [112] and [145]. The complete set of the symbols and notations used in

this document can be found in Appendix A.

### 2.2.1 Assumptions

We have made some assumptions to simplify the robot's modeling and control in this work. Many assumptions only affect the low-level controller and the control allocation matrix and do not affect the concepts proposed in other sections. Additionally, some of these assumptions will be lifted later for physical interaction. Therefore, this thesis's work remains valid even with different assumptions for the underlying low-level controller. In the development of the fixed-pitch fully-actuated multirotor, we assume that:

- The multirotor is a rigid body.
- The geometric center of the multirotor coincides with its center of mass (CoM).
- The geometric center of each rotor is the center of spinning and coincides with its actuating motor's center of mass (CoM).
- Earth is a perfect homogeneous sphere, so the gravitational vector is directed precisely towards its center.
- The flight time frames and the distances are small enough to assume the Earth as flat and non-rotating. This assumption allows using the North-East-Down (NED) reference frame as an inertial reference frame.
- The actuating motors accept rotor angular speeds as inputs (this can be achieved by another low-level controller implemented for the rotors).
- The actuating motors have an almost ideal (instantaneous) reaction to the given command with negligible transient time.
- The gyroscopic and inertial effects caused by the motors and propellers are small enough to be ignored.
- The blade flapping and the rotor-induced drag reactions can be ignored.
- The aerodynamic effects resulting from the tilted propellers are small enough to be ignored.

- The aerodynamic effects such as the wall, ceiling, and ground effects can be ignored.
- The rotors' arrangement results in a full-ranked allocation matrix (naturally resulting that the robot must have at least six rotors).

## 2.3 Translational Kinematics

Using the assumptions stated in Section 2.2.1, defining three sets of reference frames is enough for modeling the kinematics of a fixed-pitch fully-actuated multirotor: an inertial frame $\mathcal{F}^{\mathcal{I}}$, a body-fixed frame $\mathcal{F}^{\mathcal{B}}$ and the rotor references $\mathcal{F}^{\mathcal{R}_i}$ $(i = 1, 2, \ldots, n_r)$, where $n_r$ is the number of rotors in the multirotor. This section defines these reference frames and provides the necessary transformations between them.

### 2.3.1 Inertial Frame

An inertial frame is a fixed frame defined as $\mathcal{F}^{\mathcal{I}} = \{O_{\mathcal{I}}, \hat{X}_{\mathcal{I}}, \hat{Y}_{\mathcal{I}}, \hat{Z}_{\mathcal{I}}\}$, where $O_{\mathcal{I}}$ is an arbitrarily chosen origin, and $\hat{X}_{\mathcal{I}}$, $\hat{Y}_{\mathcal{I}}$, $\hat{Z}_{\mathcal{I}}$ are the three right-handed orthogonal axes. For simplicity, using the flat and non-rotating Earth assumptions, we can define the origin as a local point on the ground (e.g., the take-off point). Additionally, it is common to choose the axes in a way that the $\hat{X}_{\mathcal{I}}$ axis aligns with the North direction, the $\hat{Y}_{\mathcal{I}}$ axis aligns with the east direction, and the $\hat{Z}_{\mathcal{I}}$ axis is orthogonal to the other axes and points down (parallel to the gravity vector, towards the center of Earth). This reference frame is commonly referred to as the *North-East-Down* (*NED*) frame.

### 2.3.2 Body-Fixed Frame

The body-fixed frame is a frame fixed to the vehicle and is defined as $\mathcal{F}^{\mathcal{B}} = \{O_{\mathcal{B}}, \hat{X}_{\mathcal{B}}, \hat{Y}_{\mathcal{B}}, \hat{Z}_{\mathcal{B}}\}$. $O_{\mathcal{B}}$ is the frame's origin, which coincides with the center of mass (CoM). The $\hat{X}_{\mathcal{B}}$ axis points to the front of the vehicle, $\hat{Y}_{\mathcal{B}}$ points to the

right side, and $\hat{Z}_{\mathcal{B}}$ is in the direction of the bottom of the UAV, perpendicular to the other two axes. The origin $O_{\mathcal{B}}$ in $\mathcal{F}^{\mathcal{I}}$ is called the *position* of the vehicle and can be defined as $\vec{p}^{\mathcal{I}} = \begin{bmatrix} x & y & z \end{bmatrix}^\top \in \mathbb{R}^{3\times 1}$.

The rotation from $\mathcal{F}^{\mathcal{I}}$ to $\mathcal{F}^{\mathcal{B}}$ is performed in three steps: a rotation of $\psi$ (known as *yaw*) around the $\hat{Z}_{\mathcal{I}}$ axis, then a rotation of $\theta$ (known as *pitch*) around the resulting $\hat{Y}_{\mathcal{B}}$ axis and finally a rotation of $\phi$ (known as *roll*) around the resulting $\hat{X}_{\mathcal{B}}$ axis. This sequence of rotations is commonly known as $\hat{Z}-\hat{Y}'-\hat{X}''$ or "$3-2-1$" rotation sequence and the resulting roll, pitch and yaw angles are known as *Tait-Bryan* or *Euler-Cardan* angles. We call $\mathbf{\Phi} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top$ as the *attitude* of the multirotor.

Using $\mathrm{c}(\cdot)$ as $\cos(\cdot)$ and $\mathrm{s}(\cdot)$ as $\sin(\cdot)$, the rotation matrix $\mathbf{R}_{\mathcal{BI}} \in SO(3)$ from $\mathcal{F}^{\mathcal{I}}$ to $\mathcal{F}^{\mathcal{B}}$ can be calculated from the above sequence as:

$$\mathbf{R}_{\mathcal{BI}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\,\phi & \mathrm{s}\,\phi \\ 0 & -\mathrm{s}\,\phi & \mathrm{c}\,\phi \end{bmatrix} \begin{bmatrix} \mathrm{c}\,\theta & 0 & -\mathrm{s}\,\theta \\ 0 & 1 & 0 \\ \mathrm{s}\,\theta & 0 & \mathrm{c}\,\theta \end{bmatrix} \begin{bmatrix} \mathrm{c}\,\psi & \mathrm{s}\,\psi & 0 \\ -\mathrm{s}\,\psi & \mathrm{c}\,\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

Simplifying Equation 2.1, we will have:

$$\mathbf{R}_{\mathcal{IB}} = (\mathbf{R}_{\mathcal{BI}})^\top = \begin{bmatrix} \mathrm{c}\,\theta\,\mathrm{c}\,\psi & \mathrm{s}\,\phi\,\mathrm{s}\,\theta\,\mathrm{c}\,\psi - \mathrm{c}\,\phi\,\mathrm{s}\,\psi & \mathrm{c}\,\phi\,\mathrm{s}\,\theta\,\mathrm{c}\,\psi + \mathrm{s}\,\phi\,\mathrm{s}\,\psi \\ \mathrm{c}\,\theta\,\mathrm{s}\,\psi & \mathrm{s}\,\phi\,\mathrm{s}\,\theta\,\mathrm{s}\,\psi + \mathrm{c}\,\phi\,\mathrm{c}\,\psi & \mathrm{c}\,\phi\,\mathrm{s}\,\theta\,\mathrm{s}\,\psi - \mathrm{s}\,\phi\,\mathrm{c}\,\psi \\ -\mathrm{s}\,\theta & \mathrm{s}\,\phi\,\mathrm{c}\,\theta & \mathrm{c}\,\phi\,\mathrm{c}\,\theta \end{bmatrix}. \tag{2.2}$$

We define the range of the yaw angle $\psi$ to be $-\pi < \psi \leq \pi$. In order to avoid a singularity, both pitch and roll angles are limited to the angles where the UAV is tilted less than $90°$, therefore we have $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$ and $-\frac{\pi}{2} < \phi < \frac{\pi}{2}$. Note that we can avoid this singularity by using other rotation representations (e.g., quaternions) [40]. However, there is no practical situation in this project where this singularity can happen, and the Euler angles representation can be

devised safely.

### 2.3.3 Rotor Frames

The rotor frames are the frames fixed to the rotors and are defined as $\mathcal{F}^{\mathcal{R}_i} = \{O_{\mathcal{R}_i}, \hat{X}_{\mathcal{R}_i}, \hat{Y}_{\mathcal{R}_i}, \hat{Z}_{\mathcal{R}_i}\}$ $(i = 1, 2, \ldots, n_r)$. $O_{\mathcal{R}_i}$ is the center of spinning of the $i^{\text{th}}$ rotor. The $\hat{Y}_{\mathcal{R}_i}$ axis points towards the vehicle's CoM (from $O_{\mathcal{R}_i}$ to $O_{\mathcal{B}}$), $\hat{Z}_{\mathcal{R}_i}$ is aligned with the axis of rotation of rotor $i$ pointing to the vehicle's bottom direction, and $\hat{X}_{\mathcal{R}_i}$ is orthogonal to both and its direction can be obtained from the right-hand rule. The vector from $O_{\mathcal{B}}$ to $O_{\mathcal{R}_i}$ in $\mathcal{F}^{\mathcal{B}}$ can be defined as $\vec{r}_i^{\mathcal{B}} = \begin{bmatrix} r_{i_x} & r_{i_y} & r_{i_z} \end{bmatrix}^\top \in \mathbb{R}^{3\times 1}$ $(i = 1, 2, \ldots, n_r)$.

Figure 2.5 illustrates the frames defined for a fixed-pitch multirotor.

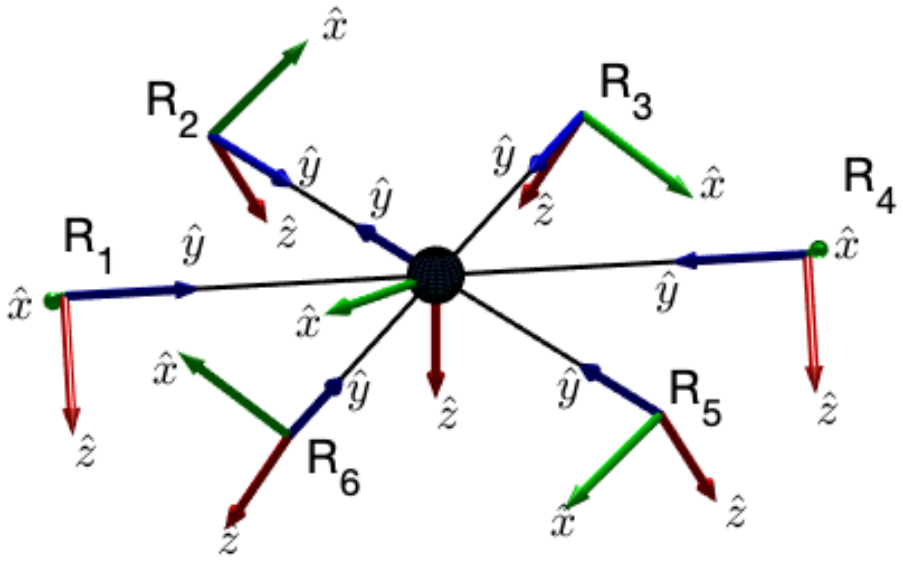



**Figure 2.5:** An illustration of the body-fixed and rotor frames for a fixed-pitch fully-actuated hexarotor.

Assuming that the origin of each rotor $i$ in the body-fixed frame is shown as ${}_{\mathcal{B}}O_{\mathcal{R}_i}$, the distance $\ell_i$ of each rotor from the vehicle's center of mass can be calculated as $\ell_i = |{}_{\mathcal{B}}O_{\mathcal{R}_i}| = |\vec{r}_i|$. Let us call the vectors from the center of the

vehicle to the origin of rotor $i$ as $\vec{r}_i$. Projecting these vectors onto the $\hat{X}_\mathcal{B}\hat{Y}_\mathcal{B}$ plane, the angle between each rotor $i$ with the next rotor on the $\hat{X}_\mathcal{B}\hat{Y}_\mathcal{B}$ plane in a clockwise direction around $\hat{Z}_\mathcal{B}$ (which is pointing down) can be named $\alpha_i$. For example, the angle between rotors $1$ and $2$ will be called $\alpha_1$. In addition, the angle between the projection of $\vec{r}_i$ on $\hat{X}_\mathcal{B}\hat{Y}_\mathcal{B}$ plane and $\vec{r}_i$ defines the *dihedral* angle of the rotor $i$'s vector and is named $\phi_{dih_i}$.

The rotation from $\mathcal{F}^\mathcal{B}$ to $\mathcal{F}^{\mathcal{R}_i}$ has one additional step compared to the $\mathbf{R}_{\mathcal{BI}}$ rotation and is performed using a sequence of four rotations: a $90°$ rotation around the $\hat{Z}_\mathcal{B}$ axis in the positive direction so that an imaginary rotor's frame in front of the vehicle on the $\hat{X}_\mathcal{B}\hat{Y}_\mathcal{B}$ plane would have its $Y$ axis pointing towards the vehicle's CoM, next rotation of $\mu_i$ around the $\hat{Z}_\mathcal{B}$ axis to align the $i^{\text{th}}$ rotor's frame to have the projection of its $\hat{Y}$ axis on the $\hat{X}_\mathcal{B}\hat{Y}_\mathcal{B}$ plane pointing towards the vehicle's CoM, then a rotation of $\phi_{xi}$ (called the *inward angle*) around the $\hat{X}$ axis of the new frame, and finally, a rotation of $\phi_{yi}$ around the new $\hat{Y}$ axis (called the *sideward angle*).

Using $\mathrm{c}(\cdot)$ as $\cos(\cdot)$ and $\mathrm{s}(\cdot)$ as $\sin(\cdot)$, the rotation matrix $\mathbf{R}_{\mathcal{R}_i\mathcal{B}} \in SO(3)$ from $\mathcal{F}^\mathcal{B}$ to $\mathcal{F}^{\mathcal{R}_i}$ can be calculated from the above sequence as:

$$\mathbf{R}_{\mathcal{R}_i\mathcal{B}} = \begin{bmatrix} \mathrm{c}\,\phi_{yi} & 0 & -\mathrm{s}\,\phi_{yi} \\ 0 & 1 & 0 \\ \mathrm{s}\,\phi_{yi} & 0 & \mathrm{c}\,\phi_{yi} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\,\phi_{xi} & \mathrm{s}\,\phi_{xi} \\ 0 & -\mathrm{s}\,\phi_{xi} & \mathrm{c}\,\phi_{xi} \end{bmatrix} \begin{bmatrix} \mathrm{c}\,\mu_i & \mathrm{s}\,\mu_i & 0 \\ -\mathrm{s}\,\mu_i & \mathrm{c}\,\mu_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Simplifying Equation 2.3 and considering that $\mathbf{R}_{\mathcal{B}\mathcal{R}_i} = (\mathbf{R}_{\mathcal{R}_i\mathcal{B}})^\top$, we will have:

$$\mathbf{R}_{\mathcal{B}\mathcal{R}_i} = \begin{bmatrix} -\mathrm{s}\,\mu_i\,\mathrm{c}\,\phi_{yi} - \mathrm{c}\,\mu_i\,\mathrm{s}\,\phi_{xi}\,\mathrm{s}\,\phi_{yi} & -\mathrm{c}\,\mu_i\,\mathrm{c}\,\phi_{xi} & \mathrm{c}\,\mu_i\,\mathrm{s}\,\phi_{xi}\,\mathrm{c}\,\phi_{yi} - \mathrm{s}\,\mu_i\,\mathrm{s}\,\phi_{yi} \\ \mathrm{c}\,\mu_i\,\mathrm{c}\,\phi_{yi} - \mathrm{s}\,\mu_i\,\mathrm{s}\,\phi_{xi}\,\mathrm{s}\,\phi_{yi} & -\mathrm{s}\,\mu_i\,\mathrm{c}\,\phi_{xi} & \mathrm{c}\,\mu_i\,\mathrm{s}\,\phi_{yi} + \mathrm{s}\,\mu_i\,\mathrm{s}\,\phi_{xi}\,\mathrm{c}\,\phi_{yi} \\ -\mathrm{c}\,\phi_{xi}\,\mathrm{s}\,\phi_{yi} & \mathrm{s}\,\phi_{xi} & \mathrm{c}\,\phi_{xi}\,\mathrm{c}\,\phi_{yi} \end{bmatrix}. \tag{2.4}$$

## 2.4 Rotational Kinematics

The angular velocity of the multirotor $\omega = \begin{bmatrix} p & q & r \end{bmatrix}^\top$ is defined as the angular rate in the body-fixed frame. Therefore, $p$, $q$ and $r$ are the angular rates around the $\hat{X}_\mathcal{B}$, $\hat{Y}_\mathcal{B}$ and $\hat{Z}_\mathcal{B}$ axes, respectively. As defined in Section 2.3, the roll angle $\phi$, the pitch angle $\theta$ and the yaw angle $\psi$ are defined in different frames than $p$, $q$ and $r$. The relationship between these variables can be obtained from the rotations between their respective frames [13]. Starting with $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ angular rates, and using $\mathrm{c}(\cdot)$ as $\cos(\cdot)$ and $\mathrm{s}(\cdot)$ as $\sin(\cdot)$, we have:

$$\begin{aligned}
\begin{bmatrix} p \\ q \\ r \end{bmatrix} &= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\phi & \mathrm{s}\phi \\ 0 & -\mathrm{s}\phi & \mathrm{c}\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\phi & \mathrm{s}\phi \\ 0 & -\mathrm{s}\phi & \mathrm{c}\phi \end{bmatrix} \begin{bmatrix} \mathrm{c}\theta & 0 & -\mathrm{s}\theta \\ 0 & 1 & 0 \\ \mathrm{s}\theta & 0 & \mathrm{c}\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & -\mathrm{s}\theta \\ 0 & \mathrm{c}\phi & \mathrm{s}\phi\,\mathrm{c}\theta \\ 0 & -\mathrm{s}\phi & \mathrm{c}\phi\,\mathrm{c}\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}
\end{aligned} \tag{2.5}$$

From inverting Equation 2.5 we get:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\mathrm{s}\theta \\ 0 & \mathrm{c}\phi & \mathrm{s}\phi\,\mathrm{c}\theta \\ 0 & -\mathrm{s}\phi & \mathrm{c}\phi\,\mathrm{c}\theta \end{bmatrix}^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & -\sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.6}$$

## 2.5 Dynamics

Assuming that the multirotor is a rigid body, we can use the Newton-Euler formalism to derive the equations of motion for the vehicle dynamics:

$$\begin{aligned} \vec{F}^{\mathcal{I}} &= m\ddot{\vec{p}}^{\mathcal{I}} \\ \vec{M}^{\mathcal{B}} &= \mathbf{I}_{\mathcal{B}}\dot{\vec{\omega}} + \vec{\omega} \times \mathbf{I}_{\mathcal{B}}\vec{\omega}, \end{aligned} \tag{2.7}$$

where $\vec{F}^{\mathcal{I}} \in \mathbb{R}^{3\times 1}$ is the total force vector in $[N]$ applied to robot's center of mass, $\vec{M}^{\mathcal{B}} \in \mathbb{R}^{3\times 1}$ is the total moment vector in $[Nm]$, $\omega \in \mathbb{R}^{3\times 1}$ is the body angular velocity vector in $[rad/s]$, $\mathbf{I}_{\mathcal{B}} \in \mathbb{R}^{3\times 3}$ is the body-frame inertia tensor in $[Nms^2]$, and $m$ is the total mass of the vehicle measured in $[kg]$.

This section defines the forces and moments acting on the system and derives a model that can be used to control an omni-directional multirotor.

### 2.5.1 Forces

The two significant forces acting on the vehicle are the *gravity force* and the *thrust force* which is the result of the spinning rotors. Several other less significant forces are applied to the vehicle that can be ignored from the model. For example, the friction between the moving multirotor and air (i.e., *drag force*) is small enough in the low speeds we have in our application and can be considered a disturbance.

Assuming a gravitational acceleration $g$ pointing towards the center of Earth, the gravity force of the vehicle will be in the direction of the $\hat{Z}_{\mathcal{I}}$ axis. Therefore, the total gravity force vector acting on the multirotor can be defined as:

$$\vec{F}^{\mathcal{I}}_{grav} = mg\hat{Z}_{\mathcal{I}} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{2.8}$$

Ideally, the thrust $\vec{F}_{thr_i}$ generated by the $i^{\text{th}}$ rotor is aligned with the negative direction of the $\hat{Z}_{\mathcal{R}_i}$ axis. Ignoring some less significant effects, the magnitude of the generated thrust $F_{thr}$ from a spinning rotor can be approximated as

$F_{thr} = c_F \Omega^2$, where $c_F > 0$ is a rotor-specific thrust constant in $[Ns^2]$ and $\Omega$ is the rotational velocity of the rotor in $[rad/s]$. Therefore, the generated thrust by the $i^{\text{th}}$ rotor in the body-fixed frame can be obtained as:

$$\overrightarrow{F}^{\mathcal{B}}_{thr_i} = \mathbf{R}_{\mathcal{B}\mathcal{R}_i} \left( -F_{thr_i} \hat{Z}_{\mathcal{R}_i} \right) = \mathbf{R}_{\mathcal{B}\mathcal{R}_i} \begin{bmatrix} 0 \\ 0 \\ -c_{Fi}\Omega_i^2 \end{bmatrix} \tag{2.9}$$

Assuming that the rotors are positioned such that the effect of their airflow on each other's thrust is insignificant, from Equations 2.8 and 2.9 we can measure the total force in the inertial frame as:

$$\overrightarrow{F}^{\mathcal{I}} = \overrightarrow{F}^{\mathcal{I}}_{grav} + \mathbf{R}_{\mathcal{I}\mathcal{B}} \sum_{i=1}^{n_r} \overrightarrow{F}^{\mathcal{B}}_{thr_i} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \mathbf{R}_{\mathcal{I}\mathcal{B}} \sum_{i=1}^{n_r} \mathbf{R}_{\mathcal{B}\mathcal{R}_i} \begin{bmatrix} 0 \\ 0 \\ -c_{Fi}\Omega_i^2 \end{bmatrix} \tag{2.10}$$

### 2.5.2 Moments

The main moments affecting the vehicle are the thrust moment $\overrightarrow{M}_{thr}$ acting on the vehicle's CoM (origin) resulting from the rotor thrusts, the reaction moment $\overrightarrow{M}_{reac}$ of a spinning rotor acting on the rotor's CoM (origin), and the moment $\overrightarrow{M}_{grav}$ acting on the vehicle's CoM resulting from the weights of the individual parts such as legs and rotors. Besides, there are other less significant moments acting on the multirotor, which can be ignored and considered disturbances for the model's simplicity. These moments include the moment resulting from air friction and the drag force acting on the vehicle, the gyroscopic moments of the spinning rotors as rotating masses, and the drag torque of each spinning rotor resulting from the rotor's acceleration.

The moment resulting from a rotor's thrust around the vehicle's body-fixed axes can be calculated as $\overrightarrow{M}^{\mathcal{B}}_{thr_{rot}} = \vec{r} \times \overrightarrow{F}^{\mathcal{B}}_{thr}$, where $\vec{r}$ is the moment arm in $[m]$

from the vehicle's CoM ($O_{\mathcal{B}}$) to the center of rotor's spinning ($O_{\mathcal{R}_s}$). Therefore the total moment resulting from the rotor thrusts is:

$$\overrightarrow{M}^{\mathcal{B}}_{thr} = \sum_{i=1}^{n_r} \left( \vec{r}^{\,\mathcal{B}}_{i} \times \overrightarrow{F}^{\mathcal{B}}_{thr_i} \right) \tag{2.11}$$

Another type of moment is the reaction of the multirotor to a spinning rotor, which is applied to the rotor's center of spinning and has the same magnitude but in the opposite direction of the motor's torque. Similar to the thrust force generated by a spinning rotor $\overrightarrow{F}_{thr}$, this reaction moment $\overrightarrow{M}_{reac}$ can also be approximated by a quadratic relationship to the rotor speed using $M_{reac} = (-1)^d c_\tau \Omega^2$, where $d = 0$ if the rotor is spinning in the positive direction of rotor's $\hat{Z}$ axis (i.e., counter-clockwise around the axis) and $d = 1$ if the rotor is spinning in the negative direction of rotor's $\hat{Z}$ axis. Therefore, the reaction moment of the $i^{\text{th}}$ rotor and the total reaction moment in the body-fixed axes can be calculated as:

$$\overrightarrow{M}^{\mathcal{B}}_{reac_i} = \mathbf{R}_{\mathcal{B}\mathcal{R}_i} \left( M_{reac_i} \hat{Z}_{\mathcal{R}_i} \right) = \mathbf{R}_{\mathcal{B}\mathcal{R}_i} \begin{bmatrix} 0 \\ 0 \\ (-1)^{d_i} c_{\tau i} \Omega_i^2 \end{bmatrix} \tag{2.12}$$

$$\overrightarrow{M}^{\mathcal{B}}_{reac} = \sum_{i=1}^{n_r} \overrightarrow{M}^{\mathcal{B}}_{reac_i} = \sum_{i=1}^{n_r} \left( \mathbf{R}_{\mathcal{B}\mathcal{R}_i} \begin{bmatrix} 0 \\ 0 \\ (-1)^{d_i} c_{\tau i} \Omega_i^2 \end{bmatrix} \right) \tag{2.13}$$

Finally, the gravity forces of different parts of the multirotor also create a total moment $\overrightarrow{M}_{grav}$ around the vehicle's CoM. These moments depend on the structure of the multirotor and can be different for each geometry. For the most common structure of multirotors, where the rotors' legs are extending from the CoM to the rotors, and assuming that $\vec{r}_i$ is the vector connecting the CoM ($O_{\mathcal{B}}$) to the $i^{\text{th}}$ rotor's CoM ($O_{\mathcal{R}_i}$) and $\vec{r}_{leg_i}$ is the vector connecting the CoM ($O_{\mathcal{B}}$) to the $i^{\text{th}}$ leg's CoM, we have:

$$\overrightarrow{M}^{\mathcal{B}}_{grav} = \sum_{i=1}^{n_r} \left[ \left( \vec{r}^{\mathcal{B}}_{i} \times \mathbf{R}_{\mathcal{BI}} \begin{bmatrix} 0 \\ 0 \\ m_{rotor_i} g \end{bmatrix} \right) + \left( \vec{r}^{\mathcal{B}}_{leg_i} \times \mathbf{R}_{\mathcal{BI}} \begin{bmatrix} 0 \\ 0 \\ m_{leg_i} g \end{bmatrix} \right) \right] \tag{2.14}$$

From the Equations 2.11, 2.13 and 2.14, the total moment around the body-fixed axes can be calculated as:

$$\overrightarrow{M}^{\mathcal{B}} = \overrightarrow{M}^{\mathcal{B}}_{thr} + \overrightarrow{M}^{\mathcal{B}}_{reac} + \overrightarrow{M}^{\mathcal{B}}_{grav} \tag{2.15}$$

### 2.5.3 Equations of Motion

Let us define the state of the system as $\mathbf{x} = \begin{bmatrix} \dot{\vec{p}}^{\mathcal{I}} & \mathbf{\Phi} & \omega \end{bmatrix}^{\top}$. By replacing Equations 2.10 and 2.15 in Equation 2.7 and by renaming the conversion matrix of Equation 2.6 to $\eta(\mathbf{\Phi})$, we can get the equations of motion as:

$$\begin{aligned} \overrightarrow{F}^{\mathcal{I}} &= \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \mathbf{R}_{\mathcal{IB}} \sum_{i=1}^{n_r} \mathbf{R}_{\mathcal{BR}_i} \begin{bmatrix} 0 \\ 0 \\ -c_{Fi}\Omega_i^2 \end{bmatrix} = m\ddot{\vec{p}}^{\mathcal{I}} \\ \Rightarrow \quad \ddot{\vec{p}}^{\mathcal{I}} &= \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{1}{m} \mathbf{R}_{\mathcal{IB}} \sum_{i=1}^{n_r} \mathbf{R}_{\mathcal{BR}_i} \begin{bmatrix} 0 \\ 0 \\ -c_{Fi}\Omega_i^2 \end{bmatrix} \end{aligned} \tag{2.16}$$

$$\overrightarrow{M}^{\mathcal{B}} = \mathbf{I}_{\mathcal{B}} \dot{\vec{\omega}} + \vec{\omega} \times \mathbf{I}_{\mathcal{B}} \vec{\omega} \Rightarrow \dot{\vec{\omega}} = \mathbf{I}_{\mathcal{B}}^{-1} \left( \overrightarrow{M}^{\mathcal{B}}_{thr} + \overrightarrow{M}^{\mathcal{B}}_{reac} + \overrightarrow{M}^{\mathcal{B}}_{grav} \right) - \mathbf{I}_{\mathcal{B}}^{-1} \left( \vec{\omega} \times \mathbf{I}_{\mathcal{B}} \vec{\omega} \right) \tag{2.17}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \underbrace{\begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & -\sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix}}_{\eta(\mathbf{\Phi})}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \Rightarrow \dot{\mathbf{\Phi}} = \eta(\mathbf{\Phi})\cdot\omega \tag{2.18}$$

## 2.6 Control-Affine Model

If the multirotor accepts the rotor rotational velocities $\Omega$ as a control command, we can define the control input $\mathbf{u}$ as the set of squared rotor speeds $\Omega_i^2$. Then we can rearrange Equations 2.10 and 2.15 to achieve a control-affine formulation for the system. We have:

$$\mathbf{u} = \begin{bmatrix}\Omega_1^2\\ \Omega_2^2\\ \vdots\\ \Omega_{n_r}^2\end{bmatrix} \in \mathbb{R}^{n_r\times 1} \tag{2.19}$$

From Equations 2.4 and 2.10 we have:

$$\begin{aligned}\vec{F}_{thr}^{\mathcal{B}} &= \sum_{i=1}^{n_r}\left(\mathbf{R}_{\mathcal{BR}_i}\begin{bmatrix}0\\ 0\\ -c_{Fi}\Omega_i^2\end{bmatrix}\right) = \sum_{i=1}^{n_r}\left(-c_{Fi}\Omega_i^2\begin{bmatrix}[\mathbf{R}_{\mathcal{BR}_i}]_{13}\\ [\mathbf{R}_{\mathcal{BR}_i}]_{23}\\ [\mathbf{R}_{\mathcal{BR}_i}]_{33}\end{bmatrix}\right)\\ &= \sum_{i=1}^{n_r}\left(-c_{Fi}\Omega_i^2\begin{bmatrix}\mathrm{c}\,\mu_i\,\mathrm{s}\,\phi_{yi} + \mathrm{c}\,\phi_{yi}\,\mathrm{s}\,\mu_i\,\mathrm{s}\,\phi_{xi}\\ \mathrm{s}\,\mu_i\,\mathrm{s}\,\phi_{yi} - \mathrm{c}\,\mu_i\,\mathrm{c}\,\phi_{yi}\,\mathrm{s}\,\phi_{xi}\\ \mathrm{c}\,\phi_{xi}\,\mathrm{c}\,\phi_{yi}\end{bmatrix}\right)\end{aligned} \tag{2.20}$$

Expanding the equation further gives:

$$\begin{aligned}\overrightarrow{F}^{\mathcal{B}}_{thr} &= \underbrace{\begin{bmatrix} -c_{F1}\left[\mathbf{R}_{\mathcal{BR}_1}\right]_{13} & -c_{F2}\left[\mathbf{R}_{\mathcal{BR}_2}\right]_{13} & \cdots & -c_{Fn_r}\left[\mathbf{R}_{\mathcal{BR}_{n_r}}\right]_{13} \\ -c_{F1}\left[\mathbf{R}_{\mathcal{BR}_1}\right]_{23} & -c_{F2}\left[\mathbf{R}_{\mathcal{BR}_2}\right]_{23} & \cdots & -c_{Fn_r}\left[\mathbf{R}_{\mathcal{BR}_{n_r}}\right]_{23} \\ -c_{F1}\left[\mathbf{R}_{\mathcal{BR}_1}\right]_{33} & -c_{F2}\left[\mathbf{R}_{\mathcal{BR}_2}\right]_{33} & \cdots & -c_{Fn_r}\left[\mathbf{R}_{\mathcal{BR}_{n_r}}\right]_{33} \end{bmatrix}}_{\mathbf{L}\in\mathbb{R}^{3\times n_r}} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \vdots \\ \Omega_{n_r}^2 \end{bmatrix} \\ &= \mathbf{L}\cdot\mathbf{u}\end{aligned} \tag{2.21}$$

$$\overrightarrow{F}^{\mathcal{I}} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \mathbf{R}_{\mathcal{IB}}\overrightarrow{F}^{\mathcal{B}}_{thr} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + (\mathbf{R}_{\mathcal{IB}}\mathbf{L})\cdot\mathbf{u} \tag{2.22}$$

Similar to Equation 2.21, from Equation 2.13 we have:

$$\begin{aligned}\overrightarrow{M}^{\mathcal{B}}_{reac} &= \sum_{i=1}^{n_r}\left((-1)^{d_i}c_{\tau i}\Omega_i^2\begin{bmatrix}\left[\mathbf{R}_{\mathcal{BR}_i}\right]_{13} \\ \left[\mathbf{R}_{\mathcal{BR}_i}\right]_{23} \\ \left[\mathbf{R}_{\mathcal{BR}_i}\right]_{33}\end{bmatrix}\right) \\ &= \sum_{i=1}^{n_r}\left((-1)^{d_i}c_{\tau i}\Omega_i^2\begin{bmatrix}\mathrm{c}\,\mu_i\,\mathrm{s}\,\phi_{yi} + \mathrm{c}\,\phi_{yi}\,\mathrm{s}\,\mu_i\,\mathrm{s}\,\phi_{xi} \\ \mathrm{s}\,\mu_i\,\mathrm{s}\,\phi_{yi} - \mathrm{c}\,\mu_i\,\mathrm{c}\,\phi_{yi}\,\mathrm{s}\,\phi_{xi} \\ \mathrm{c}\,\phi_{xi}\,\mathrm{c}\,\phi_{yi}\end{bmatrix}\right) \\ &= \underbrace{\begin{bmatrix}(-1)^{d_1}c_{\tau 1}\left[\mathbf{R}_{\mathcal{BR}_1}\right]_{13} & \cdots & (-1)^{dn_r}c_{\tau n_r}\left[\mathbf{R}_{\mathcal{BR}_{n_r}}\right]_{13} \\ (-1)^{d_1}c_{\tau 1}\left[\mathbf{R}_{\mathcal{BR}_1}\right]_{23} & \cdots & (-1)^{dn_r}c_{\tau n_r}\left[\mathbf{R}_{\mathcal{BR}_{n_r}}\right]_{23} \\ (-1)^{d_1}c_{\tau 1}\left[\mathbf{R}_{\mathcal{BR}_1}\right]_{33} & \cdots & (-1)^{dn_r}c_{\tau n_r}\left[\mathbf{R}_{\mathcal{BR}_{n_r}}\right]_{33}\end{bmatrix}}_{\mathbf{G}\in\mathbb{R}^{3\times n_r}}\begin{bmatrix}\Omega_1^2 \\ \Omega_2^2 \\ \vdots \\ \Omega_{n_r}^2\end{bmatrix} = \mathbf{G}\cdot\mathbf{u}\end{aligned} \tag{2.23}$$

Rearranging the Equation 2.11, we have:

$$\overrightarrow{M}^{\mathcal{B}}_{thr} = \sum_{i=1}^{n_r} \left( \vec{r}^{\mathcal{B}}_i \times \overrightarrow{F}^{\mathcal{B}}_{thr_i} \right) = \sum_{i=1}^{n_r} \left( \left( \underbrace{\vec{r}^{\mathcal{B}}_i \times \begin{bmatrix} -c_{Fi} \left[\mathbf{R}_{\mathcal{B}\mathcal{R}_i}\right]_{13} \\ -c_{Fi} \left[\mathbf{R}_{\mathcal{B}\mathcal{R}_i}\right]_{23} \\ -c_{Fi} \left[\mathbf{R}_{\mathcal{B}\mathcal{R}_i}\right]_{33} \end{bmatrix}}_{\mathbf{F}_i \in \mathbb{R}^{3\times 1}} \right) \cdot \Omega_i^2 \right)$$

$$= \underbrace{\begin{bmatrix} \mathbf{F}_1 & \cdots & \mathbf{F}_{n_r} \end{bmatrix}}_{\mathbf{F} \in \mathbb{R}^{3\times n_r}} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \vdots \\ \Omega_{n_r}^2 \end{bmatrix} = \mathbf{F} \cdot \mathbf{u} \tag{2.24}$$

Finally, by replacing Equations 2.23 and 2.24 in Equation 2.15, we have:

$$\overrightarrow{M}^{\mathcal{B}} = \overrightarrow{M}^{\mathcal{B}}_{grav} + \underbrace{(\mathbf{F} + \mathbf{G})}_{\mathbf{M}} \cdot \mathbf{u} = \overrightarrow{M}^{\mathcal{B}}_{grav} + \mathbf{M} \cdot \mathbf{u} \tag{2.25}$$

### 2.6.1 Control-Affine Equations of Motion

From Section 2.5 we have the state of the system defined as $\mathbf{x} = \begin{bmatrix} \dot{\vec{p}}^{\mathcal{I}} & \mathbf{\Phi} & \omega \end{bmatrix}^\top$. From the definition of input command $\mathbf{u}$ in Equation 2.19, and by replacing the control-affine force and moment from Equations 2.22 and 2.25 into equations of motion (Equations 2.16, 2.17 and 2.18, we can obtain the systems dynamics $\dot{\mathbf{x}}$:

$$\underbrace{\begin{bmatrix} \ddot{\mathbf{p}}^{\mathcal{I}} \\ \dot{\mathbf{\Phi}} \\ \dot{\omega} \end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} \begin{bmatrix} 0 & 0 & g \end{bmatrix}^\top \\ \eta(\mathbf{\Phi}) \cdot \omega \\ \mathbf{I}_{\mathcal{B}}^{-1} \left( \mathbf{M}^{\mathcal{B}}_{grav} - (\omega \times \mathbf{I}_{\mathcal{B}} \omega) \right) \end{bmatrix}}_{\mathbf{f}(\mathbf{x})} + \underbrace{\begin{bmatrix} \frac{1}{m} \mathbf{R}_{\mathcal{I}\mathcal{B}} \mathbf{L} \\ \mathbf{0}_{3\times n_r} \\ \mathbf{I}_{\mathcal{B}}^{-1} \mathbf{M} \end{bmatrix}}_{\mathbf{J}(\mathbf{x})} \underbrace{\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \vdots \\ \Omega_{n_r}^2 \end{bmatrix}}_{\mathbf{u}} \tag{2.26}$$

which can be simplified to:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x}) \cdot \mathbf{u}, \tag{2.27}$$

where $\mathbf{f}(\mathbf{x})$ is the *drift vector* due to the gravity and rotational inertia, and $\mathbf{J}(\mathbf{x})$ is the *decoupling matrix* mapping the input of the system to the state space.

## 2.7 Controller Design

Over the years, many different control methods have been proposed for the conventional underactuated multirotors to control different parameters, such as position, attitude, velocity, accelerations, or precisely track the given trajectories in free flight. Comprehensive reviews of these methods have been published in [73, 75, 170, 200, 215].

Many of the methods written initially for underactuated multirotors are already adopted for fully-actuated fixed-pitch and variable-pitch robots. However, due to a broad spectrum of different designs in this category of multirotors, we will only discuss the works that have been proposed or can be easily adapted for the fixed-pitch designs that are the focus of our work.

The most common type of controller in the literature is the *exact feedback linearization and decoupling method* proposed by [145] which has also been called *nonlinear dynamic inversion* by some authors [112]. This controller aims to minimize the control effort, and its optimal design parameters depend on the trajectory. The resulting system is linear and suitable for extension to physical interaction applications, which is why we also considered this controller.

Assuming that the goal of our controller is to track the position and the given Euler angles, we can define the output of the system as:

$$\mathbf{y} = \begin{bmatrix} \mathbf{p}^{\mathcal{I}} \\ \mathbf{\Phi} \end{bmatrix} \tag{2.28}$$

By differentiating the output $\mathbf{y}$ from Equation 2.28 twice and using the system dynamics equation (Equation 2.26), we can find a relationship between the system input $\mathbf{u}$ and the output $\mathbf{y}$ as:

$$\ddot{\mathbf{y}} = \begin{bmatrix} \ddot{\mathbf{p}}^{\mathcal{I}} \\ \ddot{\mathbf{\Phi}} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & 0 & g \end{bmatrix}^\top \\ \dot{\eta}(\mathbf{\Phi})\omega + \eta(\mathbf{\Phi})\mathbf{I}_{\mathcal{B}}^{-1}\left(\mathbf{M}_{grav}^{\mathcal{B}} - (\omega \times \mathbf{I}_{\mathcal{B}}\omega)\right) \end{bmatrix} + \begin{bmatrix} \frac{1}{m}\mathbf{R}_{\mathcal{I}\mathcal{B}}\mathbf{L} \\ \eta(\mathbf{\Phi})\mathbf{I}_{\mathcal{B}}^{-1}\mathbf{M} \end{bmatrix} \cdot \mathbf{u} \tag{2.29}$$

From Equation 2.29 we can calculate the required input to the system (the controller output) for the desired system output's second derivative (the controller input) as:

$$\mathbf{u} = \begin{bmatrix} \frac{1}{m}\mathbf{R}_{\mathcal{I}\mathcal{B}}\mathbf{L} \\ \eta(\mathbf{\Phi})\mathbf{I}_{\mathcal{B}}^{-1}\mathbf{M} \end{bmatrix}^{-1} \left( \ddot{\mathbf{y}} - \begin{bmatrix} \begin{bmatrix} 0 & 0 & g \end{bmatrix}^\top \\ \dot{\eta}(\mathbf{\Phi})\omega + \eta(\mathbf{\Phi})\mathbf{I}_{\mathcal{B}}^{-1}\left(\mathbf{M}_{grav}^{\mathcal{B}} - (\omega \times \mathbf{I}_{\mathcal{B}}\omega)\right) \end{bmatrix} \right) \tag{2.30}$$

Having a controller that can control the position and attitude accelerations, we can use any stabilizing outer controller to control the desired positions and attitudes. The most common is a proportional–integral–derivative (PID) controller used by [112, 146], and many other researchers. PID has shown to perform well for this task, and its implementation is already available in most traditional controllers, allowing an easier transition to the fully-actuated robots.

The proposed feedback linearization control method assumes that the input $\ddot{\mathbf{y}}$ can have any value to track a completely 6-DoF independent trajectory. Hence, this controller's limitation is that it does not account for the input saturation, which can result in motion instability when the required inputs for perfect trajectory tracking are not feasible [16]. [33] has discussed a solution to

this problem by introducing a suitable scheme for systems with linear dynamics and nonlinear state and input constraints. [47] specifically takes the saturation into account and addresses the control for fully-actuated vehicles with bounded lateral forces such as the fixed-pitch hexarotor designs we use in this work.

Some other controller methods proposed for the fully-actuated vehicles include the Sliding Mode Control [4, 130, 204], Nonlinear Model-Predictive Control [181], and Model Reference Adaptive Control [171].

Chapter 3

# Flexible Control Design for Fully-Actuated Multirotors

The introduction of fully-actuated multirotors has opened the door to new possibilities and more efficient solutions to many real-world applications. However, their integration had been slower than expected, partly due to the need for designing and developing new tools to take full advantage of these robots.

Despite the recent rise in popularity of fully-actuated robots, many control methods have already been introduced. However, these controllers require full pose 6-D setpoints, forcing the teams working on them to develop full-pose 6-D tools and methods around these controllers to use their robots in real applications. This re-development of the autonomy stack is inefficient, time-consuming, and requires many resources.

We propose a way of bridging the gap between the already available ecosystem for underactuated robots and the new fully-actuated vehicles. This approach can easily extend the existing underactuated flight controllers to support fully-actuated robots or enhance the existing fully-actuated controllers to work with the existing underactuated flight stacks. We introduce attitude strategies that work with the underactuated controllers, tools, planners, and remote control interfaces while taking advantage of the UAV's full actuation.

Moreover, thrust strategies are proposed to handle the limited lateral thrust properly, which is a problem with many fully-actuated UAV designs. The strategies are lightweight, simple, and allow rapid integration of the available tools with these new vehicles for the fast development of new real-world applications.

To show the flexibility of the proposed controller design, we further extend it for physical interaction tasks, making it a hybrid position and force controller. This extension allows precise control over the force and position during contact with the physical world. Such extension provides the ability to physically manipulate the environment during tasks such as inspection and maintenance.

Section 3.1 discusses the relevant work and provides an introduction to the rest of the chapter. Section 3.2 defines the problem and Section 3.3 introduces our control design solution in detail. Sections 3.4 and 3.5 explain the attitude and thrust strategies proposed with our controller design. Section 3.6 extends the control design to a complete force-position controller for controlling physical interactions. Finally, Section 3.7 illustrates the controller design's viability and performance with experiments on real and simulated robots for different free-flight and physical interaction tasks.

## 3.1 Introduction and Related Work

The past two decades have seen rapid growth in the number of multirotor applications. Traditional multirotors are designed to have co-planar rotors. Although this design choice is simple and maximizes energy efficiency, these UAVs suffer from underactuation (i.e., their rotational motion is unavoidably coupled with their translational motion).

The first attempts to address the underactuation issue of the multirotors date back to 2007 [154, 164]. The authors of these papers proposed a robot with eight rotors: four rotors were used for vertical thrust generation, similar to quadrotors; the other four rotors were used for lateral force generation, helping

the robot with lateral displacements.

The authors of [34] and [96] presented the first fully-actuated designs with the minimum number of rotors in 2011. They analyzed fully-actuated hexarotor design for fixed-pitch/variable-speed rotors and variable-pitch/fixed-speed rotors. Many other groups followed their lead by replicating the available designs for their research or slightly modifying the hexarotor designs for different optimal criteria [50, 78, 93, 98, 99, 112, 172].

These new designs allow fully independent control over both linear and angular motions [46, 151]. Many other configurations have been proposed to achieve the full actuation while optimizing or simplifying some design or controller aspects. Some examples of these configurations include modifying the quadrotors for full actuation [158], tetrahedron-shaped hexarotor [189], omni-directional octocopters [23, 24], semi-coaxial hexarotors [14], and hexarotors with additional servos [76, 77, 159]. Some other works have focused on studying the advantages and issues of the new fully-actuated designs for optimal structures [37, 113, 169, 178, 179, 180, 186], fault tolerance [50, 51, 52], wind tolerance [27], and robustness to aerodynamic effects [101].

The fully-actuated robots, in general, have less energy efficiency and generally have more complex hardware designs than underactuated UAVs. However, in many cases, the independent control over all six dimensions greatly simplifies the payload and the approach to a task, ultimately reducing the weight and final costs of the UAV's hardware, software, and development process. Moreover, it makes many new applications possible that were infeasible with underactuated designs. A more comprehensive literature review of fully-actuated multirotors can be found in [151] and [46].

The advantages of fully-actuated multirotors have already encouraged different teams to start developing commercial products. Figure 3.1 shows some of these designs for different applications. Voliro is an omni-directional multirotor developed by a spin-off from ETH Zurich for applications such as contact inspection and painting [192]; Skygauge is a fully-actuated multirotor specifically designed for non-destructive inspection that is still at the initial development

phase [174]; CyPhy is a failed Kickstarter project by an iRobot cofounder that was targeting the hobbyist community to provide a cheap gimbal-less fixed-tilt hexarotor for stable filming [35].

(a) (b) (c)

**Figure 3.1:** Examples of fully-actuated multirotors being commercialized. (a) Voliro drone for inspection, painting, and assembly. (b) Skygauge drone for inspection. (c) CyPhy drone for hobbyists.

Until recently, aerial robots were only used as contactless remote sensing devices, and the idea of the physical interaction of aerial robots with their environment has only been recently explored. The physical interaction applications are limitless, ranging from contact inspection, maintenance, and assembly to construction, sampling, assistance, delivery, and transportation. The automation of physical interactions not only has the potential to reduce the costs and the danger of many tasks, but it can also make some tasks previously deemed impossible possible, such as exploring disaster sites or operating inside nuclear plants. Hence, governments and large companies have defined large projects to find reliable solutions for these applications. Authors of [16] list some ongoing and finished projects defined and funded by the European Union.

The physical interaction of aerial robots with the environment can be divided into the following different types:

1. **Grasping and picking:** In this type, the robot's goal is to grasp or pick an external rigid or deformable object using the end-effector. The potential applications for the multirotors can include object retrieval, high-speed courier services, and intelligence gathering. The controller for this

type of contact generally aims to control the precise pose of the gripper. Depending on the application, it may also have to control the forces and moments exerted by the gripper.

2. **Momentary contact:** In this type, the robot's objective is to make brief contact with an external object, usually to hit it at the right time in the right direction. Applications of this contact type can include robots playing ball-based games, such as ping pong and volleyball. The controller generally aims to control the pose and velocity of the end-effector at a precise time.
3. **Static contact with rigid surfaces and objects:** This type of contact happens when the robot requires to exert force and torque on a rigid object. The object can be attached to the environment or separated, in which case it can be pushed with a certain amount of force. The most prominent application is non-destructive testing (NDT), which is used for the contact inspection of oil tanks, bridges, and other infrastructure for corrosion, cracks, and other problems. The controllers need to regulate the forces (and/or the moments) applied by the robot to the point of contact based on some contact model. However, precise force control may not be needed depending on the application, and the force can be controlled using passive or active compliance.
4. **Moving during contact with rigid surfaces and objects:** In this type, the robot needs to exert forces and torques to the surface while following a trajectory. Many real-world applications include this type of contact, including glass cleaning, wall painting, screw driving, and turning valves and handles. The controllers for these tasks need to control the pose and the force (and/or the moment) applied at the surface by the robot while following a pose and force/moment trajectory.
5. **Operating on deformable surfaces:** In this type, the surface may deform or change due to the robot's interaction. It requires integrating the prior knowledge about the surface properties into the motion planner and the controller and following the pose and force/moment trajectories to perform a task. Applications include manipulating rigid surfaces,

e.g., machining, drilling, hammering, and tasks involving non-rigid (i.e., deformable) objects such as cables, wires, textile, and plants.

Many industrial applications require the exertion of force to a surface or an object for repetitive tasks such as machining and assembly. Hence, force control for fixed-base industrial manipulator robots has been studied by researchers for the past several decades. With the progress of aerial robot technology, recently, there have been efforts to adapt the available force control methods or introduce new methods specific to aerial robots. However, fixed-base industrial robots have vital advantages over aerial robots, making it challenging to develop reliable force control methods for aerial robots:

- Fixed-base manipulator robots often operate in extremely controlled settings compared to the operating environment of aerial robots;
- The required applied forces for industrial robots are often small compared to their weight and power, while the same cannot be assumed for small aerial robots such as multirotors;
- The aerodynamic effects do not affect the industrial robots, while the effects such as the wall effect may become significant during the aerial robots' operation.

However, some of the challenges have been addressed before by communities working on other types of robots. The fundamental need in humanoid robots for controlling the interaction forces and moments has driven the research on controllers for physical interactions of manipulators with the environment for many years. Some recently proposed force control methods for multirotors have adapted the methods developed in the humanoid communities to introduce force and motion control working for fully-actuated multirotors (i.e., *aerial manipulation*).

The progress of aerial manipulation has dramatically accelerated in the past few years [22, 41, 115, 121, 157, 176]. However, the earliest attempts to apply force to the environment using multirotors go back to 2010. Authors of [2] added an extra actuator to a quadrotor to generate forces during physical

contact when the robot is hovering.

The work in [193] and [66] was one of the first studies of grasping using aerial platforms. The authors later modified their design to apply forces to the environment [64, 65], improved the flight controller [67, 68], and enhanced the design [71] for various applications, such as physical sampling [70] and inspection [69, 72].

Many different control schemes were initially proposed to optimize the aerial manipulation and physical interaction tasks. These schemes range from the simplest forms such as Proportional–Integral–Derivative (PID) controllers [91, 156], Linear–Quadratic Regulators (LQR) [201], or simply using the standard controllers [49] to more complex Interconnection and Damping Assignment-Passivity Based Control (IDA-PBC) [208, 209], Model Predictive Controller (MPC) [17, 107], dynamic decentralized controller [187], Null Space-based Behavioral (NSB) control [9, 10], Model Reference Adaptive Control (MRAC) [136], and geometric control methods [148, 213]. The recent trend is to adopt the force control methods of fixed-based manipulators, introducing the ideas of compliant arm force control [12, 26, 205], parallel force/motion control [149], and hybrid force-motion control [116] for aerial manipulators. Authors of [207], [185] and [16] provide comprehensive overviews of the control methods for physical interaction and manipulation.

Some aerodynamic challenges are raised by the fully-actuated designs and the specific requirements of physical interaction with the environment, such as wall and ceiling effects. The different aerodynamic effects on the aerial manipulation task and multirotor flight have been studied in [10] and [166].

Traditional underactuated multirotors require a gimbal or a multi-DoF arm to apply forces and moments at the desired direction to the contact point. This requirement increases the complexity and cost of the system, reduces the flight time and the maximum possible force and moment of the robot, and intensifies the uncertainty of the end-effector's estimated pose. Authors of [117] tried to mitigate the effects of the weight added by the manipulator's arms by exploiting the manipulator dynamics. A relatively new idea, which is

made possible with the development of fully-actuated robots, is to replace the manipulator arm with a rigidly-attached arm and control the end-effector's pose directly using the multirotor's pose. This paradigm, called *whole-body wrench generation*, eliminates the need for a complex arm controller, lowers the errors introduced by a gimbal or a multi-DoF arm, and reduces the payload of the system, increasing the maximum available forces and moments to apply at the point of contact. As far as we know, the first attempt to design a whole-body manipulator (rigidly-attached manipulator arm) to eliminate the need for the attached manipulation arm altogether was made in 2015 [132].

Many research groups have provided methods to control the end-effector's position and force on multirotors. The following works show the current state of the art in contact control using fully-actuated multirotor robots.

ETH Zürich researchers have proposed a method for Non-Destructive Testing (NDT) inspection. Their vehicle is a complex fully-actuated vehicle with 12 rotors able to take any orientation in 6-D, which is described in [18] and improved in [3] and [20]. The vehicle has a rigidly mounted manipulator arm, which was initially introduced in [19]. They use variable axis-selective impedance control, which integrates direct force control to control the desired interaction forces actively. They demonstrate force-controlled, peg in a hole, and push-and-slide interactions in different flight orientations. In their previous works, they have been able to demonstrate the use of their robots to control real-world settings such as inspection [143].

The researchers at the Laboratory for Analysis and Architecture of Systems (LAAS) have introduced methods for NDT inspection using a simpler fixed-tilt fully-actuated aerial manipulator they have introduced in [145]. They started with a complex manipulator arm in [188] and simplified the design to a rigidly attached arm in [161] and [127], introducing a whole-body force control strategy instead of controlling the arm. Authors of [161] extend the work from [160] using an outer admittance controller to control the desired admittance behavior while an inner loop controls the position. As a result, there is no direct control over the force and torque at the interaction point. In [127], direct force feedback

is used to control the energy, the motion, and the force of the end-effector using a Quadratic Programming optimization method that also respects the bounds and limits of the actuators. They demonstrate their method with force- and position-controlled push and push-and-slide interactions in the presence of external disturbances.

The researchers at the Seoul National University proposed a new omni-directional platform specifically for arbitrary wrench generation [140, 141]. Previously, their group has also worked on using the quadrotors for tool operations such as driving screws in simulation; however, they never completed their work with real experiments [97, 128, 129].

Some other groups have developed fully-actuated aerial manipulators for NDT contact inspection as well, including the researchers at the University of Twente [149, 150], and a team at the University of Seville [135, 190]. Besides, some companies have put effort towards commercializing the fully-actuated aerial manipulators for NDT inspection in recent years [174, 192].

The joint force and motion control methods can be divided into three broad categories [38]: compliant arm (indirect) force control (e.g., impedance control methods), parallel force/motion control (e.g., passivity-based methods), and hybrid force-motion control (e.g., task optimization-based methods) [137].

In the current work, we use the multirotor's body as the force generator, eliminating the need for having a manipulator arm with multiple degrees of freedom. A fixed-tilt multirotor (see Chapter 2) with a rigidly-attached arm is used for this purpose. We focus on the methods requiring the control of force and moments along with the pose of the end-effector rigidly attached to a fixed-tilt fully-actuated multirotor to perform non-destructive and environment-altering tasks such as cable manipulation at the top of a utility pole.

Despite all the progress with the fully-actuated UAVs, integrating these designs into real-life applications and physical interaction tasks has been much slower than the underactuated ones. The added design complexity, the lower energy efficiency, and the additional efforts required to develop new software tools prevent their widespread use in real projects. Even the success of the

commercial efforts (e.g., [35, 174, 192]) has been minimal so far.

Over the years, many methods and tools for underactuated multirotors have been developed and are widely available to plan missions and trajectories for different purposes and to control the motion of the robots based on the planned trajectories. The emergence of new architectures has resulted in the introduction of new control methods for the fully-actuated vehicles in different applications, ranging from exact feedback linearization [112, 145] to nonlinear model-predictive control [17]. However, most controllers developed for fully-actuated vehicles require full-pose trajectories and cannot interact with the tools developed for underactuated UAVs. An even more significant challenge is the need for new Remote Control (RC) interfaces that can control the robot in all the six degrees of freedom, making it difficult for the pilots to learn to fly or transition to flying these new robots.

Many research groups utilize tools such as Simulink to design and generate the code for their new fully-actuated controllers. While these tools can also generate the code required for a working flight control system (such as the state estimation), the generated code is too basic for any tasks performed beyond heavily-controlled lab environments. Moreover, the code cannot be directly integrated with the existing autopilots that already have a more powerful flight stack, slowing down the integration of the new robots into real applications even further.

We have realized that most of the applications do not require a completely free change in orientation and the vehicle's attitude generally needs to follow a set of patterns. We can utilize this observation to define a set of *fully-actuated operation modes* for the attitude (we refer to them as *attitude strategies*). The strategies can be switched during the flight to address the different needs of applications. Identifying such a set of strategies allows using the readily-available flight controllers, software tools, and RC interfaces developed for the underactuated UAVs.

On the other hand, many fully-actuated UAVs, including the most common architectures, such as fixed-pitch hexarotor designs, can generate only limited

lateral forces compared to the forces perpendicular to their bodies. Franchi et al. [47] have proposed to call these vehicles *LBF* (vehicles with laterally bounded force), which we will also use in this document. The operation of the LBF robots requires particular attention to handling their lateral thrust limits. We propose a set of methods (*thrust strategies*) that can handle the lateral thrust limits with minimal changes to the available flight controllers, allowing the use of either 6-D (full-pose) or traditional 4-D planners and motion controllers.

This chapter presents a new way of developing fully-actuated controllers by extending the existing underactuated flight controllers while keeping the entire flight stack intact without any modifications and allowing to take advantage of the full actuation in the UAV applications. Other fully-actuated controller methods can also adopt the approach to provide easier integration into existing flight stacks. It allows the controller to accept the traditional waypoints and simplifies the development of real-world tasks by easy switching between different strategies. Our proposed modifications to the controller have minimal overhead and can be directly integrated into any autopilot, extending only the Position Controller module.

Finally, we illustrate how the proposed controller design can be extended for tasks requiring precise physical interaction with the environment. The controller is extended into a hybrid force and position controller, allowing the control of forces and motions during the contact.

We show our experiments with the new controller design in Gazebo PX4 SITL and our MATLAB simulator as well as on the real robot to illustrate the proposed strategies and methods in free flight and during physical interaction.

## 3.2 Problem Definition

A general flight controller system for multirotors consists of a module for controlling the position (and/or position derivatives), a module for controlling the attitude (and/or angular rates), and a module for control allocation (mixing).

Figure 3.2 illustrates the architecture for a typical flight controller for underactuated UAVs. The inputs are normally the desired yaw, the desired position, and/or their derivatives. These inputs are internally utilized to generate the full desired attitude (we call it *attitude setpoint*) for its Attitude Controller module and the desired linear acceleration or forces (we call it *thrust setpoint*) for its Control Allocation module.

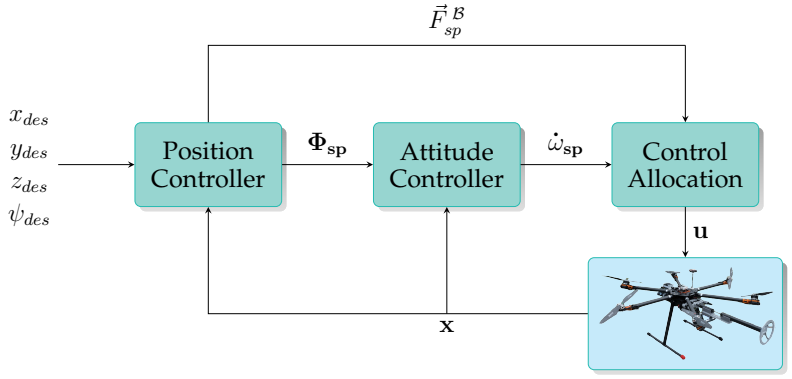



**Figure 3.2:** High-level architecture of a typical flight controller. Some architectures consist of more modules, which can be collected together to have the same general architecture with similar inputs and outputs.

The attitude and thrust setpoints generation usually happens within some parent module (e.g., Position Controller). However, without the loss of generality, we assume that they are separate submodules with their inputs and outputs. Figure 3.3 illustrates the internal structure of the Position Controller module of the PX4 flight controller with the thrust and attitude setpoint generation functions depicted as separate submodules.

Some flight controller designs may have slightly different input/output combinations. For example, the forces can be replaced by linear accelerations, which have a linear relationship with forces. Such choices do not change the concepts discussed in this section. Moreover, in some controller designs, the force outputs may be expressed inertial instead of body-fixed frames if the Control Allocation module uses the force in the inertial frame (e.g., see [145]). In such cases, the Thrust Setpoint Generation function may only apply the

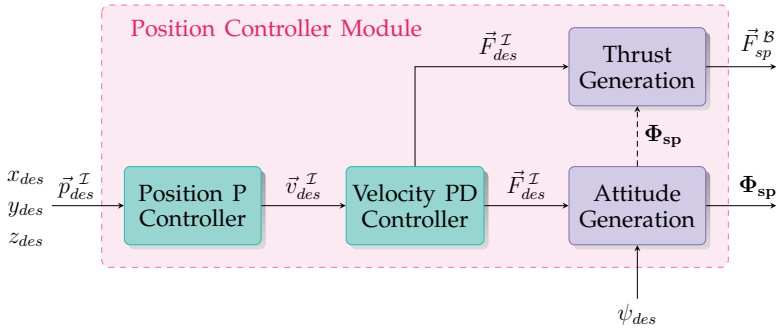



**Figure 3.3:** The internal structure of the Position Controller module for the PX4 flight controller as an example of a typical structure utilized in underactuated multirotors. The multirotor's state $\mathbf{x}$ is used as an input to all the submodules in the figure but is omitted for simplicity.

thrust limits to the input or not do anything. In this section, we add the Thrust Generation module even when it does nothing in the original controller design to allow the extension of its functions in Section 3.5.

## 3.3 Our Controller Design

Sections 2.1 and 3.1 reviewed the prior work on fully-actuated controllers and described the multirotor kinematics and dynamics model for the fixed-pitch multirotor designs along with a control allocation method based on nonlinear dynamic inversion. Tools such as Simulink and MATLAB allow the design and synthesis of entirely new autopilot systems for the fully-actuated UAVs based on the provided formulations. The resulting autopilots can run directly on a computer connected to the UAV or even compiled for the standard autopilot hardware such as Pixhawk. These tools add the necessary code for functions such as perception, state estimation, and hardware interface, allowing rapid development and testing of new control ideas in controlled lab environments. However, the added code is not as comprehensive as the standard autopilot systems and cannot directly be devised on the UAVs in the real world.

On the other hand, extending the existing autopilots to support a new type of vehicle may seem more challenging and time-consuming. However,

the functionalities of the established autopilots are more comprehensive and extensively tested. Therefore, extending the available autopilots to support the new fully-actuated multirotors accelerates the UAVs' integration with real-world applications.

We propose that extending the existing controllers is possible by the following minimal set of changes:

- Modifying or replacing the Control Allocation module to support the new architecture and prioritize angular acceleration over the linear acceleration.
- Extending the Attitude Setpoint Generation function into an Attitude Setpoint Generator module which allows utilizing the full actuation based on chosen strategies (see Section 3.4).
- Extending the Thrust Setpoint Generation function into a Thrust Setpoint Generator module, which manipulates the thrust setpoint to respect the fully-actuated vehicles' thrust limits.

Figure 3.4 illustrates the design of the controller we implemented for fully-actuated multirotors based on the existing PX4 autopilot architecture [114]. The controller only modifies the Control Allocation module but devises all other parts of the existing controller on PX4 autopilot for faster implementation, better stability, and integration with other flight controller modules (e.g., perception and state estimation). It also separates the Thrust and Attitude Setpoint Generator modules from the Position Controller and extends them for our purposes.

In addition to the modification of the Control Allocation module to support the new architecture, the overall controller structure depends on the controller's input:

1. For inputs with only position and yaw (and/or their derivatives), additional modules for thrust and attitude setpoint generation are used that depend on a desired inertial thrust input from the position controller (Figure 3.4(a)). The Attitude Setpoint Generator module produces the full desired attitude for the Attitude Controller to track, making the Attitude

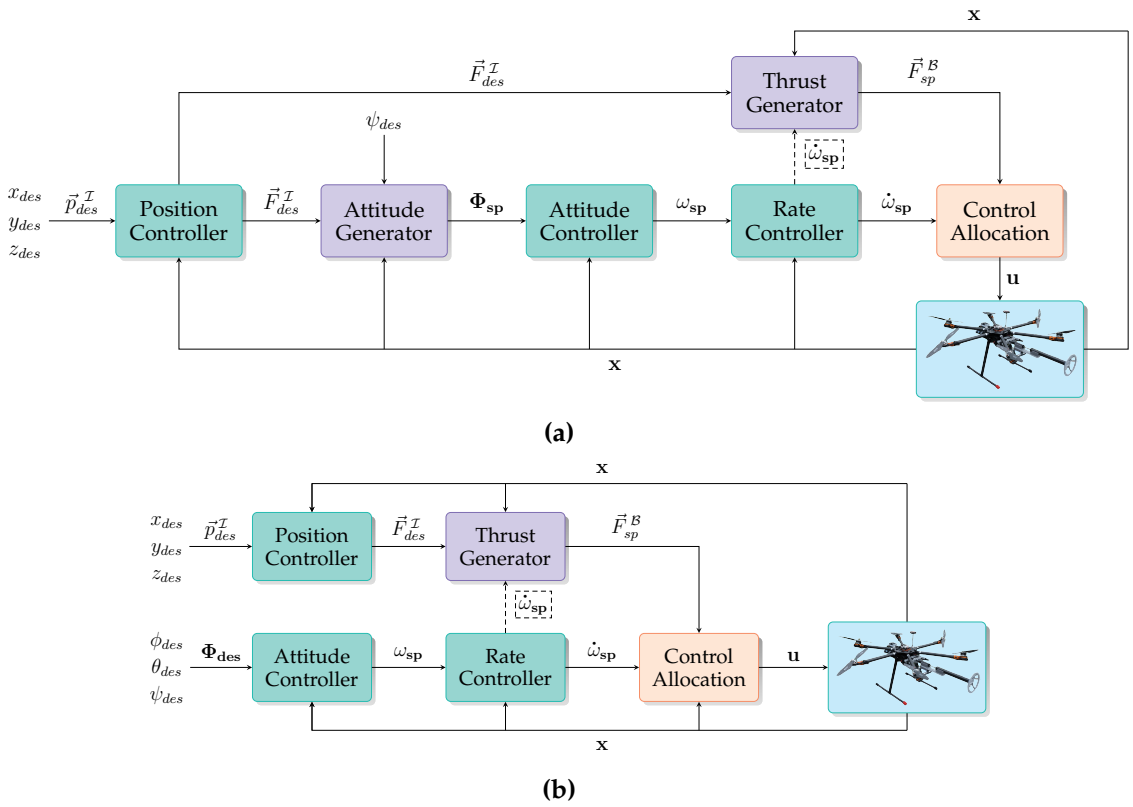



**Figure 3.4:** Our flight controller architecture for fully-actuated multirotors based on the PX4 flight controller design. (a) The controller's input is only the desired position and the desired yaw (and/or their derivatives), requiring an Attitude Generation module to calculate the pitch and roll. The input to the Attitude Controller depends on the Position Controller and the Attitude Setpoint Generator modules. (b) The controller input is the desired pose (position and orientation). The input attitude already includes the full desired attitude and does not require any manipulation, which eliminates the need for the Attitude Setpoint Generator module and makes the Attitude Controller fully independent from the Position Controller.

Controller an inner loop for the Position Controller. Depending on the application and the priorities defined by the user, a specific attitude generation strategy can be devised. Section 3.4 discusses the attitude and thrust setpoint generation methods we have developed and implemented for the current work.

2. For full-pose inputs with both the desired position and complete desired orientation (and/or their derivatives), the Attitude Setpoint Generator module of Figure 3.4(a) loses its functionality. Simplifying the structure of

Figure 3.4(a) shows that now the Position and Attitude Controller modules can work independently to generate the body thrusts and moments required for the UAV to track the desired input (Figure 3.4(b)).

After integrating new fully-actuated UAVs into an existing autopilot (by modifying the Control Allocation module to support the new architecture), we learned that a problem might arise when generating both the thrust setpoint and the angular acceleration setpoint (coming from the Attitude Controller in Figure 3.2) is not feasible. In this situation, some motor commands calculated by the Control Allocation module will be in the saturation range, which may result in instability for the whole robot if not appropriately handled.

There are many methods available to handle motor saturation. The default behavior for underactuated multirotors is usually either to bound the motor signal commands or devise a strategy that ensures the $\hat{Z}_{\mathcal{B}}$-thrust (normal thrust) is prioritized. However, with the fully-actuated UAVs, the strategy needs to be changed to prioritize the moments around the $\hat{X}_{\mathcal{B}}$ and $\hat{Y}_{\mathcal{B}}$ axes. This change is crucial in keeping the UAV's stability when large commands are given with fixed attitude strategies. Many such strategies have been introduced for underactuated UAVs and can be easily modified for the fully-actuated vehicles (e.g., see [25, 45, 175]). In our implementation, we modified the Airmode functionality of the PX4 firmware and prioritized the moments around the $\hat{X}_{\mathcal{B}}$ and $\hat{Y}_{\mathcal{B}}$ axes over the thrusts and the moment around the $\hat{Z}_{\mathcal{B}}$ axis.

## 3.4 Attitude Strategies for Fully-Actuated UAVs

In traditional coplanar multirotors, the robot can only generate thrust normal to its rotors plane, requiring it to completely tilt towards the total desired thrust direction to align the generated thrust with the desired thrust. However, fully-actuated vehicles are capable of independently controlling their translation and orientation.

When a full-pose 6-D input is passed to the controller, no additional

processing is required on the desired orientation. However, generally, with underactuated controllers, only the yaw is given to the controller, and the other two degrees of orientation are derived from the desired thrusts. We call this calculated desired orientation as *attitude setpoint* which is then sent to the Attitude Controller module.

The autonomous controller developed in Section 3.3 has an Attitude Setpoint Generation module for when the given attitude input only specifies the desired yaw (see Figure 3.4(a)). This module accepts three inputs: the desired thrust force in the inertial frame ($\vec{F}^{\mathcal{I}}_{des}$) coming from the Position Controller module, the desired yaw ($\psi_{des}$) coming from the motion controller (which may be following a trajectory or converting the user's RC commands to motion commands), and the UAV's current state $\mathbf{x}$. The module's only output is the complete attitude setpoint ($\mathbf{\Phi_{sp}}$), which serves as an input for the Attitude Controller module.

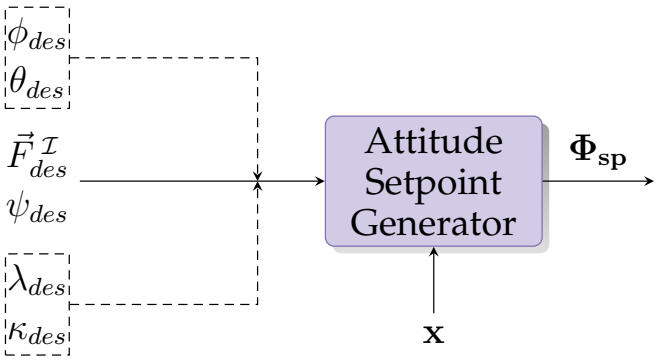



**Figure 3.5:** An illustration of the Attitude Setpoint Generation module with its inputs and outputs. The optional inputs are enclosed in dashed boxes.

Some flight controller designs may include a slightly different set of inputs. For example, the input force vector may be replaced by linear accelerations, a linear conversion that does not change the concepts discussed here.

For the set of strategies proposed in this section, there are additional strategy-specific inputs (enclosed in dashed boxes in Figure 3.5), which are not present in the underactuated controllers. These inputs will be explained in their relevant strategies later in this section.

Finally, in this problem, it is assumed that the given desired thrusts have already compensated for the gravity force to achieve the desired acceleration, i.e., the thrust equal to the vehicle's weight is added to the upward $z$ component of the desired thrust.

We have implemented several strategies for fully-actuated multirotors:

1. **Zero-tilt attitude strategy:** This strategy keeps the robot's tilt at zero at all times, allowing it to stay completely horizontal during the flight.
2. **Full-tilt attitude strategy:** This is the traditional attitude generation method for multirotors, where the output body-fixed $\hat{Z}$ axis is always in the opposite direction of the desired input thrust.
3. **Minimum-tilt attitude strategy:** This strategy minimizes the robot's tilt but does not guarantee to keep it at zero, slightly tilting it towards the desired thrust when the desired thrust is significant, allowing the robot to achieve larger accelerations than what is possible in the Zero-tilt strategy.
4. **Fixed-tilt attitude strategy:** This strategy keeps the robot's tilt at the desired angle and towards the desired direction, independent of the given desired thrust, which can be helpful in some situations, such as flight during a strong wind.
5. **Fixed-attitude strategy:** This strategy keeps the robot's roll and pitch as desired, independent of the given desired thrust, which can be useful for some situations, such as flight during physical contact with a surface in the wind or at the desired contact angle.

The attitude is usually represented as the set of Euler angles (i.e., roll, pitch, and yaw), rotation matrices, or different types of quaternions, each having its pros and cons. This section uses rotation matrices to represent attitudes and shows the direct ways of calculating the Euler angles when such shortcuts exist.

For each strategy, it is explained how to derive the attitude setpoint in the rotation matrix form. The rotation matrix is basically the composition of the unit vectors $\hat{\vec{i}}_{\mathcal{S}}$, $\hat{\vec{j}}_{\mathcal{S}}$ and $\hat{\vec{k}}_{\mathcal{S}}$, in the directions of $\hat{X}_{\mathcal{S}}$, $\hat{Y}_{\mathcal{S}}$ and $\hat{Z}_{\mathcal{S}}$ axes of the

setpoint frame $\mathcal{F}^{\mathcal{S}}$, respectively:

$$\mathbf{R}_{\mathcal{IS}} = \begin{bmatrix} \hat{\mathbf{i}}_{\mathcal{S}} & \hat{\mathbf{j}}_{\mathcal{S}} & \hat{\mathbf{k}}_{\mathcal{S}} \end{bmatrix} \tag{3.1}$$

This representation can be converted to any other representation used by the Attitude Controller module. For example, the conversion to the Euler angles can be done using Equation 3.2:

$$\mathbf{\Phi_{sp}} = \begin{bmatrix} \arctan\left(\dfrac{\hat{\mathbf{j}}_3}{\hat{\mathbf{k}}_3}\right) \\ -\arcsin\left(\hat{\mathbf{i}}_3\right) \\ \psi_{des} \end{bmatrix} \tag{3.2}$$

The rest of this section describes the proposed attitude strategies and the applications where each can be useful.

### 3.4.1 Zero-Tilt Attitude Strategy

Keeping the multirotor's attitude at zero tilt (i.e., keeping it horizontally level) during the flight can be beneficial for many situations, such as making precise contact with the vertical surface or capturing a video using an onboard camera without the need for a gimbal.

Calculating the output attitude setpoint in this strategy is straightforward. The output roll and pitch are always zero; therefore, the direction of the $\hat{Z}_{\mathcal{S}}$ axis for the attitude setpoint would be the same as $\hat{Z}_{\mathcal{I}}$ (i.e., $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$), independent of the desired inputs. Additionally, based on our definition of the body-fixed frame, the direction of the $\hat{X}_{\mathcal{S}}$ axis is the direction of the input desired yaw $\psi_{des}$. The $\hat{Y}_{\mathcal{S}}$ axis is perpendicular to it, and both lie on the horizontal plane in this strategy. Figure 3.6 shows a model used for the zero-tilt attitude generation

strategy.

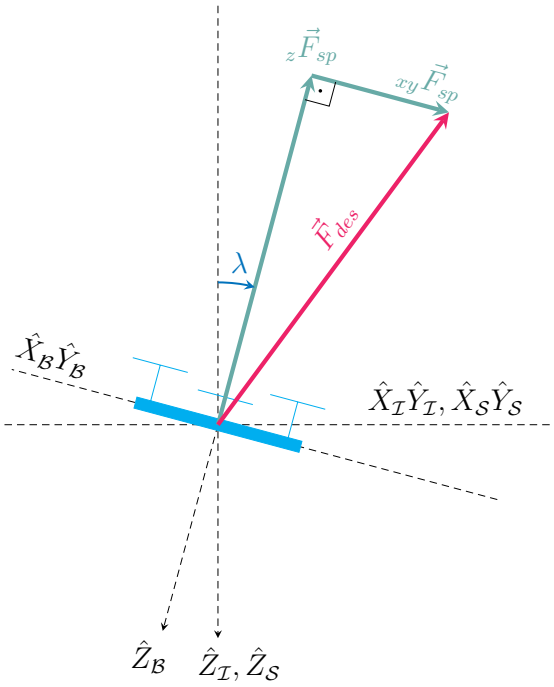



**Figure 3.6:** The model used for zero-tilt attitude calculation for fully-actuated multirotors. The resulting attitude setpoint always has its $\hat{Z}_\mathcal{S}$ axis pointing in the direction of the gravity ($\hat{Z}_\mathcal{I}$).

The rotation matrix for this attitude setpoint can be constructed from the unit vectors in the direction of the setpoint axes as:

$$\mathbf{R}_{\mathcal{IS}} = \begin{bmatrix} \cos\psi_{des} & -\sin\psi_{des} & 0 \\ \sin\psi_{des} & \cos\psi_{des} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

While it is possible to use Equation 3.2 to calculate the Euler angles, in this specific strategy, the Euler angles are directly defined as:

$$\mathbf{\Phi_{sp}} = \begin{bmatrix} 0 \\ 0 \\ \psi_{des} \end{bmatrix} \tag{3.4}$$

### 3.4.2 Full-Tilt Attitude Strategy

With coplanar underactuated multirotor designs, the only way for the robot to achieve the input desired thrust is to tilt so that the direction of the desired thrust is normal to the plane of the rotors. The strategy does not take advantage of the full actuation in the fully-actuated robots, but it is the most helpful strategy to oppose the external forces and disturbances. Additionally, it is readily available in the popular autopilots and can be used for fully-actuated robots right out of the box.

In most common multirotor architectures, this strategy requires the minimum energy and is the best choice when controlling the robot's attitude is not essential for the task. Section 5.3 describes how this method can further be used to estimate the optimal tilt when an external force is applied to the robot (e.g., in windy conditions).

Figure 3.7 shows a model to demonstrate the full-tilt attitude strategy. In this strategy, the desired input thrust $\vec{F}^{\mathcal{I}}_{des}$ should be normal to the robot's output $\hat{X}_{\mathcal{S}}\hat{Y}_{\mathcal{S}}$ plane and is in the opposite direction of the $\hat{Z}_{\mathcal{S}}$ axis. Therefore, the unit vector in $\hat{Z}_{\mathcal{S}}$ direction can be computed as:

$$\hat{\vec{k}}_{\mathcal{S}} = -\frac{\vec{F}^{\mathcal{I}}_{des}}{\left\|\vec{F}^{\mathcal{I}}_{des}\right\|} \tag{3.5}$$

If a unit vector in the desired yaw direction on the $\hat{X}_{\mathcal{I}}\hat{Y}_{\mathcal{I}}$ plane is rotated for $+90^{\circ}$ around the $\hat{Z}_{\mathcal{I}}$ axis, it will be on the plane made from axes $\hat{Y}_{\mathcal{S}}$ and $\hat{Z}_{\mathcal{S}}$. Using this observation, the unit vector in the $\hat{X}_{\mathcal{S}}$ direction can be calculated as:

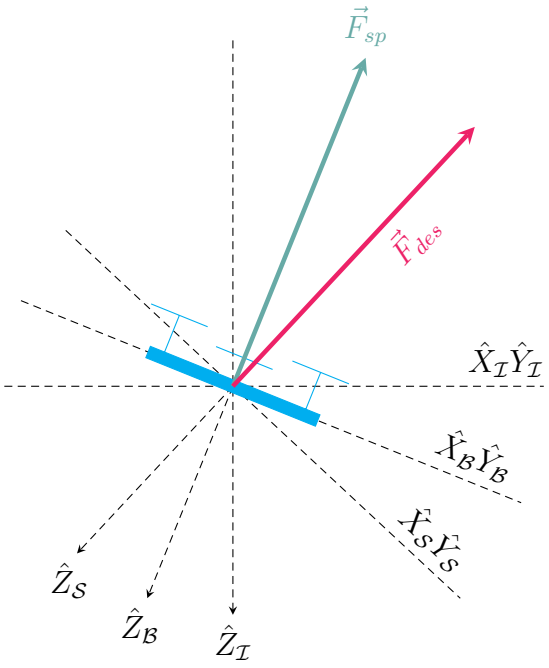



**Figure 3.7:** (a) The model used for full-tilt attitude calculation for both fully-actuated and underactuated multirotors. The thrust setpoint is aligned with the current body-fixed $ZB$ axis, while the attitude setpoint is generated based on the desired thrust direction.

$$\hat{\vec{i}}_{\mathcal{S}} = \frac{\begin{bmatrix} -\sin\psi_{des} \\ \cos\psi_{des} \\ 0 \end{bmatrix} \times \hat{\vec{k}}_{\mathcal{S}}}{\left\| \begin{bmatrix} -\sin\psi_{des} \\ \cos\psi_{des} \\ 0 \end{bmatrix} \times \hat{\vec{k}}_{\mathcal{S}} \right\|} \tag{3.6}$$

From the $\hat{X}_{\mathcal{S}}$ and $\hat{Z}_{\mathcal{S}}$ axes calculated in Equations 3.5 and 3.6, the unit vector in the direction of $\hat{Y}_{sp}$ axis can be computed as:

$$\hat{\vec{j}}_{\mathcal{S}} = \hat{\vec{k}}_{\mathcal{S}} \times \hat{\vec{i}}_{\mathcal{S}} \tag{3.7}$$

The rotation matrix and the Euler angles for the attitude setpoint can be computed from Equations 3.5, 3.6 and 3.7 using Equations 3.1 and 3.2.

### 3.4.3 Minimum-Tilt Attitude Strategy

The zero-tilt strategy described in Section 3.4.1 requires a lateral thrust input that is less than its maximum lateral thrust limit. Section 3.5 describes how to bound the lateral input thrust when it is more than the maximum available lateral thrust $F_{lmax}$. In some applications, keeping the tilt magnitude as close to zero is desirable, but achieving the input accelerations or rejecting high external forces is more important than maintaining the fixed attitude. An example scenario is filming a highly dynamic target or in high-gust winds. In such cases, a better strategy than full-tilt (Section 3.4.2) can be devised that takes advantage of the full actuation to minimize the tilt of the robot by using up the lateral thrust. The strategy is first described in [112].

In this strategy, the vehicle keeps its attitude at zero tilt until a larger than maximum lateral thrust is needed, then the vehicle minimally tilts, keeping its lateral thrust at maximum to reduce the tilt as much as possible.

Figure 3.8 helps demonstrating how the minimum-tilt attitude is calculated.

Given a desired input thrust vector $\vec{F}_{des}^{\mathcal{I}}$, if the desired thrust on the lateral plane $\|_{\hat{X}_{\mathcal{I}}\hat{Y}_{\mathcal{I}}}\vec{F}_{des}\|$ is less than the maximum possible lateral thrust $F_{lmax}$, then the zero-tilt attitude strategy is used to calculate the attitude setpoint (see Section 3.4.1). However, when the desired thrust on the horizontal plane is larger than the available lateral thrust, all the possible thrust on the lateral plane is used first, then the remaining required thrust determines the required tilt $\lambda_{sp}$, roll $\phi_{sp}$ and pitch $\theta_{sp}$. To calculate the tilt $\lambda_{sp}$, we have:

$$\lambda_{sp} = \underbrace{\arcsin\left(\frac{\left\|\hat{X}_{\mathcal{I}}\hat{Y}_{\mathcal{I}}\vec{F}_{des}\right\|}{\left\|\vec{F}_{des}\right\|}\right)}_{\chi} - \underbrace{\arcsin\left(\frac{F_{lmax}}{\left\|\vec{F}_{des}\right\|}\right)}_{\mu} \tag{3.8}$$

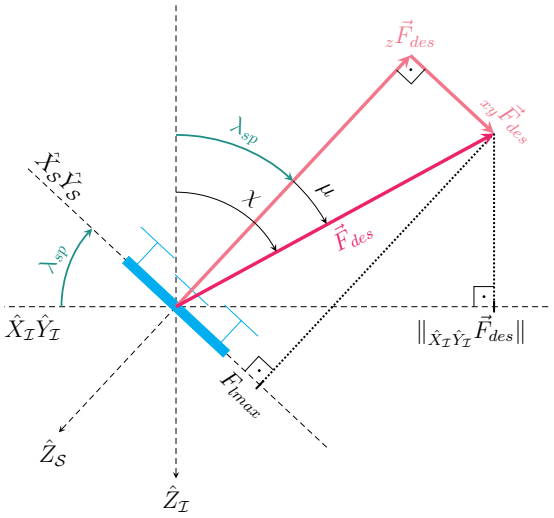



**Figure 3.8:** An illustration of the model for calculation of the minimum-tilt attitude. At the minimum tilt, the projection of the input desired thrust on robot's body-fixed horizontal plane is at the maximum available lateral thrust. The illustrated vectors $_z\vec{F}_{des}$ and $_{xy}\vec{F}_{des}$ are respectively the normal and lateral elements of $\vec{F}_{des}$ in the attitude setpoint frame.

where $\chi$ and $\mu$ angles are as illustrated in Figure 3.8, and $_{\hat{X}_I\hat{Y}_I}\vec{F}_{des}$ is the projection of the desired thrust on the horizontal plane.

The axis of rotation $\vec{r}$ for the tilt $\lambda_{sp}$ is perpendicular to the plane consisting of the desired thrust $\vec{F}_{des}$ and the inertial $\hat{Z}_I$ axis. Hence, it can be calculated as:

$$\vec{r} = \frac{\vec{F}_{des} \times \hat{\vec{k}}_I}{\left\| \vec{F}_{des} \times \hat{\vec{k}}_I \right\|} \tag{3.9}$$

The $\hat{Z}_S$ axis direction can be computed by rotating the $\hat{\vec{k}}_I$ unit vector around $\vec{r}$ using the Rodrigues' rotation formula:

$$\hat{\vec{k}}_S = (1 - \cos\lambda_{sp})\ (\vec{r} \cdot \hat{\vec{k}}_I)\ \vec{r} + \hat{\vec{k}}_I\ \cos\lambda_{sp} + (\vec{r} \times \hat{\vec{k}}_I)\ \sin\lambda_{sp} \tag{3.10}$$

The $\hat{X}_S$ and $\hat{Y}_S$ axes can be calculated similar to the full-tilt strategy (Section 3.4.2) from Equations 3.6 and 3.7. Similarly, the rotation matrix and

the Euler angles for the attitude setpoint can be computed using Equations 3.1 and 3.2 from these axes.

### 3.4.4 Fixed-Tilt Attitude Strategy

Some applications require keeping a specific tilt angle for the multirotor. An example scenario is flying in the constant wind where keeping a fixed tilt against the wind is desirable to increase the remaining thrust after opposing the wind, independent of the yaw and the movement direction.

Figure 3.9 shows a model demonstrating the axes and angles used in the calculation of the fixed-tilt attitude strategy.

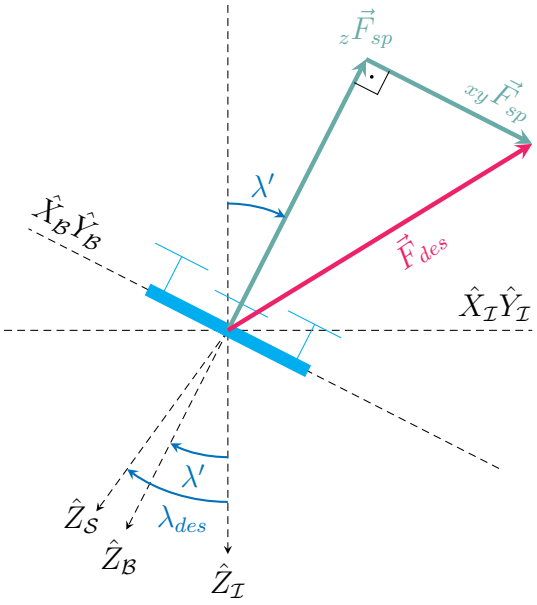



**Figure 3.9:** An illustration of the model used for the fixed-tilt attitude strategy.

In addition to the desired thrust and yaw inputs, let us assume two new inputs $\lambda_{des}$ and $\kappa_{des}$ to the system, representing the angle of the desired tilt and the direction of the tilt, respectively. These two inputs are shown in Figure 3.5 in a dashed box and are only used for this attitude strategy. We assume that

the desired direction $\kappa_{des}$ is given in the inertial frame (i.e., with respect to the north direction).

The axis of rotation to tilt the robot is perpendicular to the inertial $\hat{Z}_{\mathcal{I}}$ axis and the projection of the vector pointing in the direction of tilt on the $\hat{X}_{\mathcal{I}}\hat{Y}_{\mathcal{I}}$ plane. Hence, the axis of rotation $\vec{r}$ can be computed as:

$$\vec{r} = \begin{bmatrix} \cos\kappa_{des} \\ \sin\kappa_{des} \\ 0 \end{bmatrix} \times \hat{\vec{k}}_{\mathcal{I}} \tag{3.11}$$

Having the rotation axis and considering that $\lambda_{sp} = \lambda_{des}$, the $\hat{Z}_{\mathcal{S}}$ axis direction can be calculated using the Rodrigues's rotation formula of Equation 3.10, and the $\hat{X}_{\mathcal{S}}$ and $\hat{Y}_{\mathcal{S}}$ axes can be calculated from Equations 3.6 and 3.7. Finally, the rotation matrix and the Euler angles for the attitude setpoint can be computed using Equations 3.1 and 3.2 from these axes.

### 3.4.5 Fixed-Attitude Strategy

Suppose the controller's input trajectory from the onboard computer includes full-pose information (both full 3-D position and 3-D attitude). In that case, the attitude generator can be bypassed altogether (see the controller architecture in Figure 3.4(b)). However, in practice, some applications require achieving specific attitude angles for the multirotor without the input trajectory explicitly including the roll-pitch angles. An example scenario is during the robot's contact with the wall when a constant orientation can help control the end-effector's pose and wrench. To achieve this goal, the Attitude Generator Module (Figure 3.5) can devise two new inputs for the desired roll $\phi_{des}$ and the desired pitch $\theta_{des}$.

In this strategy, the rotation matrix can be directly calculated from the given Euler angles using the "3-2-1"-rotation sequence:

$$\mathbf{R}_{\mathcal{IS}} = \begin{bmatrix} \mathrm{c}\,\theta\,\mathrm{c}\,\psi & \mathrm{s}\,\phi\,\mathrm{s}\,\theta\,\mathrm{c}\,\psi - \mathrm{c}\,\phi\,\mathrm{s}\,\psi & \mathrm{c}\,\phi\,\mathrm{s}\,\theta\,\mathrm{c}\,\psi + \mathrm{s}\,\phi\,\mathrm{s}\,\psi \\ \mathrm{c}\,\theta\,\mathrm{s}\,\psi & \mathrm{s}\,\phi\,\mathrm{s}\,\theta\,\mathrm{s}\,\psi + \mathrm{c}\,\phi\,\mathrm{c}\,\psi & \mathrm{c}\,\phi\,\mathrm{s}\,\theta\,\mathrm{s}\,\psi - \mathrm{s}\,\phi\,\mathrm{c}\,\psi \\ -\mathrm{s}\,\theta & \mathrm{s}\,\phi\,\mathrm{c}\,\theta & \mathrm{c}\,\phi\,\mathrm{c}\,\theta \end{bmatrix} \tag{3.12}$$

where $c$ and $s$ are shorthand for $\cos$ and $\sin$ functions, respectively, and $\phi$, $\theta$, $\psi$ are used for $\phi_{des}$, $\theta_{des}$ and $\psi_{des}$.

## 3.5 Thrust Strategies for Fully-Actuated UAVs

The controller architecture for fully-actuated multirotors, which is described in Section 3.3, has a Thrust Setpoint Generator module that takes the desired thrust calculated by the Position Controller and prepares it for the Control Allocation module (see Figure 3.4). Figure 3.10 shows this module separately with all of its inputs and outputs.

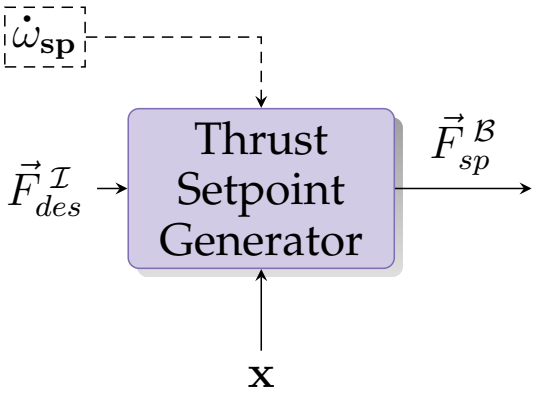



**Figure 3.10:** An illustration of the Thrust Setpoint Generator module with its inputs and outputs. The optional input is enclosed in a dashed box.

The thrust setpoint of the Attitude Setpoint Generator module (Figure 3.5) is expressed in body-fixed frame $\mathcal{F}^{\mathcal{B}}$ while here, the input desired thrust is in the inertial frame $\mathcal{F}^{\mathcal{I}}$. Assuming unlimited available thrust, the input thrust can be

simply rotated from the inertial frame to the current body-fixed frame (i.e., $\vec{F}_{des}^{\mathcal{I}}$ can be projected on the current body-fixed axes) to compute the thrust setpoint:

$$\vec{F}_{sp}^{\mathcal{B}} = \mathbf{R}_{\mathcal{BI}} \cdot \vec{F}_{des}^{\mathcal{I}} = \begin{bmatrix} \vec{F}_{des}^{\mathcal{I}} \cdot \hat{X}_{\mathcal{B}} \\ \vec{F}_{des}^{\mathcal{I}} \cdot \hat{Y}_{\mathcal{B}} \\ \vec{F}_{des}^{\mathcal{I}} \cdot \hat{Z}_{\mathcal{B}} \end{bmatrix} \tag{3.13}$$

Note that in some controller designs, the output thrust setpoint may be described in the inertial frame if the Control Allocation module requires the force to be in the inertial frame (e.g., see [145]). For such architectures, the thrust setpoint calculated using this section's methods can be simply rotated back from the body-fixed frame to the inertial frame.

In practice, the available thrust is not unlimited, and fully-actuated vehicles have thrust limits that should be considered. Particularly, fully-actuated LBF multirotors (see the definition in Section 3.1) have limited lateral thrust compared to the normal thrust (i.e., in $\hat{Z}_{\mathcal{B}}$ direction) due to their structure, and if the elements of the input desired thrust $\vec{F}_{des}$ on the body $\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$ plane (i.e., $\|_{\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}}\vec{F}_{des}\|$) are larger than the maximum possible lateral thrust ($F_{lmax}$), some motors will saturate, and the whole system may lose its stability.

One solution proposed so far in [47] can be devised when a full-pose trajectory is available, and it sacrifices the orientation over the position to track the given trajectory. The method requires a full-pose planner and a particular controller architecture and cannot be easily integrated with the available underactuated tools and controllers. The methods proposed in this section are fundamentally different. They can work with all common controller architectures by refining the thrust setpoint to respect the thrust limits and minimize the stability issues.

Before continuing to describe our methods, we need to define the terminology used in this section:

*Lateral thrust* ($\vec{F}_{lat}$): The component of the thrust on the body-fixed $\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$

plane. It is the vector constructed from the thrust's $x$ and $y$ components in the $\mathcal{F}^{\mathcal{B}}$ frame.

*Normal thrust* ($\vec{F}_{nor}$): The component of the thrust on the body-fixed $\hat{Z}_{\mathcal{B}}$ axis.

*Horizontal thrust* ($\vec{F}_{hor}$): The component of the thrust on the inertial $\hat{X}_{\mathcal{I}}\hat{Y}_{\mathcal{I}}$ plane. It is the vector constructed from the thrust's $x$ and $y$ components in the $\mathcal{F}^{\mathcal{I}}$ frame.

*Vertical thrust* ($\vec{F}_{ver}$): The component of the thrust on the inertial $\hat{Z}_{\mathcal{I}}$ axis.

*Hover thrust* ($F_{hov}$): The vertical thrust required to keep the UAV hovering when no wind is acting on it. In other words, the hover thrust of the UAV is the total weight of the UAV and all of its attached components. The hover thrust can be defined differently as the total thrust (or the percentage of the maximum possible thrust) generated by the rotors to keep the UAV hovering; however, in this section, we refer to hover thrust as the required vertical thrust.

*Maximum lateral thrust* ($F_{lmax}$): The maximum achievable force on the lateral plane ($\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$) of the UAV in the direction of the desired force at any specific state. Chapter 4 introduces methods to estimate the available thrusts, which can provide an accurate estimation of the lateral thrust limits at each time. For simplicity, here we assume that $F_{lmax}$ is a constant value independent of both the system state and the desired thrust and is uniform in all directions on the $\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$ plane.

Given the input desired thrust $\vec{F}^{\mathcal{I}}_{des}$ and its rotation, two cases can happen with the thrust setpoint $\vec{F}^{\mathcal{B}}_{sp}$ from Equation 3.13:

*Case 1.* $\|\vec{F}_{lat}\| \leq F_{lmax}$: In this case, generating the desired input thrust is feasible and the result of Equation 3.13 can be directly used as the output thrust setpoint.

*Case 2.* $\|\vec{F}_{lat}\| > F_{lmax}$: In this case, the robot will not be able to achieve the desired thrust. This section provides solutions to address this case based on the application requirements.

We describe two methods with different objectives for handling the latter

case when the required lateral thrust is larger than the available thrust:

1. Only the lateral thrust is bounded to keep the desired vertical thrust.
2. All the thrust is bounded to keep the acceleration directions.

Each method has its applications and can be used depending on the situation.

### 3.5.1 Strategy 1: Keeping the Desired Vertical Thrust

Just merely cutting the lateral input thrust to the $F_{lmax}$ value can result in losing a portion of the vertical thrust, leading to altitude tracking error and a crash in extreme cases. Therefore, an essential objective for handling the lateral thrust can be keeping the vertical thrust at the desired input thrust value so the UAV's altitude still follows the input command.

For this purpose, first, both the vertical and horizontal components of the desired thrust ($\vec{F}_{ver}$ and $\vec{F}_{hor}$, respectively) are projected on (rotated to) the body-fixed axes separately to compute the partial thrust setpoint vectors $\vec{F}^{\mathcal{B}}_{ver}$ and $\vec{F}^{\mathcal{B}}_{hor}$.

We assume that the lateral thrust of $\vec{F}^{\mathcal{B}}_{ver}$ (its components on $\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$ plane) from the vertical component of the desired input thrust is not greater than $F_{lmax}$; otherwise, the vertical desired thrust cannot be achieved. The assumption is reasonable if the robot's tilt is limited (the limit can be set based on the dynamics of the multirotor). After consuming the lateral thrust required for the vertical component, the remaining available lateral thrust for $\vec{F}_{hor}$ will be:

$$F'_{lmax} = F_{lmax} - \sqrt{({}_xF^{\mathcal{B}}_{ver})^2 + ({}_yF^{\mathcal{B}}_{ver})^2} \tag{3.14}$$

Next, the horizontal thrust $\vec{F}^{\mathcal{B}}_{hor}$ is bounded to respect the new lateral thrust limit $F'_{lmax}$:

$$\vec{F}'^{\mathcal{B}}_{hor} = \frac{{F'}_{lmax}}{\sqrt{\left({}_xF^{\mathcal{B}}_{hor}\right)^2 + \left({}_yF^{\mathcal{B}}_{hor}\right)^2}} \vec{F}^{\mathcal{B}}_{hor} \tag{3.15}$$

Finally, the output thrust setpoint is calculated as:

$$\vec{F}^{\mathcal{B}}_{sp} = \vec{F}^{\mathcal{B}}_{ver} + \vec{F}'^{\mathcal{B}}_{hor} \tag{3.16}$$

A different approach taking advantage of the knowledge that the maximum lateral thrust is already being consumed is explained in [112]. The method (explained below) directly constructs the thrust setpoint rather than modifying the result of Equation 3.13.

Let's assume that the angle between the current $\hat{\mathrm{k}}_{\mathcal{B}}$ unit vector and the desired input thrust projected on $\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$ plane is $\gamma$. Knowing that the thrust force on the body-fixed horizontal plane $\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$ is at the maximum, the $x$ and $y$ components of the commanded force $\vec{F}^{\mathcal{B}}_{sp}$ can be directly calculated as:

$$\begin{aligned} {}_xF^{\mathcal{B}}_{sp} &= F_{lmax} \cdot \cos\gamma \\ {}_yF^{\mathcal{B}}_{sp} &= F_{lmax} \cdot \sin\gamma \end{aligned} \tag{3.17}$$

As can be observed in Figure 3.11, the normal component of the thrust setpoint can be calculated by summing the projections of the vertical component of the desired thrust and the lateral thrust of the setpoint on the $\hat{Z}_{\mathcal{B}}$ axis. Therefore, assuming the current tilt $\lambda$ for the robot, we have:

$${}_zF^{\mathcal{B}}_{sp} = {}_zF^{\mathcal{I}}_{des} \sec\lambda - \|{}_{xy}F^{\mathcal{B}}_{sp}\| \tan\lambda = {}_zF^{\mathcal{I}}_{des} \sec\lambda - F_{lmax} \tan\lambda \tag{3.18}$$

Finally, from Equations 3.17 and 3.18 the output thrust command is computed as:

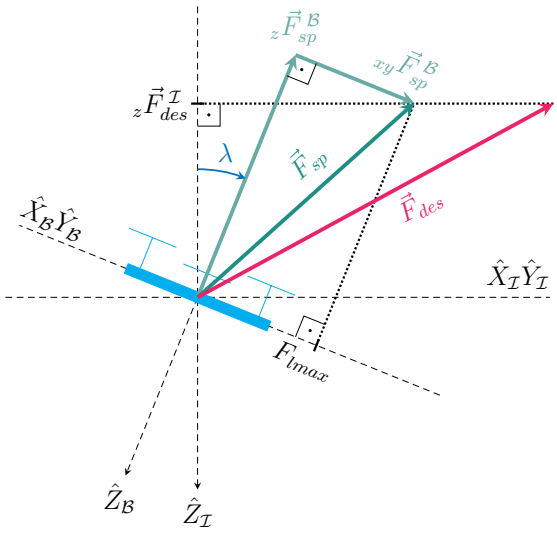



**Figure 3.11:** An illustration of the model for handling the lateral thrust limit. If generating the desired thrust is not feasible, the desired vertical thrust is prioritized to maintain altitude stability. The illustrated vectors $_z\vec{F}_{sp}$ and $_{xy}\vec{F}_{sp}$ are the normal and lateral elements of $\vec{F}_{sp}$ in the *current* body-fixed frame, respectively.

$$\vec{F}^{\mathcal{B}}_{sp} = \begin{bmatrix} {}_xF^{\mathcal{B}}_{sp} \\ {}_yF^{\mathcal{B}}_{sp} \\ {}_zF^{\mathcal{B}}_{sp} \end{bmatrix} \tag{3.19}$$

A more straightforward solution exists if the robot's roll and pitch angles are close to zero, which transforms the problem to directly limiting the input's horizontal thrust. This can be achieved by pre-processing the input desired thrust $\vec{F}^{\mathcal{I}}_{des}$ to create a new thrust vector $\vec{F}'^{\mathcal{I}}_{des}$ with limited *horizontal* thrust before rotating it to the body-fixed frame for thrust setpoint.

Defining $\vec{F}_{hor} = \begin{bmatrix} {}_xF^{\mathcal{I}}_{des} & {}_yF^{\mathcal{I}}_{des} & 0 \end{bmatrix}^\top$, the bounded horizontal thrust can be calculated as:

$$\vec{F}'^{\mathcal{I}}_{hor} = \frac{F_{lmax}}{\left\|\vec{F}_{hor}\right\|}\vec{F}_{hor} \tag{3.20}$$

Replacing the horizontal thrust components with the bounded ones in the original input $\vec{F}^{\mathcal{I}}_{des}$, the new desired thrust can be constructed as:

$$\vec{F}'^{\mathcal{I}}_{des} = \begin{bmatrix} {}_xF'^{\mathcal{I}}_{hor} \\ {}_yF'^{\mathcal{I}}_{hor} \\ {}_z\vec{F}^{\mathcal{I}}_{des} \end{bmatrix} \tag{3.21}$$

The described approach is computationally efficient; however, it can only be used when the assumption of the robot's tilt being close to zero is valid (e.g., when using the zero-tilt attitude strategy described in Section 3.4.1).

### 3.5.2 Strategy 2: Keeping the Acceleration Directions

The previously proposed solution guarantees that the output thrust setpoint has the same vertical component as the input desired thrust if it is feasible. While this tactic prevents undesired altitude changes, it may cause a severe reduction in the available lateral thrust for horizontal motion when the vertical thrust command is large, which may prove dangerous in extreme cases or in situations such as flying in the wind. To avoid those issues, the ratio of the horizontal and vertical accelerations can be maintained, or the horizontal acceleration can be given priority over the vertical acceleration.

Knowing the hover thrust $F_{hov}$ (it can easily be estimated experimentally), if the $z$ component of the input desired thrust is larger than $F_{hov}$, then the hover thrust vector $\vec{F}^{\mathcal{I}}_{hov} = \begin{bmatrix} 0 & 0 & F_{hov} \end{bmatrix}^\top$ is rotated to the current body-fixed attitude to obtain the baseline for zero acceleration:

$$\vec{F}^{\mathcal{B}}_{hov} = \mathbf{R}_{\mathcal{BI}} \cdot \vec{F}^{\mathcal{I}}_{hov} \tag{3.22}$$

Accounting for the consumed lateral thrust by the hover thrust, the remaining lateral thrust $F'_{lmax}$ is computed as:

$$F'_{lmax} = F_{lmax} - \sqrt{\left({}_xF^{\mathcal{B}}_{hov}\right)^2 + \left({}_yF^{\mathcal{B}}_{hov}\right)^2} \tag{3.23}$$

The rest of the input desired thrust ($\vec{F}'^{\mathcal{I}}_{des} = \vec{F}^{\mathcal{I}}_{des} - \vec{F}^{\mathcal{I}}_{hov}$) is used for the output thrust calculation with its lateral bound limited to $F'_{lmax}$. The lateral of the thrust setpoint $\vec{F}'^{\mathcal{B}}_{sp}$ rotated from $\vec{F}'^{\mathcal{I}}_{des}$ using Equation 3.13 is computed as:

$$\left\|\vec{F}_{lat}\right\| = \left\| \begin{bmatrix} {}_xF^{\mathcal{B}}_{sp} \\ {}_yF^{\mathcal{B}}_{sp} \\ 0 \end{bmatrix} \right\|_2 = \sqrt{\left({}_xF^{\mathcal{B}}_{sp}\right)^2 + \left({}_yF^{\mathcal{B}}_{sp}\right)^2} \tag{3.24}$$

If the resulting lateral thrust $\|\vec{F}_{lat}\|$ is larger than $F'_{lmax}$, we can bound the lateral thrust:

$$\vec{F}'^{\mathcal{B}}_{lat} = \frac{F'_{lmax}}{\|\vec{F}_{lat}\|}\vec{F}_{lat} = \frac{F'_{lmax}}{\sqrt{\left({}_xF^{\mathcal{B}}_{sp}\right)^2 + \left({}_yF^{\mathcal{B}}_{sp}\right)^2}} \begin{bmatrix} {}_xF^{\mathcal{B}}_{sp} \\ {}_yF^{\mathcal{B}}_{sp} \\ 0 \end{bmatrix} \tag{3.25}$$

The partial thrust setpoint with bounded lateral thrust is reconstructed as:

$$\vec{F}''^{\mathcal{B}}_{sp} = \begin{bmatrix} {}_xF'^{\mathcal{B}}_{lat} \\ {}_yF'^{\mathcal{B}}_{lat} \\ {}_zF^{\mathcal{B}}_{sp} \end{bmatrix} \tag{3.26}$$

Finally, the result of Equation 3.26 is combined with $\vec{F}^{\mathcal{B}}_{hov}$ from Equation 3.22 to calculate the feasible thrust setpoint $\vec{F}^{\mathcal{B}}_{sp}$.

## 3.6 Extending the Controller for Physical Interaction

Many potential applications specific to the fully-actuated multirotors, e.g., our application of utility pole maintenance, include physical interaction with the environment. These applications range from non-destructive inspections to environment-altering tasks such as wire manipulation or moving objects.

This section describes how the proposed controller design of this chapter can be extended to provide simultaneous position and force control of a multirotor with a rigidly attached end-effector during the physical interaction with the environment. We call the multirotor with a rigidly-attached end-effector as *whole-body wrench generator*.

The end-effector (rigidly attached to the multirotor) is used for physical contact with the environment. Figure 3.18(a) shows the end-effector on the hexarotor used in this project. In addition to the frames described in Section 2.3, there are two more useful frames in this problem:

- The end-effector frame $\mathcal{F}^{\mathcal{E}}$ which is attached to the end-effector as illustrated in Figure 3.12(a). Since the end-effector is rigidly attached to the UAV, its frame is fixed in the body-fixed frame.
- The contact frame $\mathcal{F}^{\mathcal{C}}$ which is attached to the contact point as illustrated in Figure 3.12(b). This frame moves with the contact point during the interaction, with its $\hat{Z}_{\mathcal{C}}$ axis always being normal to the surface.

Our assumptions in this section are:

1. The contact surface is rigid, and the forces applied by the UAV are non-destructive and non-altering.
2. The desired positions and forces are feasible for the UAV.
3. There is direct feedback from the end-effector of the UAV. This feedback can be achieved by devising a force/torque sensor at the end-effector.
4. There are no rotational constraints on the robot's motion at the point of contact.

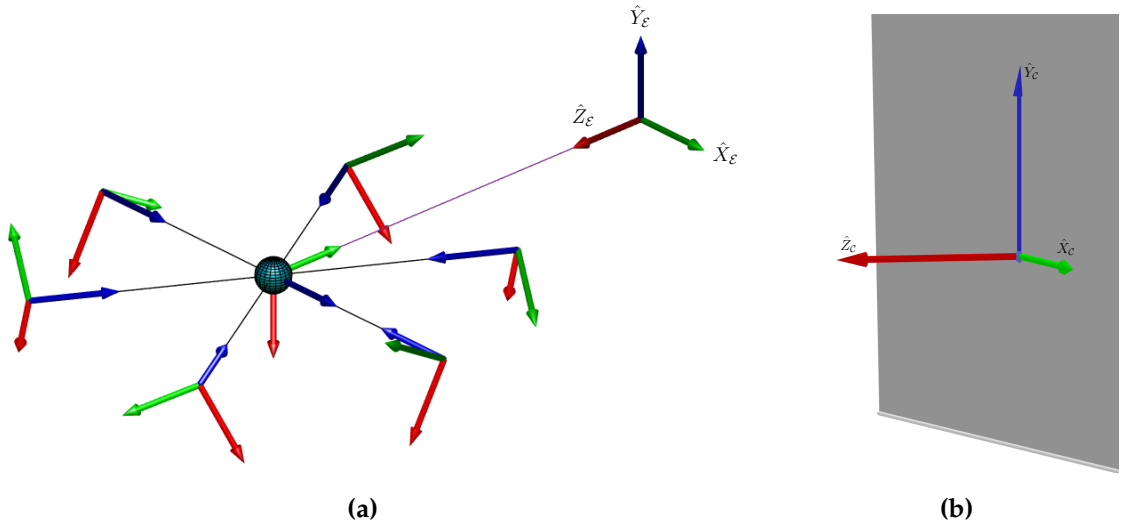



**Figure 3.12:** Coordinate frames used in a physical interaction: (a) End-effector. (b) The contact point on the surface.

### 3.6.1 Hybrid Position and Force Controller

We implemented a hybrid position-force controller (HPFC) to control both the position and the applied force during contact with a planar surface. The subspace affected by each one at the point of contact is separated using two $3 \times 3$ matrices called *selection matrices* to achieve independent control over the force and the position. Each row represents one of the 3-DoFs of space at the contact point.

The position selection matrix $\mathbf{S_p}$ defines the directions in the contact frame that are free to move, and the force selection matrix $\mathbf{S_f}$ defines the directions in which a force can be applied. The definition of $\mathbf{S_p}$ and $\mathbf{S_f}$ depends on two types of constraints: the natural constraints, which are due to the environment's geometry, and the artificial constraints, which depend on the task. For example, the end effector cannot move into the wall (along $-\hat{Z}_\mathcal{C}$), creating a natural constraint, and if the motion along $\hat{X}_\mathcal{C}$ is restricted based on the task, that is an artificial constraint.

The selection matrices are square diagonal matrices with only $0$ or $1$ elements, and if there are only natural constraints, the two matrices are complements:

$$\mathbf{S_p} = \mathbf{I_{3\times 3}} - \mathbf{S_f} \tag{3.27}$$

where $\mathbf{I_{3\times 3}}$ is the $3 \times 3$ identity matrix.

For example, if only natural constraints are present when facing a planar surface, the selection matrices would be as follows:

$$\mathbf{S_p} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{S_f} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.28}$$

When artificial constraints are added, some non-zero diagonal elements become zero. Therefore, the Hadamard product of the two selection matrices will always stay zero:

$$\mathbf{S_p} \odot \mathbf{S_f} = \mathbf{0_{3\times 3}} \tag{3.29}$$

where $\mathbf{0_{3\times 3}}$ is the $3 \times 3$ zero matrix.

The selection matrices can be defined based on the task before the execution. As mentioned, they are utilized to separate the subspace for the applied force and the position at the contact frame. The contact frame $\mathcal{F}^{\mathcal{C}}$ (i.e., the rotation $\mathbf{R}_{\mathcal{IC}}$) can be computed from the normal of the contact surface and is arbitrary as long as it is consistent with the devised selection matrices.

We add a new Force Controller module to control the force during the contact. The module's inputs are the desired force to apply in the contact frame ($\vec{F}^{fc}_{des}$), the environment information that includes the contact frame ($\mathbf{R}_{\mathcal{IC}}$), and the state information that includes the force feedback in the end-effector frame measured by the force sensor. Like the Position Controller module, the output is the force in the inertial frame, which is now combined with the Position Controller's output force based on the subspaces before feeding into the Thrust Setpoint Generator module. Figure 3.13 illustrates the hybrid position-force

controller architecture developed based on our free-flight controller of Figure 3.4.

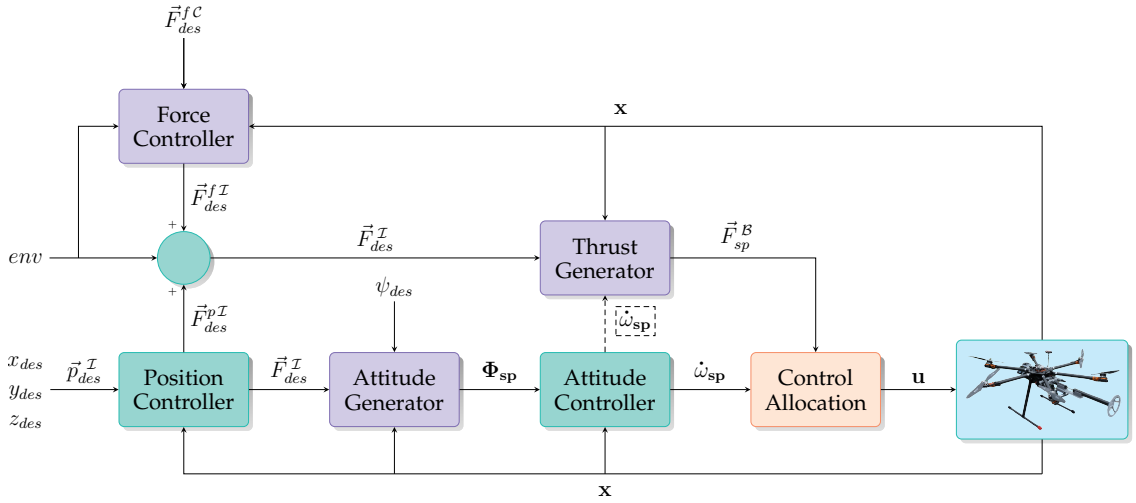



**Figure 3.13:** The design of our Hybrid Position-Force controller extended based on the controller architecture of Section 3.3. The force and position control modules independently calculate the necessary thrusts (accelerations) to achieve the desired inputs, then are combined based on their respective subspaces.

Note that the force controller computes the desired force in the $\mathcal{F}^{\mathcal{C}}$ frame, while the measured force feedback is in the $\mathcal{F}^{\mathcal{E}}$ frame. So, independent of the force control method, the force feedback should be transformed into the contact frame. Finally, after the output is calculated in the contact frame, the output is transformed into the inertial frame.

We used a PID controller to follow the reference force in our work. Since both the input and the desired output are of the same type, the PID loop generates changes to the last output and not the output itself.

Finally, the selection matrices are applied to the outputs of force and position controllers ($\vec{F}^{f\mathcal{I}}_{des}$ and $\vec{F}^{p\mathcal{I}}_{des}$, respectively) in the contact frame and the results are combined together to obtain the input for the Thrust Setpoint Generator module:

$$\vec{F}^{\mathcal{I}}_{des} = \mathbf{R}_{\mathcal{IC}} \left( \mathbf{R}_{\mathcal{CI}}\ \mathbf{S_p}\ \vec{F}^{p\mathcal{I}}_{des}\ +\ \mathbf{R}_{\mathcal{CI}}\ \mathbf{S_f}\ \vec{F}^{f\mathcal{I}}_{des} \right) \tag{3.30}$$

Note that, since the Hadamard product of the two selection matrices is always zero, they transform $\vec{F}_{des}^{p\mathcal{I}}$ and $\vec{F}_{des}^{f\mathcal{I}}$ to orthogonal subspaces. Therefore, the outputs of position and force controllers can only affect their respective subspaces and do not affect each other, completely decoupling the force from the position.

## 3.7 Experiments and Results

### 3.7.1 Hardware and Software

We have tested the proposed controller both in simulation and on real robots.

Three simulation environments were devised for testing the new fully-actuated UAV development and the methods described in this section:

1. **MATLAB simulator:** We developed a complete simulator for fully-actuated UAV controller development in MATLAB, which allows us quickly define, analyze and visualize new architectures and control methods. Most of the analysis of different architectures and methods in this chapter is done with this simulator. Figure 3.14 shows a snapshot of our multirotor flying in this simulation environment.
2. **Gazebo simulator with PX4 SITL:** Our UAV model is developed in the Gazebo simulator, and the SITL simulation provided by the base PX4 firmware is enhanced in our code to support fully-actuated vehicles. Considering that the code-base in this simulation is the real robot's firmware, it is used for testing the code developed on the autopilot before performing tests on the real robot. Additionally, it is used for testing the onboard software developed for different missions and tasks. Figure 3.15 shows the screenshot of this simulation environment.
3. **MATLAB Simulink model:** A Simulink model of the controller for our hexarotor is designed for faster simulation when required. A library is designed to allow faster model changes to test new ideas and architectures.

Figures 3.16 and 3.17 show the overall multirotor and controller models, as well as the developed library and a screenshot of the visualization.

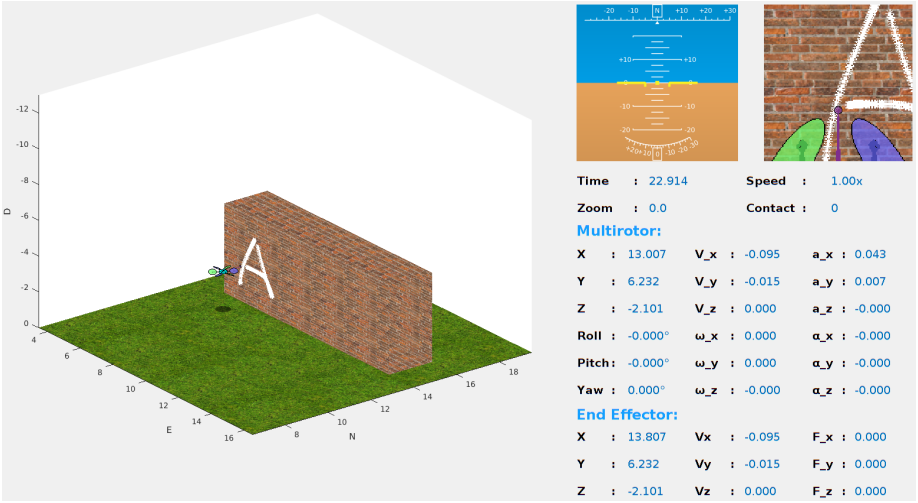



**Figure 3.14:** A screenshot of the MATLAB simulator for the fully-actuated hexarotor used in our experiments. This environment is used to develop, analyze, and test new ideas, architectures, and control methods.

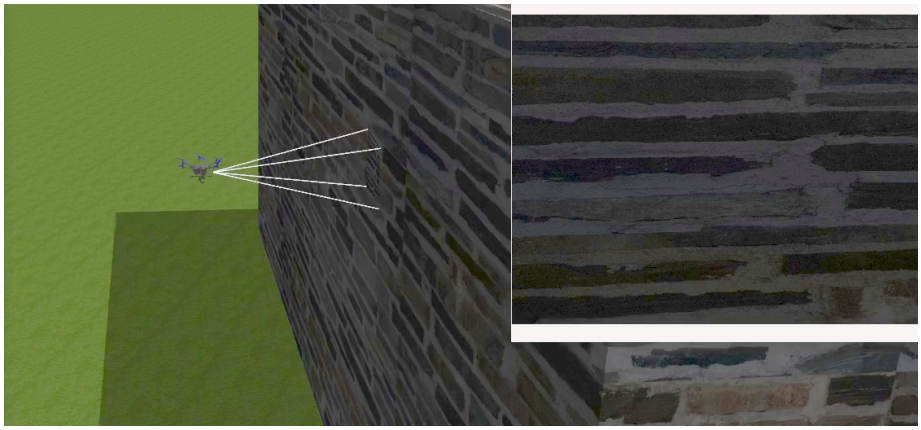


**Figure 3.15:** A screenshot of the PX4 SITL Gazebo simulator with the fully-actuated hexarotor used in our experiments. This environment is used for firmware development and testing the autopilot before deploying the code on the real UAV.

Some fixed-tilt hexarotors with two different frame sizes but similar designs are built and tested. All rotors on the robots are rotated sideways, alternatively

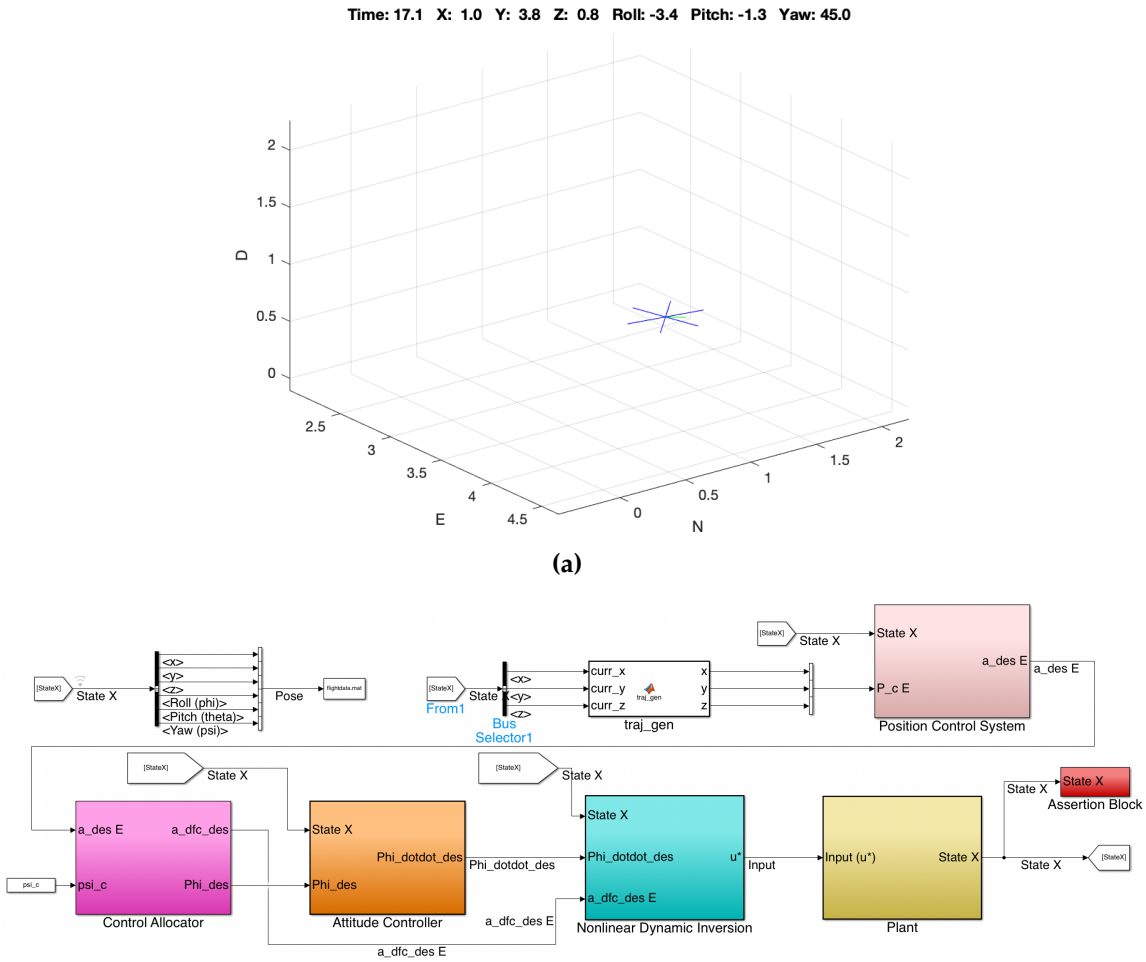
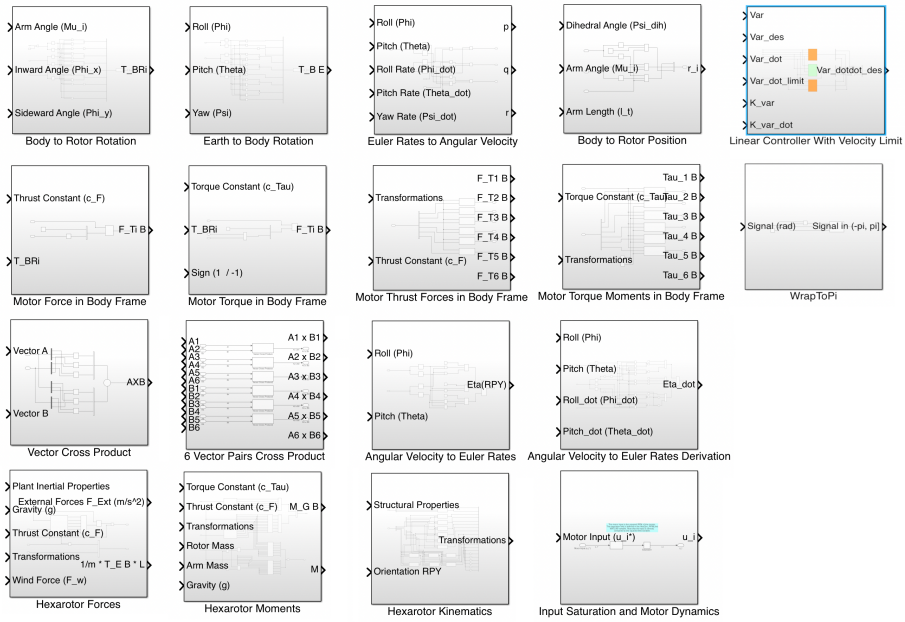


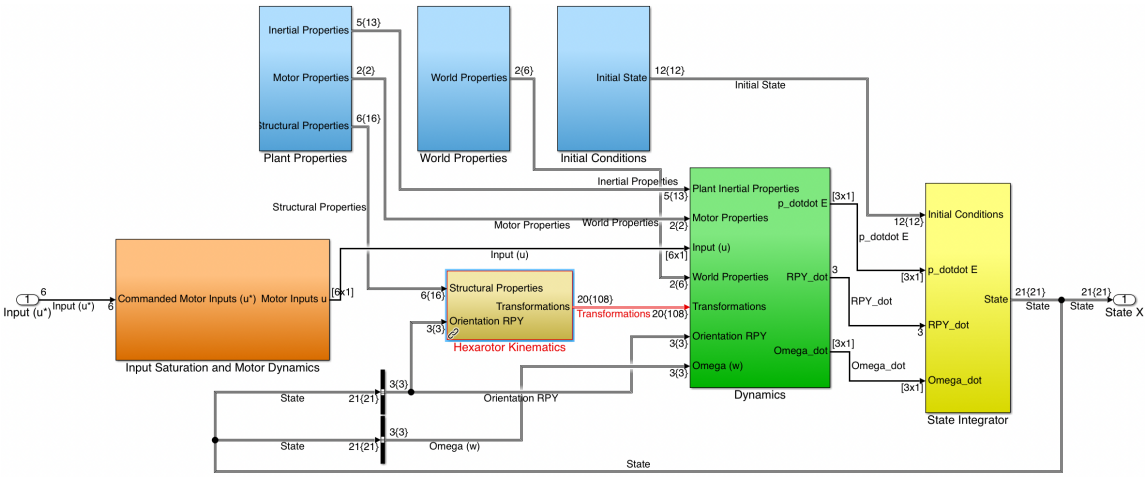

**Figure 3.16:** The Simulink simulation developed for testing the controller of the fully-actuated hexarotor used in our experiments. (a) A snapshot of the visualization of a trajectory following simulation. (b) The overall model of our controller and simulation.

for 30, and -30 degrees, similar to the design described in [112].

The main body frame for the larger robot design is Tarot T960 with KDE-3510XF-475 motors, KDE-UAS35HVC electronic speed controllers (ESCs), and 14-inch propellers. It is equipped with a mRo Pixracer autopilot, an Nvidia Jetson TX2 onboard computer, a u-Blox Neo-M8N GPS module, Futaba T10J transmitter/receiver, and a 900 MHz radio for communication with the Ground Control Station computer. Figure 3.18(a) shows the larger hexarotor.




**(a)**




**(b)**

**Figure 3.17:** The Simulink library developed for testing the controller of the fully-actuated hexarotor used in our experiments. (a) The library implemented to enable rapid testing of different architectures and controllers. (b) The fully-actuated hexarotor model made with our library.

The smaller robot is built on the Tarot X6 frame with a 0.96 $[m]$ motor-to-motor diameter and a maximum payload of 7.5 $[kg]$. The frame can be size-adjusted for specific tasks. Propulsion is achieved with six KDE Direct

KDE-4215XF-465 brushless motors and KDEXF-UAS55HVC electronic speed controllers (ESCs). It is equipped with the same flight controller, onboard computer, and GPS module as the larger UAV. Additionally, Futaba T8J and T10J transmitters/receivers are used for the pilot's manual control. Figure 3.18(b) shows this smaller hexarotor.

(a) (b)

**Figure 3.18:** The fixed-pitch hexarotors used in our experiments. (a) The larger design with Tarot T960 frame. (b) The smaller design with Tarot X6 frame.

For the outdoor tests, the UAVs are also equipped with Intel T265 RealSense cameras for visual odometry and D435 RealSense cameras for RGB and depth imaging.

An Opti-Track system is used for pose estimation in indoor tests, which requires several reflective markers attached to the UAV.

Additionally, for the force and hybrid motion-force control tests, the ATI Gamma Force/Torque sensor with the Digital Interface [7] is attached to our robot to measure the forces applied at the end-effector. Tool transformation ensures the correct force/torque measurements at the end-effector instead of the sensor. Figure 3.19(a) shows the sensor and how it is attached to the UAV.

The onboard computers on the robots run Linux Ubuntu 18.04 (Bionic Beaver) with Robot Operating System (ROS) Melodic Morenia. Depending on the task at hand, different software packages run to plan and control the missions and trajectories.

(a) (b)

**Figure 3.19:** (a) Illustration of the ATI Gamma Force/Torque Sensor used for the force control tests [7]. (b) Our fully-actuated multirotor with the force/torque sensor attached.

We extended the PX4 v1.11.0 firmware to support our fully-actuated vehicles and implemented the methods presented in this section. Figure 3.4 illustrates the controller architecture of our developed firmware.

### 3.7.2 Experiments

We have performed tens of indoor and outdoor flights with our fully-actuated hexarotor platform running the proposed controller and strategies. The experiments include real and simulated flights in free flight and during physical contact with the environment. Figure 3.20 shows our UAV during contact with a wall to measure the properties of the contact point using an ultrasonic sensor. The contact is unmodeled, but the UAV can keep its zero-tilt attitude and reject the disturbances.

#### Free Flight Experiments

The proposed controller design is tested in both the MATLAB and Gazebo simulations and on our robot.

**Figure 3.20:** A fully-actuated hexarotor using the PX4 controller extended with our proposed design making an unmodeled contact with the wall.

Figure 3.21 shows the attitude and position responses of our fixed-pitch hexarotor model in the developed MATLAB simulator. After tuning the underlying PID controller gains, we were able to get good responses to the commands.

Figure 3.22 shows the position responses of a fully-actuated octorotor with four co-planar upward rotors and four auxiliary motors perpendicular to the main rotors modeled in our MATLAB simulator. The response shows similar results to the fixed-pitch hexarotor of our project.

All the proposed attitude strategies have been implemented in simulation and for the real robot.

We used the same trajectory with two waypoints to simulate all five attitude strategies in our MATLAB simulator to compare the results. The starting point is inertial zero with a zero attitude. The waypoints are $\begin{bmatrix} 2 & 2 & -4 & 0 \end{bmatrix}^\top$ and $\begin{bmatrix} 2 & 6 & -3 & 30 \end{bmatrix}^\top$, respectively. The waypoints' elements are $x$, $y$, $z$, and $yaw$ in degrees. Our project's hexarotor with fixed-pitch arms is used in all the trials.

Figure 3.23 presents the attitude and position plots for zero-tilt strategy. The tilt in this strategy is always zero to keep the robot horizontally level. As a result, when the desired acceleration is high, it cannot be achieved.

Figure 3.24 shows the attitude plots for full-tilt and minimum-tilt strategies

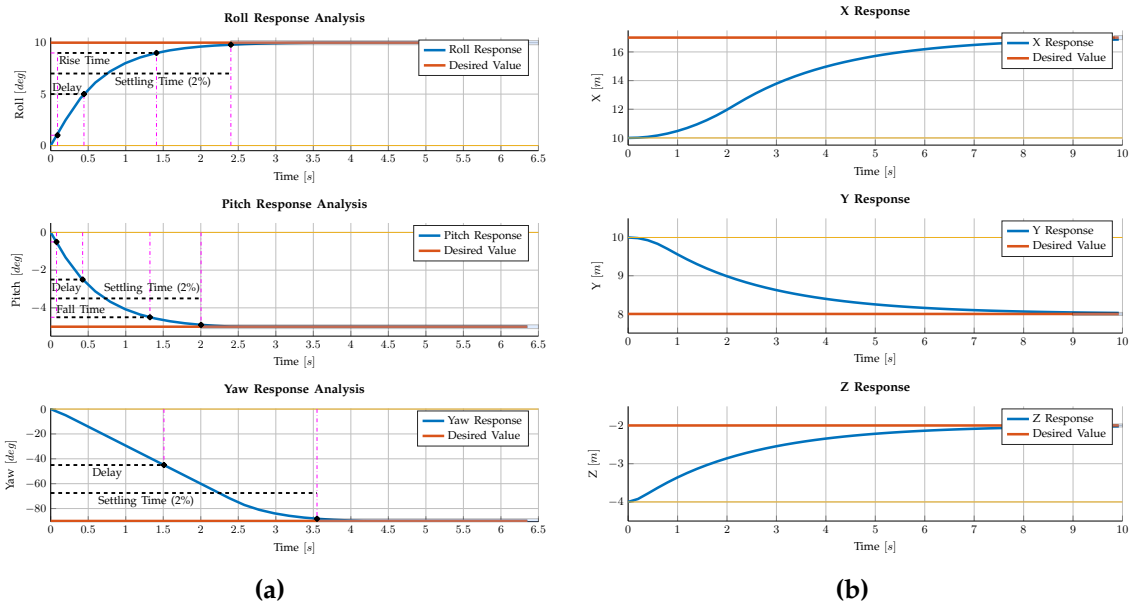



**Figure 3.21:** Responses of the fixed-tilt hexarotor used in this project to attitude and position commands simulated in the MATLAB simulator. (a) Attitude response to $\begin{bmatrix} 10 & -5 & -90 \end{bmatrix}^\top$ roll-pitch-yaw command starting from $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top$. (b) Position response to $\begin{bmatrix} 17 & 8 & -2 & -45 \end{bmatrix}^\top$ x-y-z-yaw command starting from $\begin{bmatrix} 10 & 10 & -4 & 0 \end{bmatrix}^\top$.

for the same trajectory. The tilt direction in the full-tilt strategy is always towards the direction of acceleration. The minimum-tilt strategy tends to keep the tilt at zero when the acceleration is low, but the robot starts tilting towards the acceleration direction when the command is higher. Therefore, the tilt is generally lower than in the full-tilt strategy. However, the acceleration is higher than in the zero-tilt strategy and is similar to the full-tilt strategy.

Figure 3.25 shows the attitude plots for fixed-tilt and fixed-attitude strategies for the same trajectory. The tilt's direction is set to the north in the fixed-tilt strategy, with an $8$-degree tilt. The UAV tends to keep the tilt angle and direction the same, even during and after the turn. The fixed-attitude strategy has $7$ degrees of roll and $-4$ degrees of pitch.

Three of the strategies have been tested on the real robot as well. Figure 3.26 shows the robot keeping its zero-tilt attitude while aggressively flying and turning.

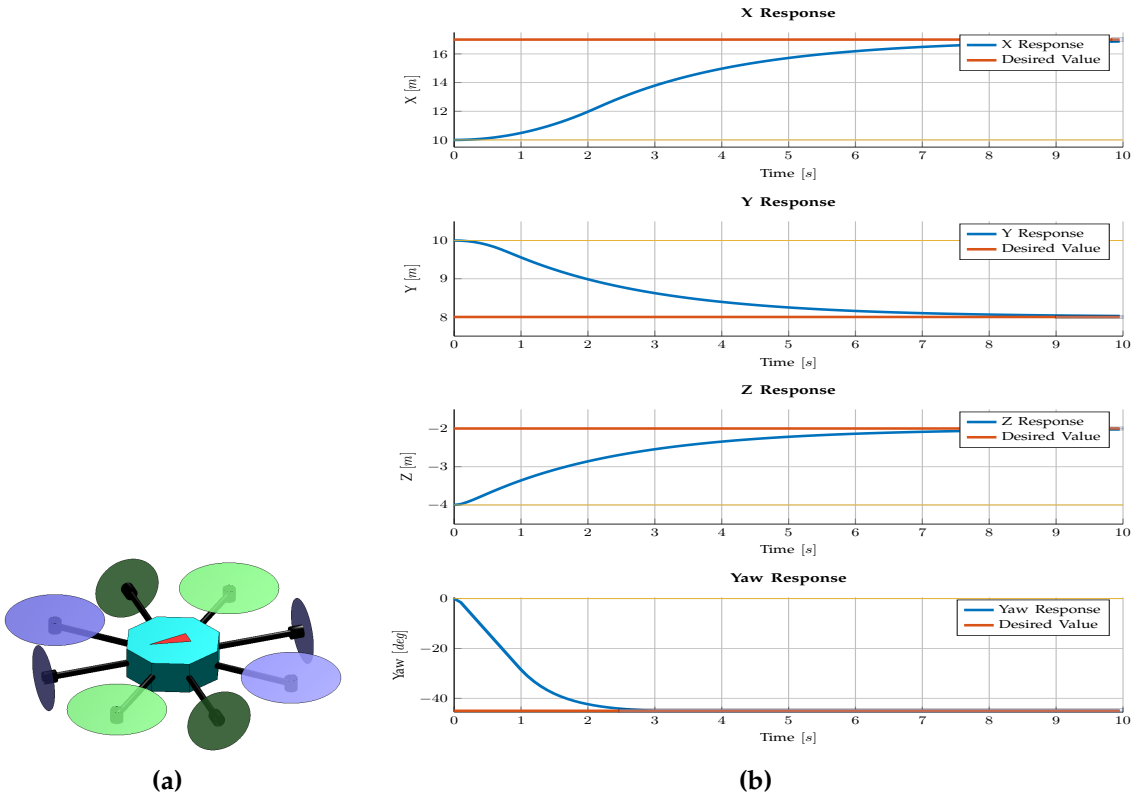


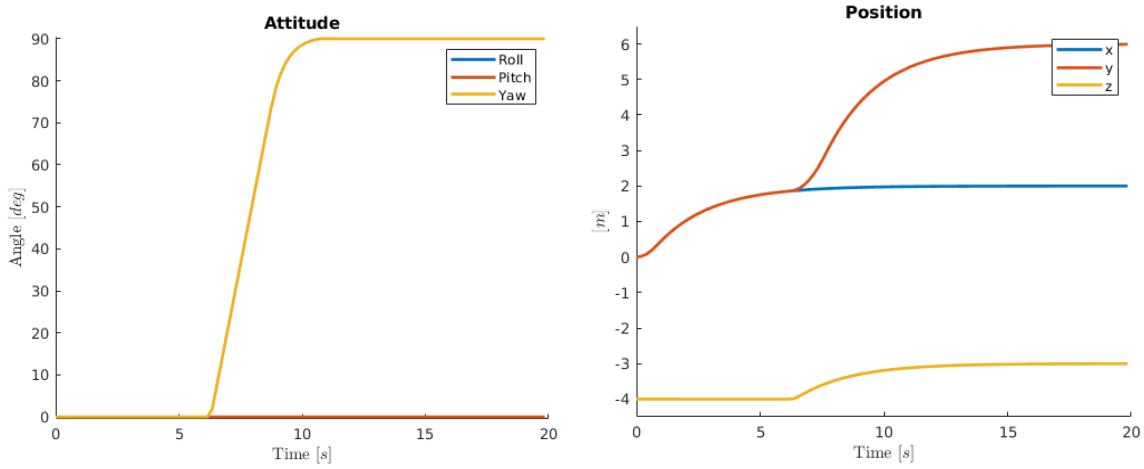

**Figure 3.22:** (a) A fully-actuated octorotor with four co-planar upward rotors and four auxiliary motors perpendicular to the main rotors. (b) The position-yaw response to $\begin{bmatrix}17 & 8 & -2 & -45\end{bmatrix}^\top$ x-y-z-yaw command starting from $\begin{bmatrix}10 & 10 & -4 & 0\end{bmatrix}^\top$.

Figure 3.27 plots another flight segment flying with the fixed-tilt attitude strategy in a strong and gusty wind. The figure illustrates the tilt angle staying almost constant while the multirotor performs aggressive motions.

### Physical Interaction Experiments

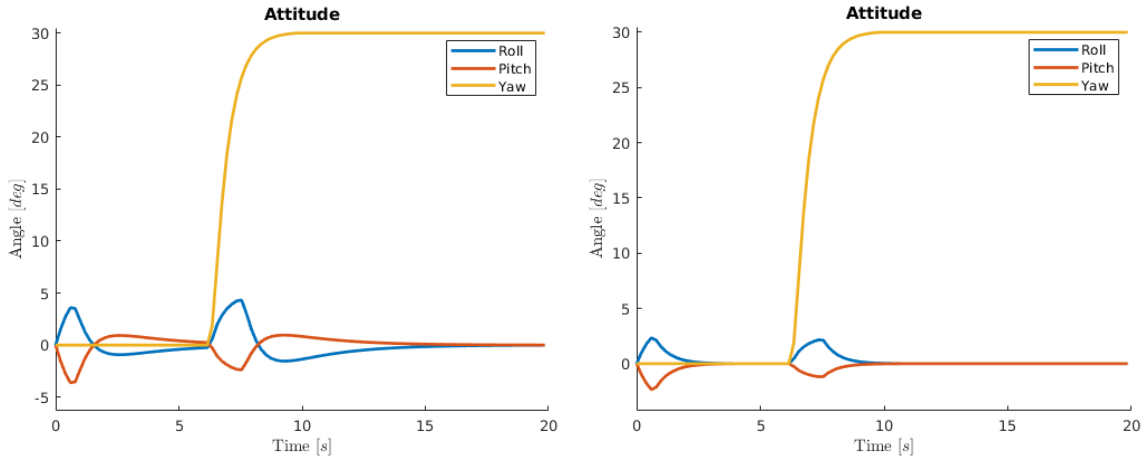
We used the MATLAB and Gazebo simulators and our hexarotor UAV with tilted arms (see Figure 3.7.1) to perform the tests for the proposed position-force controller.

For the MATLAB simulations, a scalar constant $\mu$ defines the friction at the surface. The friction force on the $\hat{X}_C\hat{Y}_C$ plane is $\mu\left(\vec{F}_a^C \cdot \hat{\vec{k}}_C\right)$ in the opposite




(a) (b)

**Figure 3.23:** The MATLAB simulation of the trajectory followed by the fixed-pitch hexarotor of our project using the zero-tilt attitude strategy. The tilt in this strategy is always zero to keep the robot horizontally level.

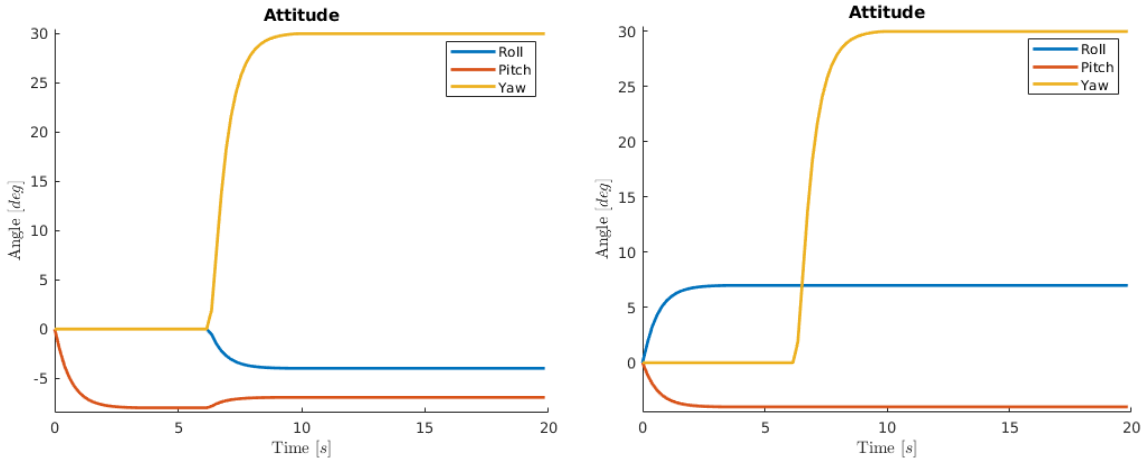



(a) (b)

**Figure 3.24:** The MATLAB simulation of the trajectory followed by the fixed-pitch hexarotor of our project using the (a) full-tilt attitude strategy. (b) The minimum-tilt attitude strategy.

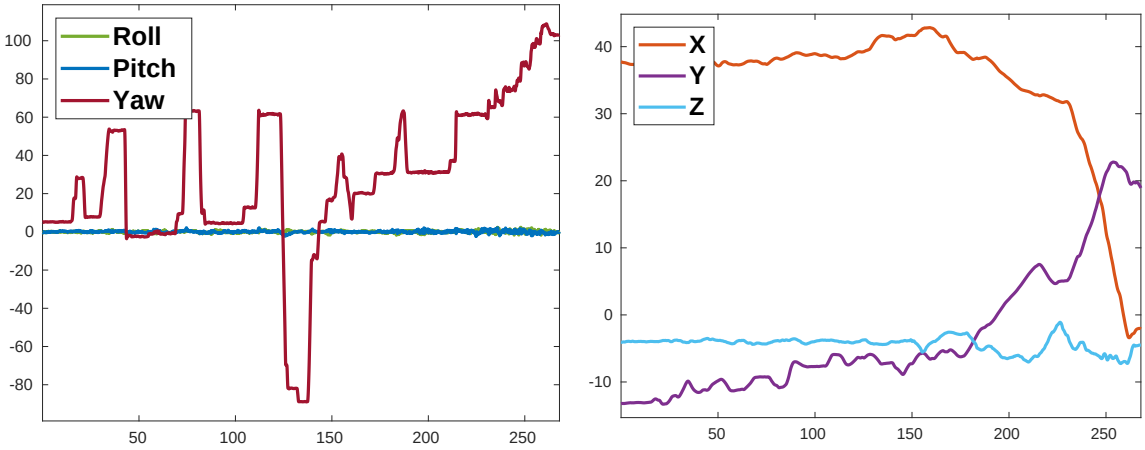
direction of motion, where $\overrightarrow{F}_a^{\mathcal{C}}$ is the applied force by the end effector in contact frame and $\hat{\vec{k}}_{\mathcal{C}}$ is the unit vector in the $\hat{Z}_{\mathcal{C}}$ direction.

We experimented with multiple attitude strategies for our fixed-pitch multirotor interacting with a straight wall.




(a) (b)

**Figure 3.25:** The MATLAB simulation of the trajectory followed by the fixed-pitch hexarotor of our project using (a) The fixed-tilt attitude strategy. (b) The fixed-attitude strategy.




(a) (b)

**Figure 3.26:** An outdoor flight segment with the zero-tilt strategy in the presence of winds and gusts. The yaw changes aggressively, and the multirotor is flying around while the roll and pitch stay close to zero.

Figures 3.28, 3.29 and 3.30 show the simulation of the fixed-pitch hexarotor with the proposed hybrid position-force controller applying a 5 $[N]$ normal force to a straight wall using zero-tilt, full-tilt and fixed-orientation attitude strategies (see Section 3.4).

Figure 3.31 shows the same fixed-pitch hexarotor painting on the wall. The

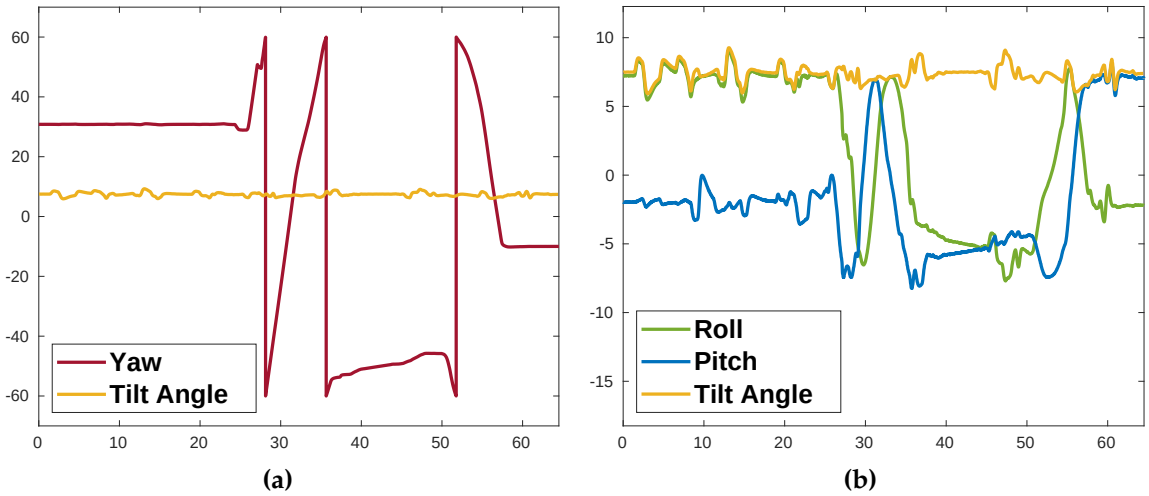



**Figure 3.27:** An outdoor flight segment with the fixed-tilt strategy in the presence of winds and gusts. The tilt is locked around 7.5 degrees, while the yaw changes aggressively. The right plot shows that while the tilt angle is constant, the roll and pitch change when the yaw changes. The yaw is scaled by $1/3$ in the plot.

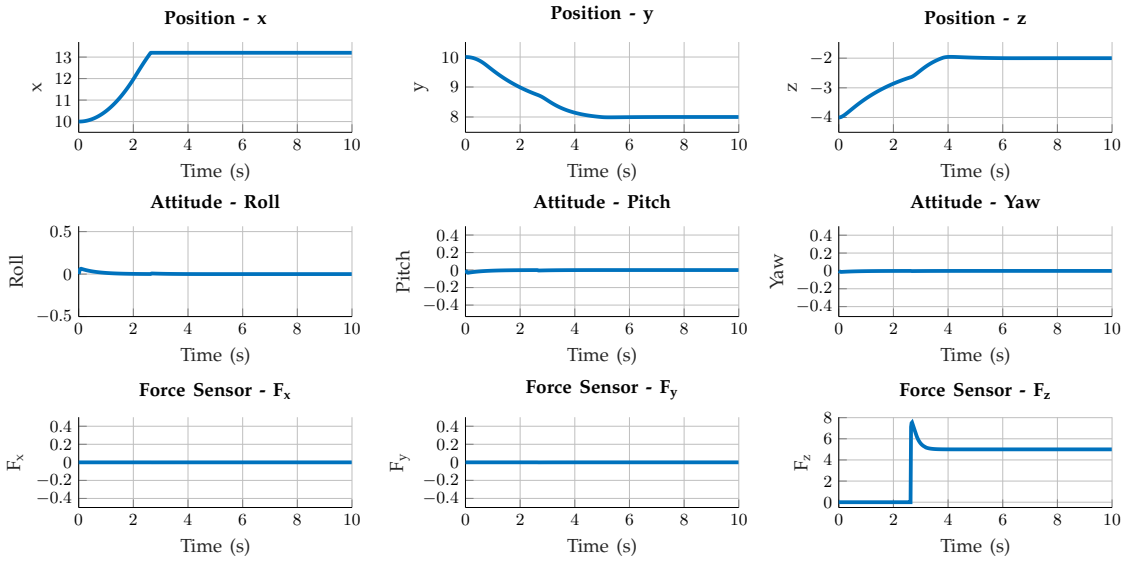



**Figure 3.28:** The Hybrid Position-Force controller with the zero-tilt attitude strategy applying 5 $[N]$ normal force to the wall at point $p = \begin{bmatrix} 14 & 8 & -2 \end{bmatrix}^\top$.

paint is only released on the wall when the multirotor's applied force is within 0.1 $[N]$ error from 5 $[N]$. The wall friction is $\mu = 0.1$.

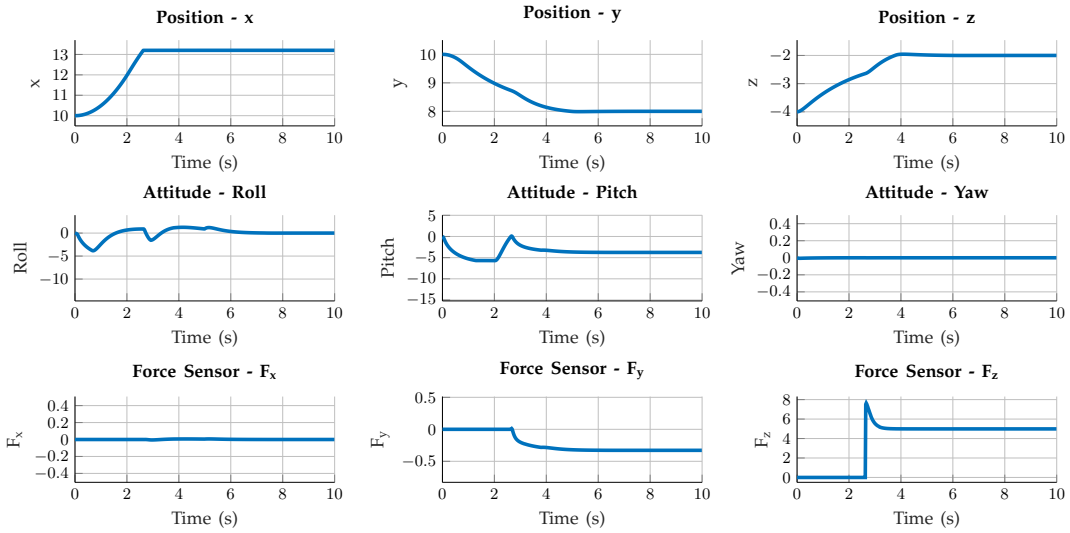



**Figure 3.29:** The Hybrid Position-Force controller with the full-tilt attitude strategy applying 5 $[N]$ normal force to the wall at point $p = \begin{bmatrix} 14 & 8 & -2 \end{bmatrix}^\top$.

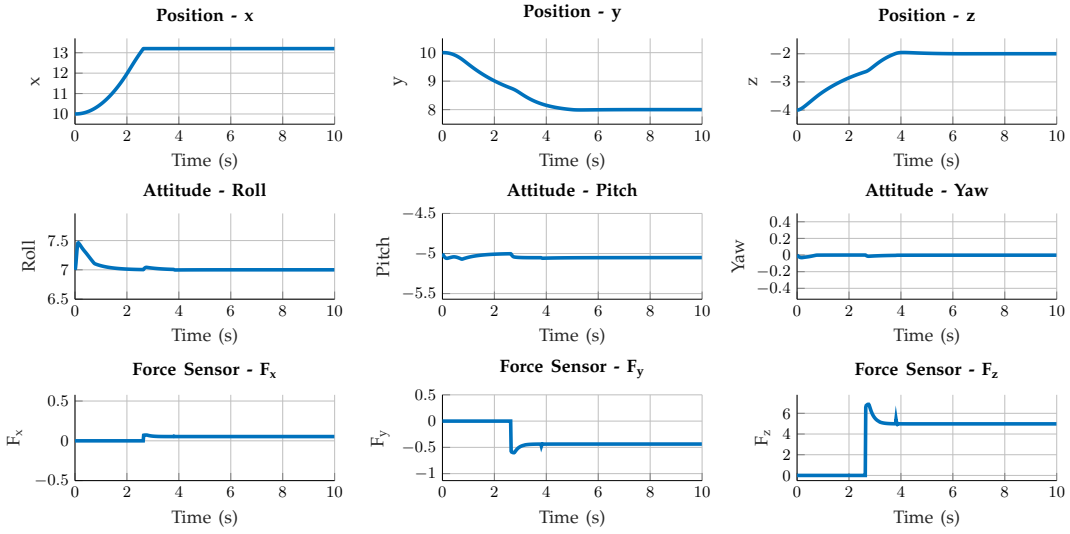
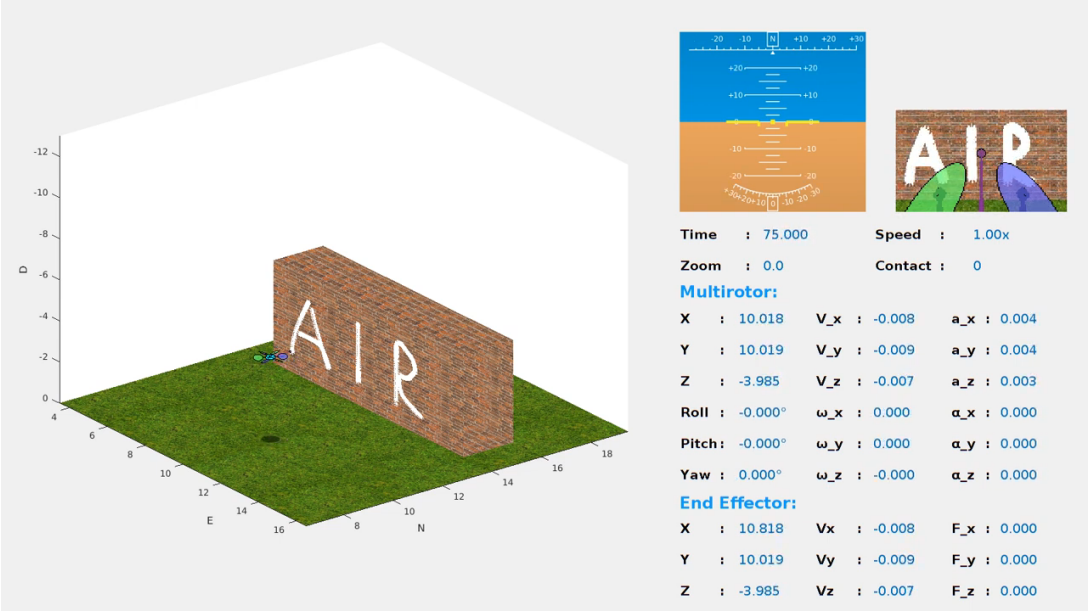
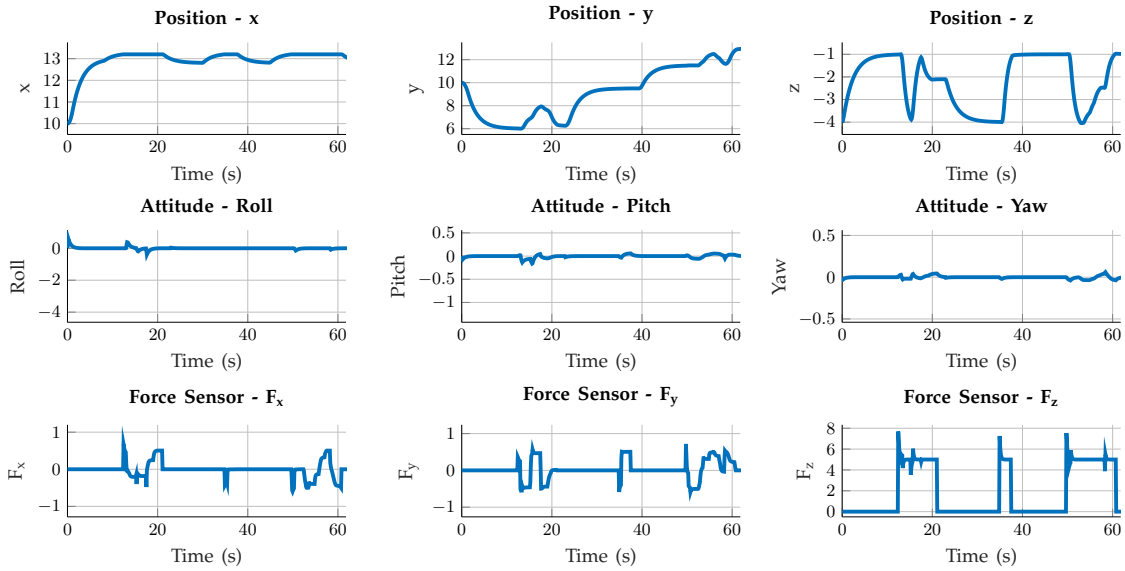



**Figure 3.30:** The Hybrid Position-Force controller with the fixed-attitude strategy applying 5 $[N]$ normal force to the wall at point $p = \begin{bmatrix} 14 & 8 & -2 \end{bmatrix}^\top$. The robot's roll and pitch are set to 7 and $-5$ degrees, respectively.

To test the robustness of the HPF controller of Section 3.6 with respect to imperfect knowledge of the environment, we numerically tested contact with walls with different slopes and angles while the controller thinks that it is




**(a)**




**(b)**

**Figure 3.31:** The Hybrid Position-Force controller with the zero-tilt strategy applying 5 $[N]$ normal force to the wall to paint characters 'AIR'. (a) The screenshot from the MATLAB simulator. (b) The position, attitude and applied force plots.

contacting a straight wall. Figure 3.32 shows the setup for the tests.

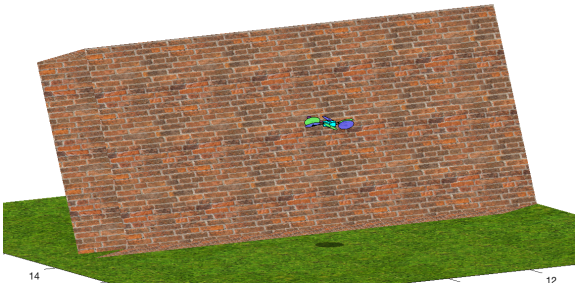
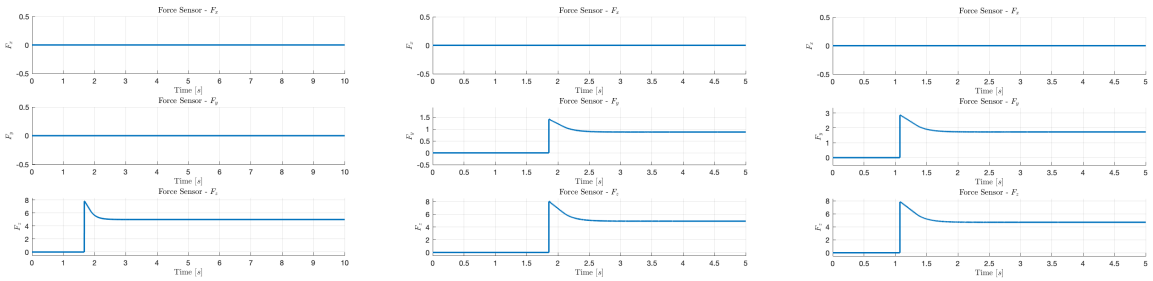


**Figure 3.32:** The sloped wall pitched back 20 degrees with the multirotor contacting it to regulate 5 $[N]$ force, assuming that it is a straight wall.

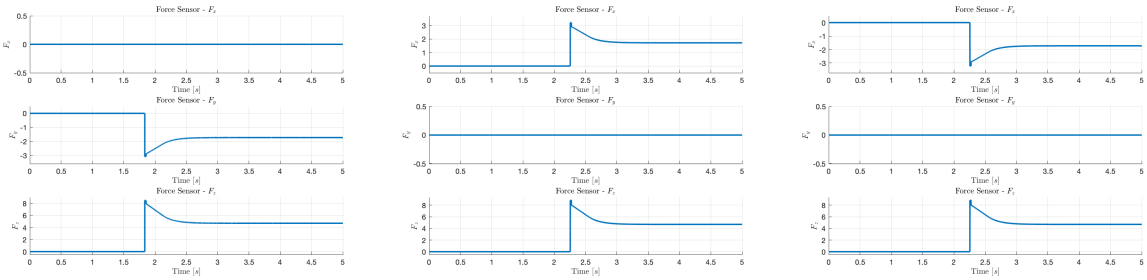
In all experiments, the assumption given to the controller is that the wall normal (i.e., the $\hat{Z}_C$ of contact frame) is parallel to the ground, and it is aligned with the current robot's yaw. The robot is asked to apply a steady 5 $[N]$ force at a specific point on the wall in the contact frame's $\hat{Z}_C$ direction. However, in each experiment, the wall is either pitched (forward or backward) or has a different yaw angle without the controller's knowledge. Figure 3.33 shows the force response of the HPF controller to walls with 10 and 20 degrees pitch back. As can be seen, the controller can still regulate the 5 $[N]$ force along its assumed $\hat{Z}_\mathcal{E}$ (which it assumes is almost the opposite of $\hat{Z}_\mathcal{E}$). However, the application of force along that direction results in forces being inadvertently applied in its $\hat{Y}_\mathcal{E}$ direction as well.



**(a)** **(b)** **(c)**

**Figure 3.33:** Hybrid Position-Force controller applying 5 $[N]$ force in contact frame's $\hat{Z}_C$ direction on walls with different pitch slopes with imperfect contact knowledge assuming that the walls are straight. (a) Straight wall (as baseline). (b) Wall with -10 degrees pitch. (c) Wall with -20 degrees pitch.

Figure 3.34 shows the force response of the HPF controller to a wall with 20 degrees pitch forward and walls rotated at 20 degrees to the left and right. As can be seen, the controller can still regulate the 5 $[N]$ force along its assumed $\hat{Z}_{\mathcal{C}}$ (which it assumes is almost the opposite of $\hat{Z}_{\mathcal{E}}$). However, the application of force along that direction results in forces being inadvertently applied in its $\hat{Y}_{\mathcal{E}}$ and $\hat{X}_{\mathcal{E}}$ directions as well.



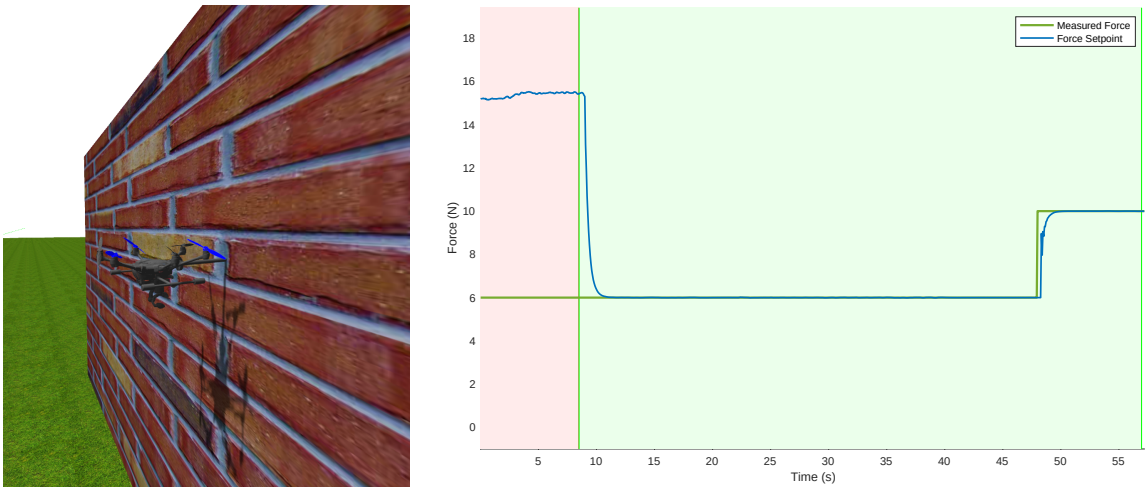
(a) (b) (c)

**Figure 3.34:** Hybrid Position-Force controller applying 5 $[N]$ force in contact frame's $\hat{Z}_{\mathcal{C}}$ direction on walls with different pitch slopes and yaw rotations with imperfect contact knowledge assuming that the walls are straight. (a) Wall with +20 degrees pitch. (b) Wall with -20 degrees yaw rotation. (c) Wall with +20 degrees yaw rotation.

To test our HPF controller implementation on the real robot, we first tested it in Gazebo. We equipped our hexarotor Gazebo model with a rigidly-attached arm and tested it with a straight wall. Figure 3.35 shows the setup and the response of the applied force to the setpoints when the controller is turned on.

We tested the implemented force controller on our UAV. The experiments included testing different force setpoints during contact with a whiteboard while the position is fixed.

Figure 3.36 shows an example experiment with the desired force setpoint of 10 $[N]$ to be applied in the $\hat{Z}_{\mathcal{E}}$ direction. The plot illustrates the difference between when the force is not controlled (area shaded in red) and when it is controlled (area shaded in green). Table 3.1 shows the statistical information for the measured force in free flight, uncontrolled contact and force-controlled contact. It signifies that during the controlled contact, the force error is similar




**(a)** **(b)**

**Figure 3.35:** Hybrid Position-Force controller applying 6 $[N]$ and 10 $[N]$ forces in contact frame's $\hat{Z}_C$ direction on a straight wall. (a) A screenshot of the setup. (b) In response to the controller switching on, the measured force initially controls the 6 $[N]$ force and then switches to 10 $[N]$. The area shaded in red shows when the end-effector is in contact with the wall, but the force controller is inactive. The area shaded in green shows the times when the hybrid force-position controller is active.

to the sensor's natural noise (measured during free-flight), illustrating the effectiveness of the force controller in controlling the applied force as exactly as possible given the sensor's characteristics.

Table 3.1: Statistical comparison of the measured force (in $[N]$) during the point contact example of Figure 3.36.

| | Mean | Std. Dev. | Min | Max |
|---|---|---|---|---|
| Free Flight | 0.07 | 0.72 | -2.26 | 9.70 |
| Uncontrolled Contact | 11.25 | 12.48 | 0.14 | 39.40 |
| Force-Controlled Contact | 10.07 | 0.82 | 8.12 | 13.32 |

We further performed experiments to test the complete hybrid force-position controller described in Section 3.6. Figure 3.37 shows an example experiment with the desired force setpoint of 6 $[N]$ to be applied in the $\hat{Z}_E$ direction while the robot is writing the letter "A" on the whiteboard. The plot illustrates the

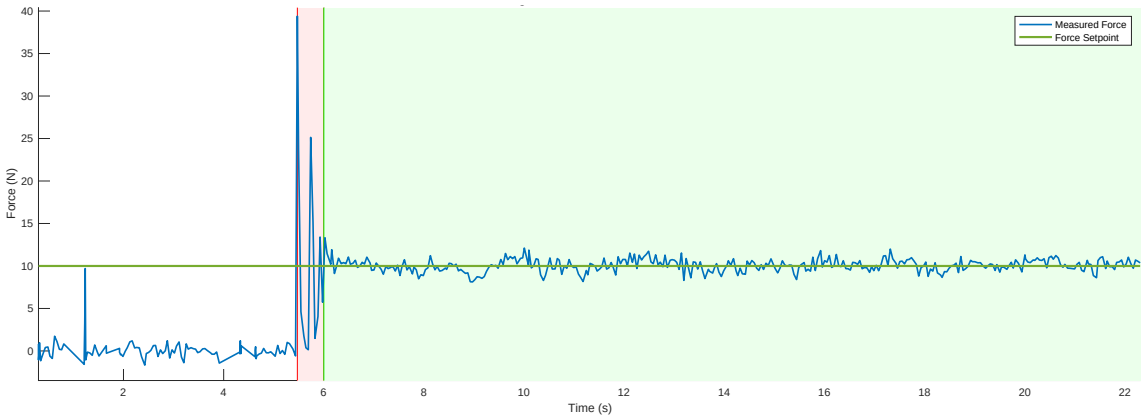


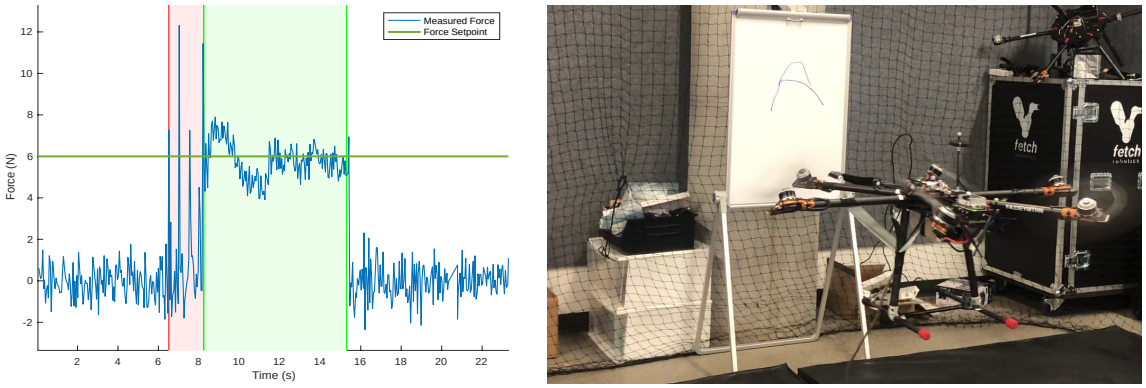

**Figure 3.36:** A force control experiment on the UAV during physical interaction with a whiteboard. The plot shows the measured force in the $\hat{Z}_{\mathcal{E}}$ direction compared to the force setpoint in $\hat{Z}_{\mathcal{E}}$. The area shaded in red shows the time frame when the end-effector is in contact with the whiteboard, but the force controller is inactive. The area shaded in green shows how the force controller follows the desired setpoint of 10 $[N]$.

difference between when the force is not controlled (area shaded in red) and when it is controlled (area shaded in green). Table 3.2 shows the statistical information for the measured force in free flight, uncontrolled contact and force-position controlled contact. It signifies that during the controlled contact, the force error is similar to the sensor's natural noise (measured during free-flight), illustrating the effectiveness of the hybrid force-position controller in regulating the applied force as exactly as possible given the sensor's characteristics while following the desired position setpoints.

Table 3.2: Statistical comparison of the measured force (in $[N]$) during the example of Figure 3.37 writing the letter "A" on the whiteboard.

| | Mean | Std. Dev. | Min | Max |
|---|---|---|---|---|
| Free Flight | -0.03 | 0.68 | -2.35 | 2.31 |
| Uncontrolled Contact | 1.31 | 3.32 | -1.79 | 12.31 |
| Force-Controlled Contact | 5.77 | 0.86 | 3.91 | 7.90 |

During the hybrid force-position experiments, we noticed that if the given force setpoint is high, it severely affects the horizontal motion of the robot




**(a)** **(b)**

**Figure 3.37:** A hybrid force-position control experiment with the UAV writing the letter "A" on the whiteboard. (a) The plot shows the measured force in the $\hat{Z}_{\mathcal{E}}$ direction compared to the force setpoint given in $\hat{Z}_{\mathcal{E}}$. The area shaded in red shows the time frame when the end-effector is in contact with the whiteboard but the force controller is inactive. The area shaded in green shows how the force controller follows the desired setpoint of 6 $[N]$ while writing the letter. (b) A screenshot of the experiment right after finishing writing the letter.

during the contact. Figure 3.38 shows the scenario where the drawing of a 20 $[cm]$ line on the whiteboard is affected by the desired 10 $[N]$ applied force. The figure illustrates the jittery motion caused by the higher friction and the limited available horizontal thrust. Chapter 4 introduces methods to estimate the available thrusts, which would allow planning for such physical interaction tasks with the appropriate force setpoints.

The importance of controlling the force as opposed to the uncontrolled contact with lateral motion is emphasized in the sequences of Figure 3.39. We observed two behaviors: the excessive force applied during the uncontrolled contact may result in damaging the environment (in this scenario, toppling the whiteboard), and the uncontrolled contact results in repeated "banging" of the robot on the surface (see Figures 3.36 and 3.37(a)), which is undesirable in many physical interaction applications.

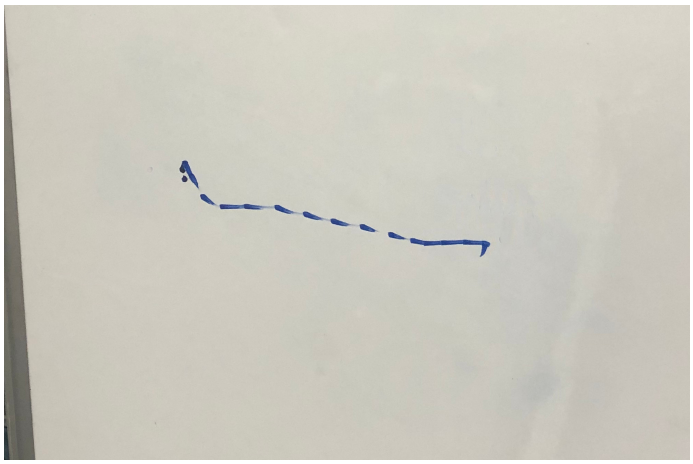
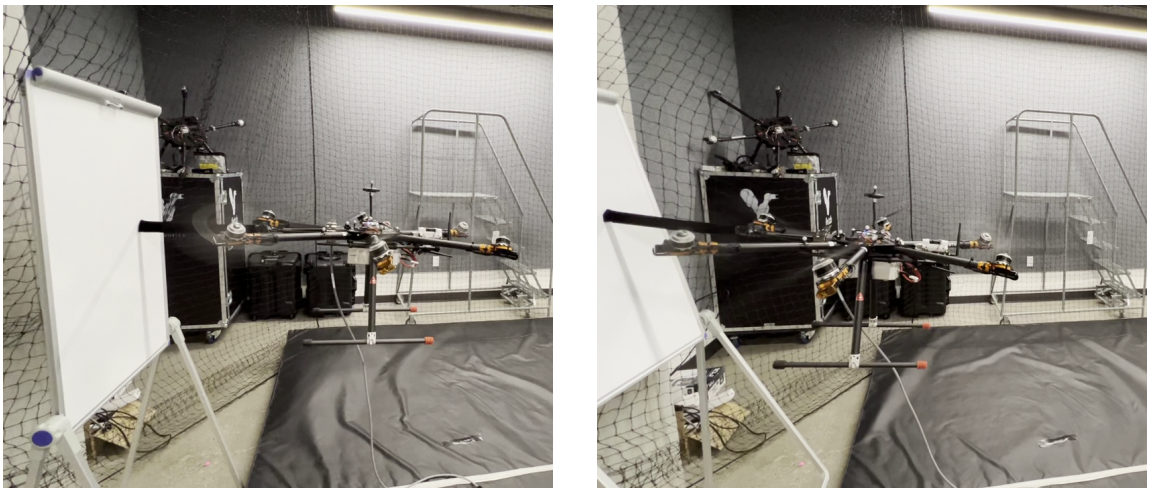


**Figure 3.38:** A hybrid force-motion control experiment to draw a 20 $[cm]$ horizontal line with a desired applied force of 10 $[N]$. The horizontal motion of the robot is affected by the high force setpoint, resulting in a jittery motion.



**(a)** **(b)**

**Figure 3.39:** An experiment with uncontrolled force to draw on the whiteboard. The uncontrolled force results in pushing the whiteboard until it falls.

## 3.8 Conclusion and Discussion

This section described the development of the fully-actuated controller for our fixed-pitch hexarotor robots of different sizes. We described how the existing flight controllers for underactuated multirotors can be extended to the fully-actuated vehicles to save time and effort and reduce application development

time and costs.

We explained our controller's architecture, which is implemented on top of the existing PX4 flight stack and can accept the inputs with either complete orientation or just yaw. We proposed a set of attitude strategies that compute the multirotor's entire attitude setpoint (desired orientation) when the full attitude is not given as the input to the system. The concept of attitude strategies allows the controller developed in Section 3.3 to interact with the same motion control and trajectory planner tools developed for underactuated UAVs while allowing the UAV to devise its full actuation capabilities. We proposed five such strategies and explained how the complete attitude setpoint could be computed based on the strategy.

Furthermore, we proposed using the lateral thrust limits of LBF robots in the thrust setpoint calculation. Two strategies were described with different objectives. The described strategies assumed a constant lateral thrust but can easily extend to use the lateral thrust limits calculated in real-time.

The calculation of the thrust setpoint ($\vec{F}_{sp}^{\mathcal{B}}$) is independent of the devised attitude strategy. However, many flight controller systems (i.e., PX4) use the calculated attitude setpoint frame $\mathcal{F}^{\mathcal{S}}$ instead of the current body-fixed attitude $\mathcal{F}^{\mathcal{B}}$ for the output thrust calculation. The choice can simplify the calculation and avoid some issues caused by delays in reading the current attitude and may not significantly affect underactuated vehicles, as the thrust setpoint is just a scalar and projecting it on the wrong frame only has a minor effect on the magnitude. However, this projection needs to be explicitly done on the current body-fixed frame for fully-actuated robots to avoid stability issues during large commands and sudden direction changes.

Multiple simulators were developed for the new controller's development and testing, which allowed us to experiment rapidly with different methods and ideas and facilitated the whole process.

We shared the lessons we learned from the real-world experiments that can enhance the fully-actuated vehicles' stability and performance. Changes, such as modified motor saturation strategies, are necessary for these vehicles to be

useful in practical applications.

We further expanded our proposed controller to simultaneously control the UAV's (or the end-effector's) position and the force applied to the contact surface. The developed hybrid position-force controller devises independent control loops for the position and the force in the contact frame. The result of the loops is combined based on the orthogonal subspaces controlled by each position and force. These subspaces are defined using two selection matrices that depend on both natural constraints (the geometry of the contact) and the artificial constraints (the task's requirements). The combined result of the position and force loops is then used as the total desired thrust (or acceleration) that the UAV needs to generate.

Finally, we showed the simulated and real experiments that illustrate how the proposed controllers work. We further performed indoor and outdoor flight experiments, tested the controller's disturbance rejection using the strategies by making uncontrolled physical contact with the environment, and illustrated the force-motion controller working for multiple tasks and different attitude strategies.

Chapter 4

# Wrench-Set Analysis for Fully-Actuated Multirotors

The last decade has seen increased research focus on the physical interaction of UAVs with their environment. Such interactions range from moving boxes using a single or a group of aerial robots to UAVs' precise or compliant application of force to their surroundings. Many proposed methods for such interactions have been successfully implemented in controlled settings. However, the existing approaches use conservative limits for applied forces and moments in their interactions, which is inefficient and does not use the full potential of the aerial manipulator. This conservatism may result in some tasks being completed non-optimally, and in the worst scenario, some feasible tasks can be deemed infeasible.

In the context of our multirotor, we propose a set of real-time methods to estimate the instantaneous wrenches (i.e., forces and moments) that a robot manipulator can generate. The methods estimate the wrenches based on the multirotor's design, state, and the desired forces applied during the physical interaction and output the force and moment (wrench) polytopes. The wrench-set estimation enables the use of the full potential of the existing wrenches for optimizing the accelerations in free flight and for physical interaction tasks.

Chapter 5 presents different possible applications of the real-time estimation method presented in this chapter.

In this chapter, we first introduce the problem and review the related work (Section 4.1). Then we provide our solutions for decoupled thrust and moment set estimations in real-time (Sections 4.2) and extend it to coupled lateral thrust estimation and coupled wrench set estimation(Section 4.4 and 4.3). Later, we review the possible extensions to variable-pitch and more complex robots (Section 4.5). Finally, we present our experiments and test results (Section 4.6).

## 4.1 Introduction and Related Work

The progress in research on the physical interaction of UAVs in the last decade has provided novel tools to enable new applications ranging from package delivery to contact inspection and aerial manipulation. Different research groups and companies have shown successful results in controlled and carefully-crafted settings. However, so far, these approaches have been ignoring the robot's capabilities by using conservative force and moment limits in their interactions. These limits result in sub-optimal actions and underuse the aerial manipulator's full potential. For some tasks, the conservative limits may make a feasible task infeasible. On the other hand, if the limits are set high, other additional precautions need to be taken to monitor and correct the controller's output; otherwise, the result can be a disastrous crash.

The effectiveness of a specific fully-actuated multirotor can be described by analyzing its *dynamic manipulability*. This concept has been used for ground robots since Yoshikawa introduced it in 1985 [206]. However, it has entered the UAV research only recently, in 2015, primarily for optimization and classification of the fully-actuated multirotors [89, 178, 179, 180]. *Dynamic manipulability ellipsoid* is the set of 6-D accelerations of the robot's CoM with unit-spherical input thrust force (rotor thrusts), represented as a six-dimensional ellipsoid. This ellipsoid illustrates how multirotor's thrusts and moments can be generated in different translation and rotation directions. Figure 4.1 shows an example of

the dynamic manipulability ellipsoid of the system developed in [78].

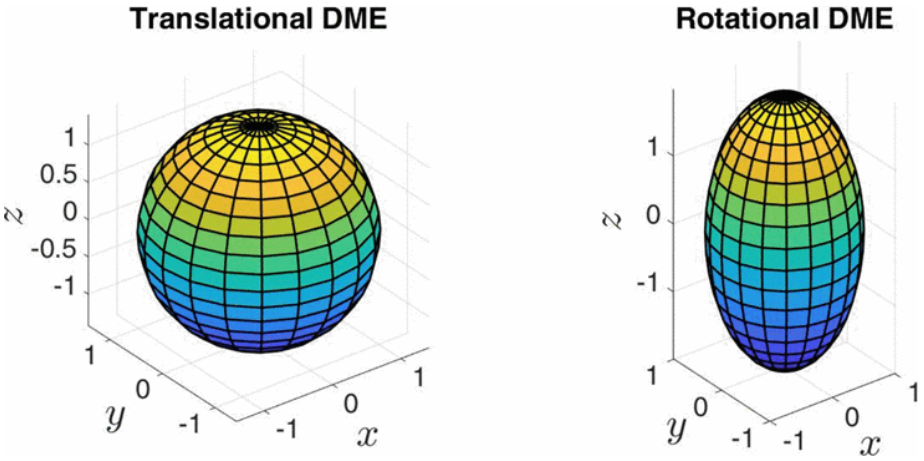



**Figure 4.1:** An example dynamic manipulability ellipsoid of the system developed in [78] (figure from [180]).

Dynamic manipulability ellipsoids have been shown to work fine in practice and are fast to compute but still fail to capture the entire space of available forces and moments. A more advanced descriptor, called *dynamic manipulability polytopes*, has been introduced by Chiacchio et al. [28, 30], which can represent the entire wrench space. We compute and use this latter descriptor to capture the whole space of available forces and moments in this work.

From a geometrical perspective, the thrust vectors generated by the UAV rotors can be seen as a set of force vectors in 3-D space with lower and upper limits for each vector's magnitude. In literature, an analogous example is a multi-fingered grasp of an object, where there is a set of force vectors in 3-D (at the contact points) with lower and upper limits defined by the forces each finger can apply to its contact point [153]. Therefore, the wrench polytopes for multirotors have geometrical similarities with the *grasp polytopes* used in grasp analysis, and many of the methods related to grasp polytopes can be adapted for use in the estimation of the wrench set for aerial robots.

In 1992, Avis and Fukuda [8] introduced a pivoting algorithm that has been widely used for offline computation of the wrench set. In 2020, we introduced

an online method for computation of the wrench set [79] in multirotors, which uses the geometrical properties and the convexity of the wrench set to compute the force and moment polytopes in real-time. Recently, our real-time thrust set estimation has been re-introduced by [173] in the context of general manipulation systems (including ground and humanoid robots). However, the authors have not expanded the method further to full wrench with desired (fixed) components.

The thrust and moment limits of fully-actuated multirotors have been generally treated as static values for each multirotor architecture. Franchi et al. [47] have proposed a fast optimization-based solution for lateral thrust estimation in real-time from the thrust set, but it only uses the static thrust set computed for the hovering. They estimate only the thrust set (ignoring the wrench set) using an analytical solution, which is only possible due to the symmetric nature of their tilted hexarotor and cannot be easily generalized to other architectures. On the other hand, in reality, the sets of feasible wrenches and the wrench limits depend not only on the architectural design but also on the current state and the instantaneous wrenches generated at each time. On the other hand, considering the wrench limits as static has resulted in thrust and moment limits being assumed decoupled, despite the strong dependency between them.

This chapter comprehensively discusses estimating the thrust and moment limits and proposes real-time methods to estimate the instantaneous wrench limits. We describe our real-time thrust set estimation method and expand it by taking into account the current state of the robot and the coupling between the thrusts and moments.

To continue with the chapter, we need to define the terminology used here:

*Thrust or Force Set* ($\mathcal{S}_\mathcal{T}$): The set of all the feasible 3-D thrust setpoints at the current instance of time. In other words, it is the set of all the thrusts that the multirotor will be able to generate almost instantaneously (i.e., in a short time period $\delta t$ from the current time $t$).

*Moment or Torque Set* ($\mathcal{S}_\mathcal{M}$): The set of all the feasible 3-D moment setpoints

at the current instance of time. In other words, it is the set of all the moments that the multirotor will be able to generate almost instantaneously (i.e., in a short time period $\delta t$ from the current time $t$).

*Wrench Set* ($\mathcal{S}_{\mathcal{W}}$): The set of all the feasible 6-D wrench (i.e., thrust and moment) setpoints at the current instance of time. In other words, it is the set of all the wrenches that the multirotor will be able to generate almost instantaneously (i.e., in a short time period $\delta t$ from the current time $t$).

## 4.2 Decoupled Thrust and Moment Set Estimation

This section discusses our method for constructing the Thrust Set $\mathcal{S}_{\mathcal{T}}$ and then explains how the Moment Set $\mathcal{S}_{\mathcal{M}}$ can be constructed similarly.

Let us assume $\mathcal{S}_{\mathcal{T}}(t)$ is the set of all feasible thrust vectors that our multirotor can generate at time $t$. We will omit the time dependency notation $(t)$ from now on when the context is clear. The set $\mathcal{S}_{\mathcal{T}}$ depends on the UAV's architectural design and hardware, the current state, and the environment. In this section, when we mention the current state, we also account for the external forces (such as wind and gravity).

In reality, due to the imperfect motors and motor controllers, delays in the system, and other uncertainties, calculating the exact set of feasible thrusts is impossible. Moreover, the actual thrust set would be an infinitesimally small volume around the current thrust due to the motor and system delays.

In practice, our main interest in $\mathcal{S}_{\mathcal{T}}$ is to gain the ability to modify the thrust setpoint so that it lies within the set. Therefore, the goal is calculating an estimation $\mathcal{S}'_{\mathcal{T}}$ to the union of all possible sets $\mathcal{S}_{\mathcal{T}}(t + \delta t)$ computed for a short time period $\delta t$ from time $t$. $\delta t$ is the shortest time that allows the thrust and moment setpoints to realize (i.e., $\delta t$ is assumed to be larger than system and motor delays but too small for any other state or environment variables to change meaningfully).

An approximation to the $\mathcal{S}_{\mathcal{T}}$ set can be mathematically estimated directly

from the UAV structure using numerical or exact methods. The resulting thrust set ignores the effects such as aerodynamic interference between the rotors, imperfections in the model, the current state of the system, and the current moment setpoint. Tadokoro et al. [178, 180] define Dynamic Manipulability Measure for UAVs to quantify the relationship between the structure of the UAVs and their feasible thrust and moment sets. These measures are beneficial for optimizing the UAV design but have limited use in estimating the feasible thrust set in practice.

For some special (mostly symmetric) structures, it is possible to formulate the $\mathcal{S}_{\mathcal{T}}$ set mathematically [47]. While these formulas can be helpful due to their simplicity, they completely ignore the vehicle's state, which affects the thrust set. In our experiments (see Section 4.6), we demonstrate how the thrust set changes with the change in the architecture and the state of the UAVs.

We propose a real-time method to numerically estimate the $\mathcal{S}_{\mathcal{T}}$ set for a general UAV system from the mathematical model of the system that can be used in real-time. The choice of the input $\mathbf{u}$ to the system (see Figure 3.2) is flexible, as long as each element $\mathbf{u}$ is the command for one of the motors, and each element is a monotonically increasing or decreasing function of the generated motor thrust. For example, $\mathbf{u}$ can be the vector of rotor thrusts, the vector of motor speeds (usually squared), or the vector of motor PWM signals. We also assume that the input elements' upper and lower bounds are known.

Algorithm 1 shows our proposed algorithm for real-time feasible thrust set calculation.

The proposed algorithm calculates the convex hull of all the thrusts resulting from the model's combinations of minimum and maximum rotor/motor inputs. The algorithm's expected time complexity is $O(2^{1.5r})$, where $r$ is the number of rotors. However, considering that the number of rotors in common UAV architectures is small, an efficient implementation of the algorithm can run on standard autopilot systems to approximate the feasible thrust set for the system's current state.

**Algorithm 1** Proposed approach for thrust set estimation

▷ This function estimates the thrust set for the UAV model and state
**function** ESTIMATETHRUSTSET(*model*, *state*)
  ▷ Update the UAV model state to the input state
  *model* ← SETSTATE(*model*, *state*)
  ▷ Get the number of rotors in the UAV
  $r$ ← GETNUMOFROTORS(*model*)
  ▷ Extract system's min and max input values as $r \times 1$ arrays, with each cell corresponding to a motor/rotor
  $(minU, maxU)$ ← GETINPUTRANGE(*model*)
  ▷ Define an empty set for the thrusts
  *thrusts* ← $\emptyset$
  ▷ Generate thrusts from all min/max combinations of motor commands
  **for** $i = 0$ **to** $2^r - 1$ **do**
    ▷ Convert the iterator $i$ into an $r \times 1$ binary array
    *iArray* ← NUMTOBINARYARRAY($i, r$)
    *iArrayNegated* ← NOT(*iArray*)
    ▷ Calculate the input array from the binary array
    ▷ $\odot$, $\oplus$: element-wise multiplication and addition
    $u$ ← $(maxU \odot iArray) \oplus (minU \odot iArrayNegated)$
    ▷ Calculate the thrusts generated by the model
    $t$ ← CALCULATETOTALTHRUST($model, u$)
    ▷ Add the calculated thrust to the thrust set
    *thrusts* ← ADDTOSET(*thrusts*, $t$)
  **end for**
  ▷ Calculate the convex hull of the thrusts
  *feasibleThrusts* ← CONVEXHULL(*thrusts*)
  ▷ Return the result
  **return** *feasibleThrusts*
**end function**

**Proposition 4.2.1.** *The feasible thrust set approximated by Algorithm 1 ($\mathcal{S}'_T$) is the complete thrust set that can be generated by the model and the robot state given to the algorithm as inputs.*

*Proof.* To restate the problem, the goal is to find the vector span of $r$ vectors (thrusts normal to the rotors), having the minimum and maximum magnitudes of the vectors.

The proposition claims that the vector span is the convex hull of all the vertices created by combinations of minimums and maximums of all the vectors. In other words, the sum of the vectors (each with a magnitude between their minimum and maximum) is within the convex hull, and each point within the convex hull can be written as a sum of the vectors with valid magnitudes.

Let us define the unit vector in the positive thrust direction of the $i^{\text{th}}$ rotor as $\vec{v}_i$ and the set of all $r$ unit vectors as $\mathcal{S}_\mathcal{V}$. Assuming $min_i$ as the minimum and $max_i$ as the maximum magnitude of the $i^{\text{th}}$ rotor's thrust, we can define the set of all the thrust intervals as $K$. The span of $\mathcal{S}_\mathcal{V}$ over $K$ (with its vectors bounded by their maximums and minimums) can be defined as:

$$\text{Span}_K(\mathcal{S}_\mathcal{V}) = \left\{ \sum_{i=1}^{r} \lambda_i \vec{v}_i \mid \vec{v}_i \in \mathcal{S}_\mathcal{V}, \ \lambda_i \in [min_i, max_i] \right\} \tag{4.1}$$

First, we define the set of points resulting from the sum of maximum and minimum values of the vectors as:

$$\mathcal{S}_\mathcal{P} = \left\{ \sum_{i=1}^{r} \lambda_i \vec{v}_i \mid \vec{v}_i \in \mathcal{S}_\mathcal{V}, \ \lambda_i \in \{min_i, max_i\} \right\} \tag{4.2}$$

The proposition claims that the span of $r$ vectors with their corresponding bounds is the convex hull of all the points generated by the combinations of those bounds. In other words:

$$\text{Span}_K(\mathcal{S}_\mathcal{V}) = \text{Conv}\left(\mathcal{S}_\mathcal{P}\right) \tag{4.3}$$

To prove the proposition, we use induction. For simplicity, we are dropping $K$ out of the notation.

***Base Case:*** With only one rotor ($r = 1$), the set $\mathcal{S}_{\mathcal{U}_i}$ of all the possible 3-D thrusts (represented as 3-D points) is a segment of the line passing the center in the direction of $\vec{v}_1$ spanning from $min_1$ to $max_i$. In this case, $\text{Span}(\mathcal{S}_\mathcal{V})$ is obviously the convex hull created only by $2^1$ points: the minimum and

maximum thrusts.

***Induction Step:*** Assuming the proposition holds for $r = n$, we want to show that it also holds for $r = n + 1$. In other words, if we add a vector $\vec{v}_{n+1}$ to our set of $n$ vectors, the resulting span is still a convex hull created only from the $2^{r+1}$ combinations of minimum and maximum thrust magnitudes.

Considering that both the $\mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots n}})$ and $\mathcal{S}_{\mathcal{U}}n + 1$ are convex, the new vector span is the Minkowski sum of the two sets, which is proven to be convex:

$$\mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots(n+1)}}) = \mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots n}}) +^{M} \mathcal{S}_{\mathcal{U}}n + 1 \tag{4.4}$$

where $+^{M}$ is the Minkowski sum of the two sets.

The resulting $\mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots(n+1)}})$ can be obtained by the sum of infinite sets resulting from shifting the $\mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots n}})$ by all points in $[min_{n+1}, max_{n+1}]$ interval in the direction of $\vec{v}_{n+1}$ vector. The result is equivalent to shifting $\mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots n}})$ in the direction of $\vec{v}_{n+1}$ vector by $min_{n+1}$, then shifting it by $max_{n+1}$ and taking the convex hull of the two sets. Note that the $\mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots n}})$ is already assumed to be the convex hull created from only the $2^r$ combinations of minimum and maximum thrust magnitudes.

Now, considering that $\mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots(n+1)}})$ is the convex hull created only by the vertices of $\mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots n}})$ shifted once by $min_{n+1}$ and once by $max_{n+1}$, the vertices of the convex hull can only be a subset of the vertices of $\mathrm{Span}(\mathcal{S}_{\mathcal{V}_{1\dots n}})$ shifted by $min_{n+1}$ or $max_{n+1}$ in $\vec{v}_{n+1}$ direction.

□

Noting that moments are also fundamentally vectors and have vector properties, a similar algorithm can be used to compute the moment set $\mathcal{S}_{\mathcal{M}}(t)$. The only difference would be using the computed moments instead of thrusts at each iteration (i.e., using $CalculateTotalMoment$ instead of $CalculateTotalThrust$) in Algorithm 1.

For multirotors with fixed rotor angles (i.e., fixed-pitch multirotors) and with a second-order model, the *shapes* of the feasible thrust and moment sets approximated by Algorithm 1 ($\mathcal{S}'_{\mathcal{T}}$) are independent of the current state and the external wrenches. However, the set's orientation depends on the current robot's attitude, and its location in the body-fixed frame $\mathcal{F}^{\mathcal{B}}$ translates with the external forces (e.g., gravity and wind). This observation for these architectures can be leveraged to speed up the computation significantly. A base set (e.g., $\mathcal{S}_{\mathcal{T}_b}$) can be computed only once in the body-fixed frame, ignoring the external forces. Then, to obtain the thrust set $\mathcal{S}'_{\mathcal{T}}$ for any external force and robot attitude, the base set $\mathcal{S}_{\mathcal{T}_b}$ can be rotated to the inertial frame and then shifted with the external forces. Figure 4.2 shows how the same thrust set rotates around when the UAV attitude changes.

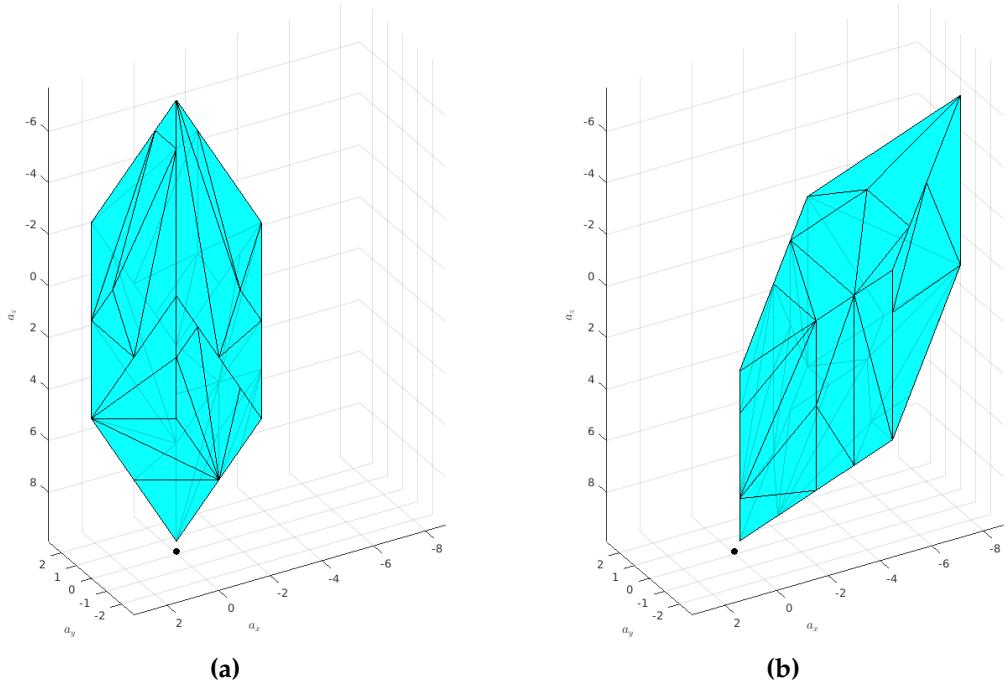



**Figure 4.2:** Thrust sets for architectures with fixed rotor angles rotate around the point of the total external force, shown as a black dot in the plots. The thrust sets are shown for our fixed-pitch hexarotor architecture at (a) zero roll, pitch, and yaw. (b) zero roll and yaw, but 30° pitch.

Our experiments (Section 4.6) show that the time complexity of this rotation

method is still exponential to the number of rotors (similar to Algorithm 1). However, utilizing it is still a few orders of magnitude faster than recalculating the entire wrench space at each iteration. Note that, even though this approach can significantly improve the execution speed, its use is only limited to the cases where all the following conditions are met: the robot is a fixed-pitch multirotor, the robot model is second-order, and the coupling of thrusts and moments is not important for the application.

The following section will present a solution to capturing the coupling between thrusts and moments when the goal is to estimate the lateral thrusts available to the UAV.

## 4.3 Coupled Lateral Thrust Estimation

The thrust set approximated by Algorithm 1 is more accurate compared to the fixed thrust set assumption of the current literature. However, it does not take into account the required moments when the thrust is generated. The actual thrust set with a nonzero moment is usually an even smaller set. In fully actuated multirotors, the moments should have priority over the thrusts (see Section 3.3). If we assume that the multirotor can achieve the moments without delay (a reasonable assumption for multirotors), the generated moments that affect the thrust set would be the same as the current moment setpoint.

If the goal is only to estimate the UAV's lateral thrust bound, a Monte Carlo method can be devised that calculates the lateral thrust set for a desired normal thrust ${}_zF_{sp}$. The method devises the available Control Allocation module and takes the current system state and the current moment setpoint into account. Similar to Algorithm 1, it requires knowledge about the input limits of the motors.

Algorithm 2 illustrates the proposed lateral thrust estimation method. Note that the algorithm requires an additional moment setpoint (or desired angular acceleration) input to the Thrust Setpoint Generator module, which is illustrated

in Figures 3.4 and 3.10 in dashed boxes.

**Algorithm 2** Proposed approach for lateral thrust limits estimation

▷ This function estimates the lateral thrust for input UAV model, current state, desired normal thrust and desired angular acceleration (alpha)
**function** EstimateLateralThrust(*ctrlAlloc*, *model*, *state*, *zd*, *alphad*)
  ▷ Initialize the control allocation
  *ctrlAlloc* ← Init(*ctrlAlloc*, *model*, *state*)
  ▷ Extract system's min and max input values as $r \times 1$ arrays, with each cell corresponding to a motor/rotor
  (*minU*, *maxU*) ← GetInputRange(*model*)
  ▷ Get the possible range for the UAV lateral thrust
  ▷ This can be guessed or calculated from Algorithm 1
  (*xRange*, *yRange*) ← GetLateralRange(*model*)
  ▷ Define an empty set for the thrusts
  *thrusts* ← ∅
  ▷ Generate and test random samples
  ▷ $K$ is the desired number of iterations
  **for** $i = 1$ **to** $K$ **do**
    ▷ Choose random numbers for $x$ and $y$ thrusts
    (*xd*, *yd*) ← Rand(*xRange*, *yRange*)
    ▷ Form the desired thrust vector
    *thrustd* ← MakeVector(*xd*, *yd*, *zd*)
    ▷ Check if the inputs are in the valid range
    *u* ← CalcInput(*ctrlAlloc*, *thrustd*, *alphad*))
    ▷ Form the desired thrust vector
    *thrustd* ← MakeVector(*xd*, *yd*, *zd*)
    ▷ Check if the inputs are in the valid range
    **if** IsInRange(*u*, *minU*, *maxU*) **then**
      ▷ Add the valid thrust to the thrust set
      *thrusts* ← AddToSet(*thrusts*, *thrustd*)
    **end if**
  **end for**
  ▷ Calculate the 2-D convex hull of the thrusts
  *latheralThrusts* ← ConvexHull(*thrusts*)
  ▷ Return the result
  **return** *lateralThrusts*
**end function**

The execution time depends on the speed of the Control Allocation's input calculation function (e.g., the mixer function) and the selected number of iterations $K$. The choice of $K$ is a trade-off between the execution time (keeping it real-time) and the precision required in estimating the lateral thrust.

The following section will show a more general solution to capturing the coupling between thrusts and moments.

## 4.4 Coupled Wrench Set Estimation

Algorithm 1 proposed in Section 4.2 can provide a real-time estimation of the thrust and moment sets (i.e., $\mathcal{S}_{\mathcal{T}}$ and $\mathcal{S}_{\mathcal{M}}$), which depends on the design of the multirotor and its current state. However, it does not capture the coupling between the thrusts and moments, which is especially necessary for physical interaction applications. This section describes our real-time wrench set estimation method that considers the desired forces and moments in its calculation.

Let us assume $\mathcal{S}_{\mathcal{W}}(t)$ is the set of all feasible 6-D wrenches that our multirotor can generate at time $t$. Similar to the previous section, we omit the time dependency notation $(t)$ from now on when the context is clear. The set $\mathcal{S}_{\mathcal{W}}$ not only depends on the UAV's design, the current state, and the environment but also on the desired moments and thrusts for the physical interaction with the environment. In this section, when we mention the current state, we consider the external forces (such as wind and gravity) also included in the current state.

In practice, our main interest in $\mathcal{S}_{\mathcal{W}}$ is to modify the thrust and moment setpoints, so they lie within the wrench set when some desired thrust and moment components are already fixed (e.g., when applying a 2 $[N]$ force to the wall). Therefore, the goal is to calculate an estimation $\mathcal{S}'_{\mathcal{W}}$ to the union of all possible sets $\mathcal{S}_{\mathcal{W}}(t+\delta t)$ computed for a short time period $\delta t$ from time $t$. The $\delta t$ is the shortest time that allows the thrust and moment setpoints to realize

(i.e., $\delta t$ is assumed to be larger than system and motor delays but too small for any other state or environment variables to change meaningfully).

The proposed real-time method for estimating the wrench set $\mathcal{S}_{\mathcal{W}}$ first calculates the 6-D full wrench sets similar to Algorithm 1, ignoring the knowledge about the desired moment and thrust components. Algorithm 3 shows the steps to calculate the full 6-D wrench set $\mathcal{S}_{\mathcal{W}_{6D}}$.

**Algorithm 3** Proposed approach for 6-D wrench set estimation

▷ This function estimates the 6-D wrench set for the input UAV model and the current state
**function** ESTIMATE6DWRENCHSET(*model*, *state*)
  ▷ Update the UAV model state to the input state
  *model* ← SETSTATE(*model*, *state*)
  ▷ Get the number of rotors in the UAV
  $r$ ← GETNUMOFROTORS(*model*)
  ▷ Extract system's min and max input values as $r \times 1$ arrays, with each cell corresponding to a motor/rotor
  $(minU, maxU)$ ← GETINPUTRANGE(*model*)
  ▷ Define an empty set for the 6-D wrenches
  *wrenches* ← $\emptyset$
  ▷ Generate wrenches from all min/max combinations of motor commands
  **for** $i = 0$ **to** $2^r - 1$ **do**
    ▷ Convert the iterator $i$ into an $r \times 1$ binary array
    *iArray* ← NUMTOBINARYARRAY($i, r$)
    *iArrayNegated* ← NOT(*iArray*)
    ▷ Calculate the input array from the binary array
    ▷ $\odot$, $\oplus$: element-wise multiplication and addition
    $u$ ← $(maxU \odot iArray) \oplus (minU \odot iArrayNegated)$
    ▷ Calculate the wrench generated by the model
    $w$ ← CALCULATETOTAL6DWRENCH($model, u$)
    ▷ Add the calculated wrench to the wrench set
    *wrenches* ← ADDTOSET(*wrenches*, $w$)
  **end for**
  ▷ Calculate the convex hull of the wrenches
  *feasibleWrenches* ← CONVEXHULL(*wrenches*)
  ▷ Return the result
  **return** *feasibleWrenches*
**end function**

Now, assume that the 6-D wrench set $\mathcal{S}_{\mathcal{W}_{6D}}$ is computed, and we would like to know the feasible wrenches when we are applying (or desiring to apply) a specific force or moment to the environment during the physical interaction. The result is all the forces and moments inside $\mathcal{S}_{\mathcal{W}_{6D}}$ that lie on the hyperplane defined by the fixed force or moment component. In other words, to compute the set of feasible wrenches when one of the 6-D dimensions is fixed, we can simply intersect the hyperplane of the fixed dimension with the convex hull of the wrench set $\mathcal{S}_{\mathcal{W}_{6D}}$. The intersection represents the feasible set of wrenches (i.e., forces and moments) where the given wrench dimension is fixed to the desired value. Note that the intersection of a hyperplane with a convex shape is a convex shape with a smaller dimension. Therefore, the resulting wrench set after the intersection is also convex.

To take it further, for each desired dimension of forces and moments, we can iteratively intersect the wrench set and get the new wrench set with the desired constraints. Algorithm 4 shows the steps for estimating the wrench set $\mathcal{S}_{\mathcal{W}}$ when one or more components of the forces and moments are fixed to the desired value.

**Algorithm 4** Wrench set estimation with desired (fixed) components

```
▷ This function estimates the wrench set for the input UAV model, the current state and the desired wrench components
function EstimateWrenchSet(model, state, desiredWrenches)
    ▷ Estimate the full 6-D wrenches first
    feasibleWrenches ← Estimate6DWrenchSet(model, state)
    ▷ Iterate through the fixed (desired) wrench dimensions
    for i = 1 to Size(desiredWrenches) do
        ▷ Get the hyperplane for the fixed dimension
        hyperplane ← ConstructHyperplane(desiredWrenches[i])
        ▷ Intersect the hyperplane with the convex region of wrenches
        feasibleWrenches ← Intersect(feasibleWrenches, hyperplane)
    end for
    ▷ Return the result
    return feasibleWrenches
end function
```

The final result of the proposed approach is a convex (can be empty) set of all the feasible wrenches with the desired components fixed. Note that if $n$ fixed dimensions are desired, the final convex set $\mathcal{S}_\mathcal{W}$ can have a dimension of at most $6 - n$, and an empty set means that the UAV cannot achieve all the desired forces and moments at the same time.

Section 4.6 shows the results of our implementation of the algorithm and illustrates how different desired wrench components can affect the wrench set.

The following section briefly describes how this method can be further extended to multirotors with variable-pitch rotors.

## 4.5 Additional Extensions of the Method

The proposed Algorithm 4 presented in Section 4.4 estimates the instantaneous wrench set $\mathcal{S}_\mathcal{W}$ for multirotors with fixed-pitch rotors. For multirotors with variable-pitch rotors, the same algorithm can estimate the instantaneous wrench set $SetW$ given the following two conditions:

1. The multirotor state also includes the current pitch of the rotors (i.e., the angle of the servos controlling the rotor pitch),
2. For the purposes of the wrench set estimation, the change rates for the rotor pitch angles are assumed to be low, allowing us to ignore the rotor pitch changes for the short future time $\delta t$, which is used for wrench set definition (see Section 4.2).

However, there are applications where estimating the set of all possible wrenches is desired, assuming that the rotor pitches can take any angle within their limits. For example, when a single or more motors fail, knowing the whole wrench space for any rotor pitch angle is desired to recover from the failure. Remember that each rotor can generate wrenches in a single direction for fixed-pitch rotors and with the magnitude between its minimum and maximum wrench. In the variable-pitch rotor scenario, each servo motor connected to the rotor can create a sweep over the space, resulting in the space of possible

wrenches looking like a sector of a circle between the angles of the servo rotor and filling the radius between the minimum and maximum wrench of the rotor itself. Figure 4.3 illustrates the range of possible wrenches for a single rotor that is connected to a servo pitching from $\theta_1$ to $\theta_2$.

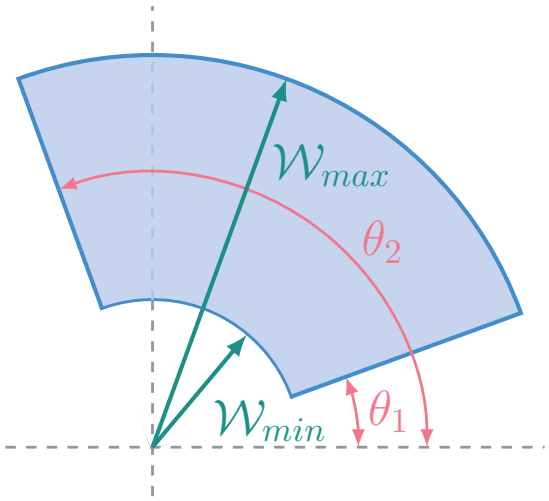



**Figure 4.3:** The space of possible generated wrenches for a single rotor connected to a servo pitching with the angle limit between $\theta_1$ to $\theta_2$.

With such a non-linear space of possible wrenches for each rotor, quickly combining two or more spaces becomes intractable and difficult to compute analytically. However, it is possible to estimate the wrench space by sampling different pitch angles. Section 4.6 illustrates this idea and Chapter 5 shows some of the applications of these analysis.

## 4.6 Experiments and Results

All the methods proposed in this section have been implemented and tested. Figures 4.4 and 4.5 illustrate how Algorithm 1 can estimate the thrust and moment sets of different multirotor architectures. All the thrust sets are estimated for zero attitude. The cross-sections calculated using Algorithm 2

show how the lateral thrust changes with the change in the desired $Z$ thrust.

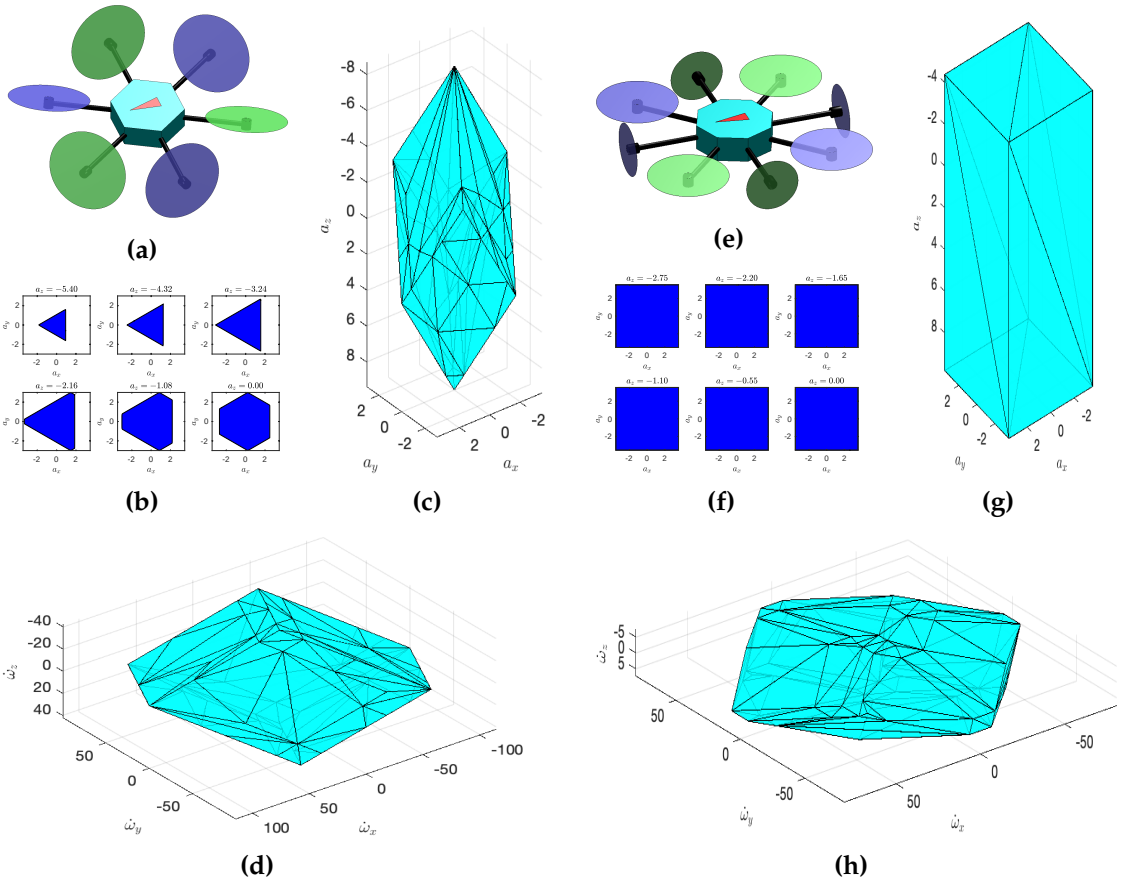



**Figure 4.4:** Thrust and moment sets have different shapes depending on the architecture of the fully-actuated UAV. (a) A hexarotor with rotors tilted sideways along with the (d) moment set, (b, c) thrust set, and its cross-sections along the $\hat{Z}_{\mathcal{I}}$ axis. The larger $Z$-thrust reduces the available lateral thrust. (e) An octorotor with four co-planar upward rotors and four auxiliary motors perpendicular to the main rotors along with the (h) moment set, (f, g) thrust set and its cross-sections along the $\hat{Z}_{\mathcal{I}}$ axis. The lateral thrust in this architecture is entirely independent of the normal thrust.

Figure 4.4 demonstrates how the thrust and moment sets can significantly differ for different multirotor architectures. On the other hand Figure 4.5 highlights how different UAV states can also result in very different thrust and moment sets.

The shapes of the thrust and moment sets computed using Algorithm 1

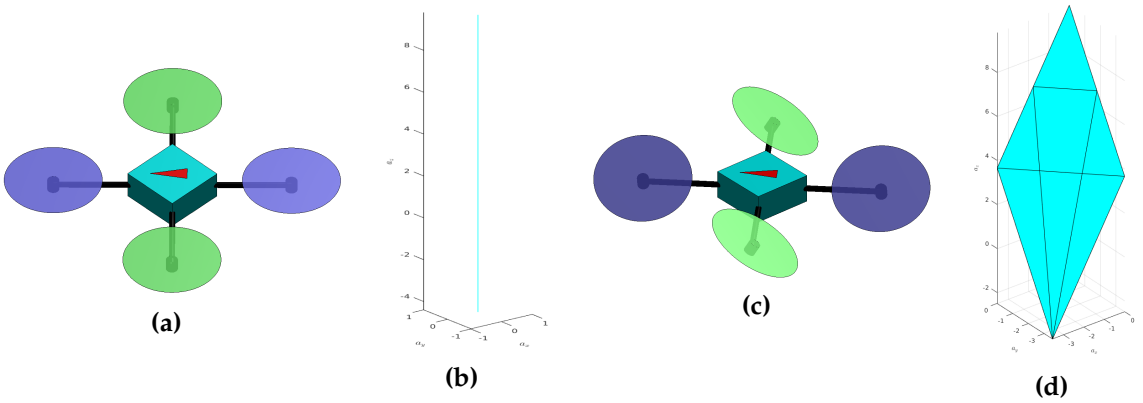



**Figure 4.5:** Thrust and moment sets have different shapes depending on the state of the UAV. A quadrotor with variable-pitch thrusters is shown along with its thrust set at different states: (a, c) When all the rotors are upward and parallel, the thrust set is a line. (b, d) Front and back rotors are tilted to the left, and side rotors are tilted to the back. The thrust set is a planar subspace.

depend on the architecture as well as the robot's state. However, as discussed in Section 4.2, for the specific class of multirotors with fixed-pitch rotors, the shapes of the decoupled thrust and moment sets do not change, i.e., only rotating with the body-fixed frame in the inertial frame $\mathcal{F}^{\mathcal{I}}$ and shifting with the external forces. Figure 4.6 shows thrust sets computed for different orientations of two robot architectures with fixed-pitch rotors.

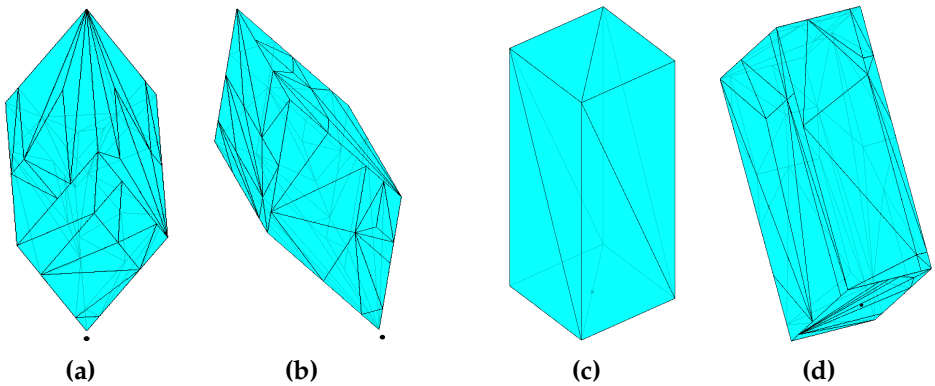



**Figure 4.6:** In fixed-pitch multirotors, the thrust sets have a fixed shape and are fixed in the body-fixed frame $\mathcal{F}^{\mathcal{B}}$. Illustrated are thrust sets at different robot orientations for (a, b) the hexarotor in Figure 4.4(a), and (c, d) the octorotor in Figure 4.4(e). The center of rotation is shown with a dot at the bottom of each shape.

Figure 4.7 shows the lateral thrust sets calculated using Algorithm 2 for the hexarotor with tilted rotors (in Figure 4.4(a)) with different desired normal thrusts and angular accelerations. All the thrust sets are calculated for the same architecture and state, but the resulting limits are very different.

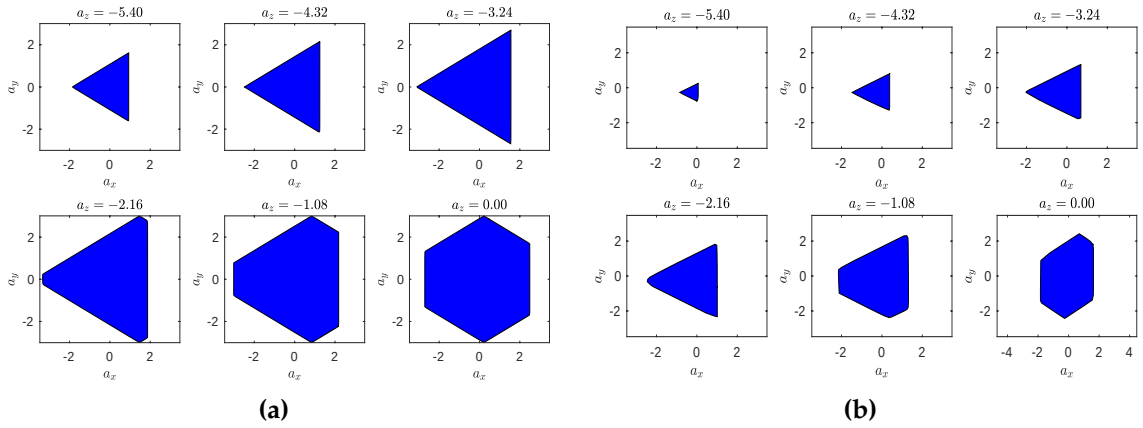
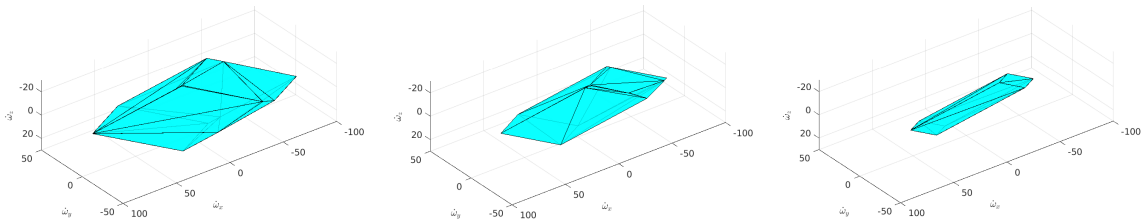



**Figure 4.7:** Thrust set depends on the desired angular accelerations (moments) as well as the UAV architecture and state. The lateral thrust set calculated for different normal thrusts by Algorithm 2 for the hexarotor of Figure 4.4(a) at the same state. Each figure (a) and (b) shows the thrust sets calculated with the same desired angular acceleration but different desired normal thrusts. The two figures are plotted for the same set of desired normal thrusts and only differ in their desired angular accelerations.

Figure 4.8 illustrates how the wrench set can be estimated using Algorithm 4 when some of the wrench components already have assigned (desired) values. This example shows how the decoupled moment set of Figure 4.4(d) shrinks in Figure 4.8 when the desired thrusts are generated during the physical interaction.

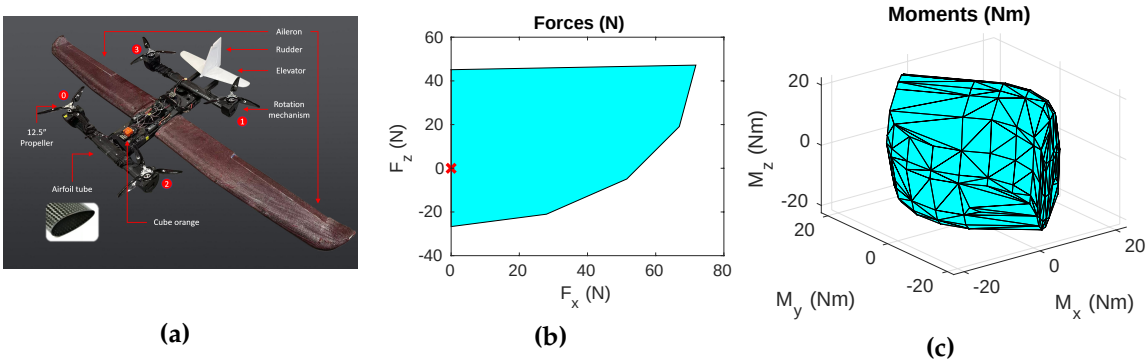
Figure 4.9 shows the wrench sets for a Vertical Take-Off and Landing (VTOL) aerial robot estimated for different actuator failures from our work in [124, 125]. The wrench sets are calculated by sampling different rotor pitch angles as discussed in Section 4.5. For each sample of pitch angles, Algorithm 4 is utilized to estimate the wrench set, and all the points of the wrench sets are used at the end to estimate the new convex hull for the complete wrench set.

We measured the execution speeds for different real-time wrench set estima-



(a) (b) (c)

**Figure 4.8:** Moment sets have different shapes and limits depending on fixed (i.e., desired) thrusts. A hexarotor with tilted arms applying forces to the environment. Here, the forces on axes $\hat{X}_{\mathcal{B}}$ and $\hat{Z}_{\mathcal{B}}$ are the same, but the forces on $\hat{Y}_{\mathcal{B}}$ are (a) 0, (b) 1 and (c) 2 Newtons.




(a) (b) (c)

**Figure 4.9:** The wrench set of a VTOL with four variable-pitch rotors estimated using the sampling method discussed in Section 4.5 and Algorithm 4. (a) VTOL UAV. (b) Thrust set. (c) Moment set. Figure taken from our work in [124, 125].

tion methods introduced in this chapter. Table 4.1 presents the execution times and frequencies for our hexarotor with tilted arms (see Figure 4.4(a)) and the fully-actuated octorotor shown in Figure 4.4(e). The following scenarios were measured in the table:

- Decoupled thrust and moment set estimation using Algorithm 1.
- Decoupled thrust and moment set estimation by rotation of the initial sets pre-computed using Algorithm 1.
- Lateral thrust estimation using Algorithm 2 with 500 query points.
- Coupled wrench set estimation, by first using Algorithm 3 to estimate the decoupled 6-D wrench, then using Algorithm 4 to estimate the coupled wrench set with three fixed wrench components.

- Coupled wrench set estimation, by first rotating the initial 6-D wrench (pre-computed using Algorithm 3), then using Algorithm 4 to estimate the coupled wrench set with three fixed wrench components.

All tests were performed in our MATLAB simulator on a system with Intel® Core™ i9-10885H CPU and 64 GB DDR4 RAM.

Table 4.1: Execution times and frequencies for different wrench set estimation methods for the hexarotor with tilted arms in Figure 4.4(a) and the octorotor in Figure 4.4(e), measured in our MATLAB simulator.

| | Hexarotor | | Octorotor | |
|---|---|---|---|---|
| | Time (ms) | Hz | Time (ms) | Hz |
| Decoupled (Algorithm 1) | 22.90 | 43.67 | 60.35 | 16.57 |
| Decoupled (Rotation) | 0.11 | 8813 | 0.45 | 2209 |
| Lateral (Algorithm 2) | 25.09 | 39.86 | 21.17 | 47.25 |
| Coupled (Algorithms 3 + 4) | 51.99 | 19.23 | 153.13 | 6.53 |
| Coupled (Rotation + Algorithm 4) | 34.09 | 29.34 | 106.60 | 9.38 |

The results from Table 4.1 emphasize that the time complexity of the wrench set estimation algorithms is exponential on the number of rotors, regardless of being coupled or decoupled, and with or without leveraging the rotation (in fixed-pitch robots). Another observation is that rotation of the pre-computed initial force and moment sets can significantly speed up the thrust and moment set estimation, even though rotation also has exponential time complexity. The effect of rotating the pre-computed wrench space in coupled wrench set estimation is still positive; however, the resulting execution time is in the same order of magnitude as re-computing the 6-D wrench set at each iteration using Algorithm 3. Note that the rotation method can only be used for a limited class of fixed-pitch multirotors. However, given the provided speed improvements, it should be utilized whenever all of the rotors in the UAV have fixed pitches.

Finally, the execution times for the lateral thrust estimation of Algorithm 2 show that, unlike the other methods, the time complexity of the lateral thrust

estimation does not have an exponential dependency on the number of rotors, which is an advantage over the other methods when only the coupled lateral thrust is required for a multirotor with many rotors.

## 4.7 Conclusion and Discussion

This chapter proposed several real-time algorithms to calculate the decoupled and coupled moment and thrust sets for the aerial robot in the current state and with the desired applied wrenches. The algorithms can be used for many purposes ranging from planning and control allocation to failure recovery and design optimization to improve the physical interaction of UAVs with their environment. Some of the applications of the methods are presented and discussed in Chapter 5.

First, we introduced a real-time method to independently estimate the forces (thrusts) and moments. This geometry-based method creates the convex set of forces and moments by considering the lower and upper limits of each rotor's possible thrusts and torques.

Then a random sampling-based real-time method was described to obtain the lateral forces considering the desired moments. This method directly computes the lateral thrust without computing the entire thrust and moment sets, which allows it to run without the exponential time complexity of the full set computation.

Then we extended the force and moment set estimation into a real-time wrench set computation method that captures the coupling between the forces and moments and can provide a much more accurate estimation of the wrenches during physical interaction. We further explored how these methods can be extended to more complex robotic configurations such as aerial robots with variable-pitch rotors (i.e., thrusters).

The experiments on different multirotor architectures and conditions illustrated how the methods described here work in practice.

The following chapter presents some of the applications of the methods proposed in this chapter.

Chapter 5

# Wrench-Set Applications for Fully-Actuated Multirotors

Dynamic manipulability analysis and the wrench set estimation for multirotors have opened the door to many possible improvements for different applications. Offline decoupled analysis was introduced around seven years ago and has since been devised to optimize UAV designs and improve the controller. With our real-time coupled wrench set estimation method proposed in Chapter 4, not only the existing applications can improve further, but also many new applications become possible.

This chapter briefly describes some example applications for our proposed real-time wrench set estimation method. However, we believe that this powerful tool enables many more enhancements from hardware to different parts of the flight software to improve the flight quality and boost the abilities of UAVs to have physical interaction with their environment.

Section 5.1 reviews the wrench-set applications presented in this chapter. Section 5.2 illustrates how the wrench set estimation method can be utilized to improve control allocation for fully-actuated multirotors. Section 5.3 shows how the performance can be optimized when an external force is applied to the UAV. Finally, Section 5.4 explains the ways that wrench set estimation can

optimize planning for physical interaction tasks.

## 5.1 Introduction

The introduction of dynamic manipulability analysis in the robotics community enabled improvements in many applications, optimizing the physical interaction tasks for ground manipulators by providing information about their instantaneous and total forces and moments. As discussed in Chapter 4, real-time applications have been constrained to using the limited dynamic manipulability ellipsoids due to the high computational cost of dynamic manipulability polytopes, which has forced their use mainly to offline scenarios. Our work in Chapter 4 has provided a novel solution for the computation of dynamic manipulability polytopes (i.e., wrench set) that can be used in many manipulator robots in real-time applications. However, due to the scope of our work in this thesis, this chapter illustrates example applications of this powerful tool for aerial robots.

Offline decoupled computation of the force and moment sets for aerial robots was introduced five years ago and has since been devised to optimize UAV designs and improve the controller.

Control allocation for multirotors is a vital module that computes the actuator commands (i.e., rotor speeds) based on the desired forces and moments computed by the controller. A simple control allocation system is just a function that uses the inverted dynamics of the system to compute the actuator commands. However, the output of such a system is not protected from actuator commands being computed outside the physical range of the actuators. Therefore, when the desired forces and moments given to the control allocation module are outside the feasible range, the physical saturation of the commands by the actuator may make the UAV unstable or degrade its performance. The physical interaction tasks are even more critical, and the UAV needs to be very precise at controlling its poses and wrenches. Section 5.2 explains how our real-time wrench set estimation method can improve the control allocation

module in multirotors.

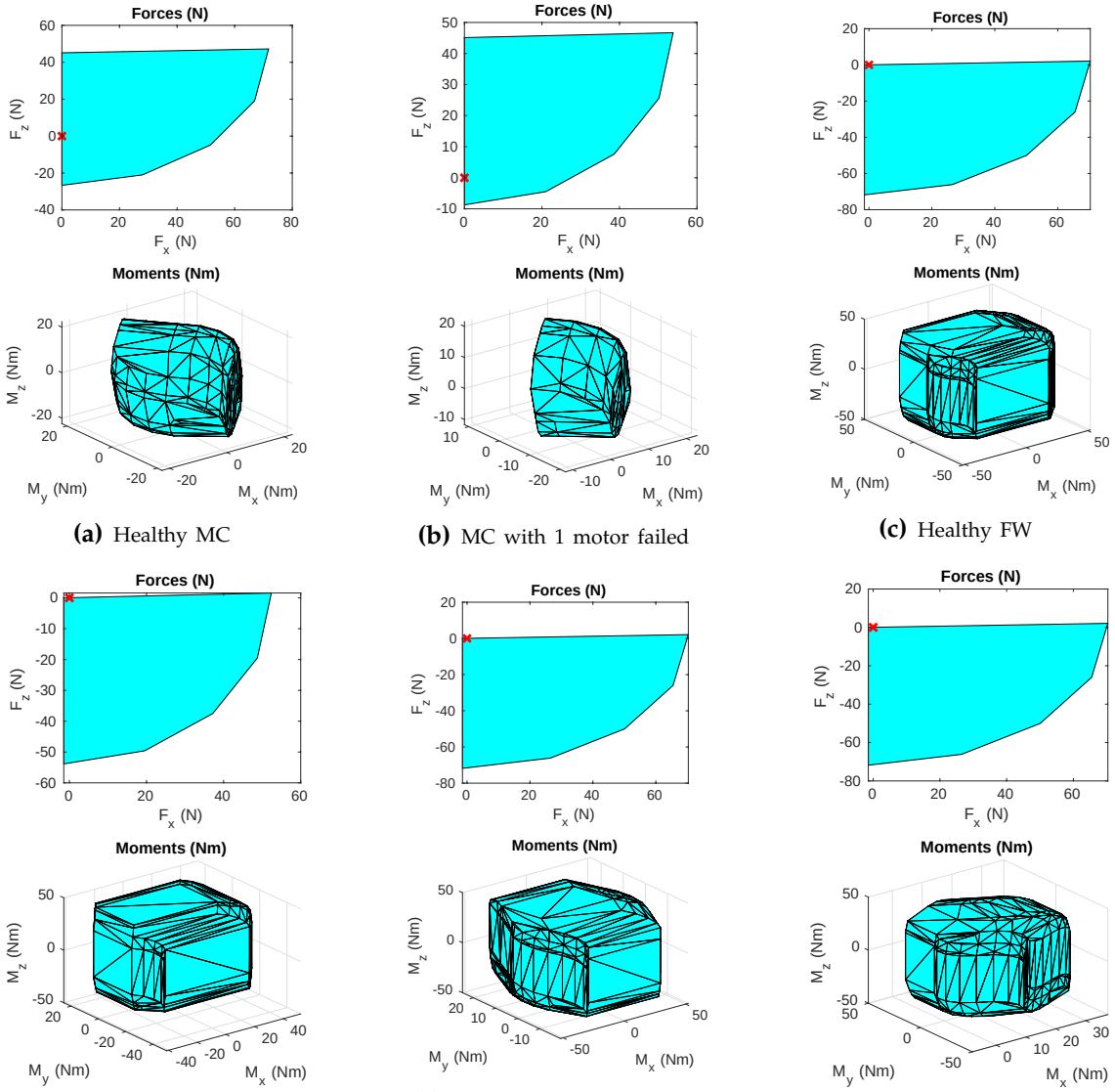
The ambient wind or physical contact with the environment can affect the UAV's performance by exerting forces on the robot and changing the aerodynamic properties. When LBF fully-actuated vehicles try to keep the desired attitude, they may lose a significant portion of their lateral thrust to oppose these forces. In extreme cases, they may consume all their lateral thrust and drift, losing their tracking ability. Section 5.3 describes our solution to estimate the optimal tilt, which reduces the consumption of the lateral thrust when constant forces are applied to the UAV.

Planning controlled physical interaction tasks for multirotors requires accounting for the required wrenches to perform the task in addition to the desired motions. With conservative assumptions for the available wrench limits at each step of the planning, the planner will not utilize the full potential of the robot, and the resulting plan may be suboptimal. Knowing the possible wrenches at each state allows optimizing the time, energy, or other parameters of the physical interaction task. In extreme examples, the conservative wrench limit assumptions may result in the planner failing to find a viable plan for finishing the task that could be otherwise performed with the knowledge about the complete wrench set. Section 5.4 expands on this idea and describes the benefits of the real-time wrench set in planning in more detail.

Although we showcase only the mentioned applications of the wrench set, there are many other ways that the real-time wrench set estimation can benefit aerial robots.

For example, it has been used for multirotor design optimization and finding the optimal design parameters for the desired objective [89, 140, 148, 179].

Mochida et al. [119, 120] have used the idea of thrust-set estimation for optimization of the multirotor design to make it robust to failure, as well as for hoverability and failure analysis. We have also recently used it in our work on VTOL failure recovery to understand how the wrench set changes when a failure happens [124, 125]. Figure 5.1 shows the computed thrust and moment sets when different actuator failures happen.




**(a)** Healthy MC **(b)** MC with 1 motor failed **(c)** Healthy FW

**(d)** FW with 1 motor failed **(e)** FW with 1 elevator failed **(f)** FW with 1 aileron failed

**Figure 5.1:** The wrench set of a VTOL with four variable-pitch rotors estimated for different actuator failure conditions using the sampling method discussed in Section 4.5 and Algorithm 4. Figure taken from our work in [124, 125].

## 5.2 Improving Control Allocation Performance

In Chapters 2 and 3 we have shown the conventional control design for a fully-actuated controller (see Figure 3.2). This design has a Control Allocation

module tasked with converting the desired forces and moments (or linear and angular accelerations) computed by the previous parts of the controller and converting them to the required rotor inputs $\mathbf{u}$ that can result in the desired forces and moments (or accelerations). Figure 5.2 shows this module with its inputs and outputs.

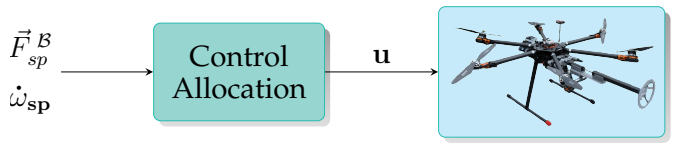



**Figure 5.2:** A high-level illustration of a typical control allocation module with its inputs and outputs.

Section 2.7 derived the most common control allocation method for fully-actuated multirotors which uses *exact feedback linearization and decoupling method* (a.k.a. *nonlinear dynamic inversion*) [23, 34, 112, 145]. The major drawback of this method is that the resulting output (i.e., the elements of $\mathbf{u}$) may exceed the minimum or maximum physical limits of what the rotors can achieve. If not properly treated, the over- or under-saturated rotors can result in a different total wrench than desired, causing instability and, in the worst case resulting in a crash. The problem exacerbates when the demands for the generated forces and moments are higher and more precise control over wrenches is required, such as during the robot's interaction with the environment.

Researchers have proposed three approaches to alleviate the issue:

- Limiting the computed output $\mathbf{u}$ to within the limits of the motor inputs after $\mathbf{u}$ is computed by the control allocation module.
- Utilizing an optimization-based allocation method to keep the computed motor commands within the desired limits.
- Limiting the desired forces and moments to ensure that the computed output $\mathbf{u}$ is within the physical limits of the multirotor motors and what rotors can generate.

A possible solution with the first approach (limiting $\mathbf{u}$ after the computation) is to saturate the computed values in order to send feasible commands to the motors. However, this approach has similar results to not doing anything for the actual UAV because sending an out-of-bounds command to motors will naturally result in saturation regardless. Another solution is to change the computed commands prioritizing the stability of the robot, usually by trying to reduce the rotor commands in a way that the moments are kept intact, and the thrusts are reduced [114]. However, this approach is strongly architecture-dependent and hard to generalize.

In the second approach, i.e., devising an optimization-based allocation method, the motor limits are considered as constraints, while the dynamic inversion equation is considered a soft constraint in the cost function [43, 123]. In addition to the computational cost of the nonlinear optimization, the approach makes it very difficult to tune the resulting output wrenches when the input desired wrench is infeasible. Therefore, these methods are not suitable for physical interaction applications where keeping the direction of the wrenches and prioritizing the moments over the thrusts to keep the orientation may be desirable.

For the third approach, i.e., limiting wrenches before the computation of outputs in the control allocation module, most existing solutions only consider the thrust limits, and all of them consider the set of possible thrusts and moments as static [42, 61, 112]. The primary issue with considering static bounds for wrenches is that the full potential of the robot is not utilized if the limits are set too small. On the other hand, even if the smallest limits are chosen based on the desired operation of the robot, any unpredicted situation causing an even smaller limit for the wrenches may cause instability due to the control allocation exceeding the limits and saturating the motors.

One of the most advanced methods in this class is proposed by Franchi et al. [47], which limits the lateral thrusts by estimating the static thrust set at hovering, ignoring the orthogonal thrust and all moment limits, and ignoring the effect of the UAV state and the desired wrenches on the thrust set. Therefore,

their solution works well for near-hovering conditions but does not result in correct limits for other scenarios.

A recent improvement is proposed by Bezzera and Santos [15], which uses a method similar to our proposed decoupled algorithm for thrust set estimation (Algorithm 1). Their work is limited to thrusts and ignores the effects of the UAV orientation on the wrenches. However, it demonstrates the improvements that can be made using wrench set estimation.

Using the coupled wrench set estimation method of Section 4.4 (i.e., Algorithm 4), the control allocation can be improved further. Figure 5.3 illustrates our proposed control allocation method.

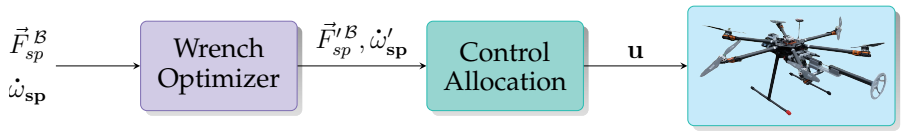



**Figure 5.3:** A high-level illustration of the control allocation module with its inputs and outputs.

The Wrench Optimizer works as follows:

1. Estimates the wrench set using Algorithm 3,
2. Checks if the input thrust and moment setpoints are feasible (i.e., fall within the wrench set):
   (a) If feasible, it outputs the setpoints without a change to the conventional control allocation module.
   (b) Otherwise, bring the thrust and moment setpoints to within the wrench set using the priorities set by the task and the developer.

In this method, the developer has flexibility over what to prioritize when the thrust or moment setpoints are outside the feasible wrench set. Depending on the task at hand, the priority might be keeping the attitude and altitude or the force applied to the environment. In either of these cases, due to the geometric nature of the problem, the final solution can be computed by finding

the intersection of a 6-D wrench vector with the boundary of the 6-D convex set, which simplifies into the intersection of a line with a hyperplane and checking if the intersection point is in the polygonal face of the set.

## 5.3 Flight Optimization in the Presence of Constant Force

External forces acting on the multirotor, such as forces from the physical interaction or the ambient wind, can affect the UAV's performance by exerting forces on the robot and changing the aerodynamic properties. Fully-actuated vehicles may lose a portion of their lateral thrust to oppose the external force when trying to keep the desired attitude. In LBF vehicles, the consumed lateral thrust can be a large portion of their maximum possible lateral thrust. In extreme cases, the external force may overpower the UAV, in which case it starts drifting (e.g., with the wind) and can lose its tracking ability in the direction of the external force.

Many hardware and software solutions for wind estimation have been introduced to mitigate the mentioned effects (see [1, 60, 167]). Hardware devices, such as flow sensors (pitot tubes) and ultrasonic anemometers, are generally more accurate than the software methods. However, they add to the total payload weight, increase hardware complexity, and reduce the UAV's balance and stability due to their placement requirements. Moreover, their use cannot be extended to other sources of constant force applied to the robot, such as physical contact. On the other hand, purely software approaches, such as the tilt angle and rotor speeds methods, are simple to implement and work with other sources of applied force but have lower accuracy and require extensive calibration and specific working conditions.

Our goal, however, is to increase the performance of the LBF vehicles in the presence of the constant force (e.g., ambient wind), and directly measuring these forces may be unnecessary for this purpose.

We define performance as the ability of the UAV to accelerate in any direction. Let us call the maximum acceleration that the UAV can achieve in all directions in its current state as *omni-directional acceleration* $a_o$. Maximizing the performance of the UAV in the wind would mean maximizing the omni-directional acceleration.

The variables controllable by the UAV controller that can affect the solution are UAV's attitude and generated thrust. Therefore, we can define the problem as finding the optimal force and attitude setpoints that maximize the omni-directional acceleration:

$$\begin{aligned}(\overrightarrow{F}_{sp}, \mathbf{\Phi_{sp}}) &= \underset{(\overrightarrow{F},\ \mathbf{\Phi})}{\operatorname{argmax}}\, a_o \\ \text{subject to } \overrightarrow{F} &\in \mathcal{S}_{\mathcal{T}}\end{aligned} \tag{5.1}$$

In our solution, we make some assumptions:

1. All UAV rotors have fixed angles and positions in the body-fixed frame.
2. When hovering with no external forces other than the gravity acting on the UAV, the optimal tilt is zero (i.e., the $\hat{Z}_{\mathcal{B}}$ and $\hat{Z}_{\mathcal{I}}$ axes should align). In a more general case, when the assumption is not valid (e.g., when the horizon is not leveled), we can consider the optimal (non-zero) tilt for hovering as the baseline and add the offset after the optimal tilt is calculated with respect to this baseline.
3. If the force in the $ZI$ direction increases (e.g., we add the robot's weight), the optimal tilt remains zero. The assumption is valid for many UAV architectures (including the fixed-pitch hexarotor in our experiments) as long as the UAV remains an LBF robot (when the remaining normal thrust is much higher than the available lateral thrust).

When the aerial robot is interacting with the physical world or is flying in the wind, a force $\overrightarrow{F}_{app}$ is impacting its flight. Combined with the gravity force $\overrightarrow{F}_{grav}$ (i.e., the weight), this force applies a total external force to the UAV.

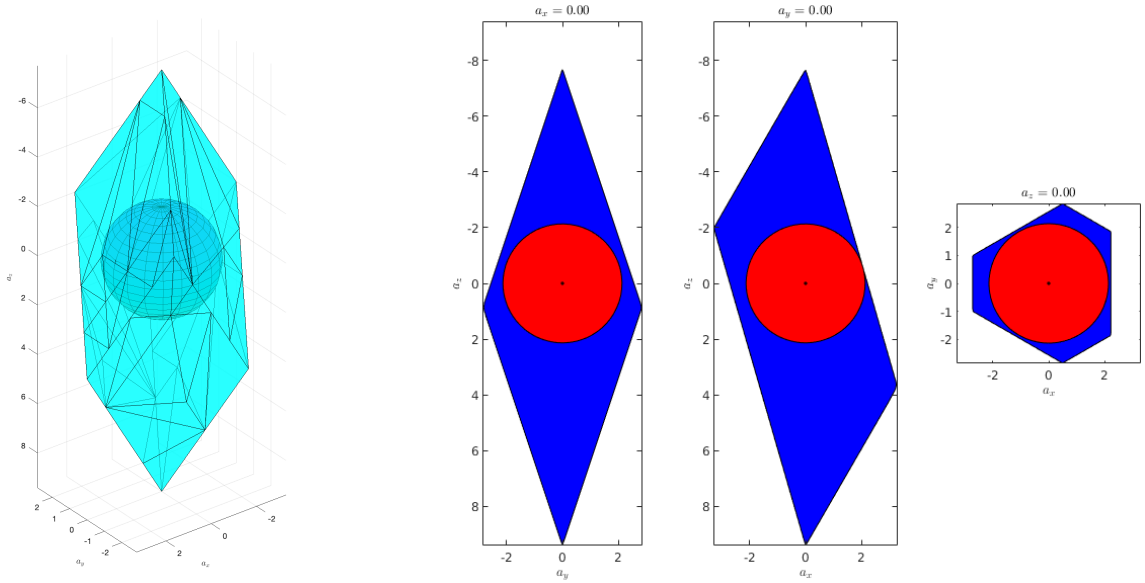



**Figure 5.4:** The illustration of the largest sphere centered around $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$ inscribed in the acceleration set of Figure 4.4(c). The sphere's radius is the largest acceleration that the UAV can generate in any desired direction at the current state called *omni-directional acceleration*. (a) The full acceleration set. (b) The cross-sections passing through the center along the $\hat{X}_\mathcal{I}$, $\hat{Y}_\mathcal{I}$, $\hat{Z}_\mathcal{I}$ axes.

Ignoring minor aerodynamic effects, the total external force $\overrightarrow{F}_{ext}$ is:

$$\overrightarrow{F}_{ext} = \overrightarrow{F}_{app} + \overrightarrow{F}_{grav} \tag{5.2}$$

Let us imagine the maximum inscribed sphere inside the current acceleration set (the set of all the possible accelerations at the current state) centered around our desired acceleration. Figure 5.4 illustrates this sphere on the acceleration set of our fixed-pitch hexarotor.

Without the loss of generality, we assume that the desired acceleration is zero, meaning that the UAV either intends to hover or fly at a constant speed. Thus, the center of the inscribed sphere should be $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$. Extending the method to non-zero acceleration is straightforward and will not be discussed.

An LBF multirotor has a much smaller lateral acceleration compared to the

acceleration normal to its body. Therefore, the radius of the inscribed sphere will ultimately be limited due to the lateral acceleration limit. This radius is the maximum acceleration that the robot can achieve in any desired direction at its current state and is the same as the omni-directional acceleration $a_o$ defined above.

To extend the current analysis to devise the thrust set estimation methods described in Chapter 4, note that the shape of the acceleration set is just a linear scaling of the thrust set. However, to achieve zero acceleration (to stay in equilibrium), the UAV needs to generate the total thrust with the same magnitude but in the opposite direction of the total external force (i.e., $-\vec{F}_{ext}$). Therefore, to find the omni-directional *thrust* $F_o$, we need to find the maximum inscribed sphere inside the thrust set centered around $-\vec{F}_{ext}$.

**Proposition 5.3.1.** *With the assumptions of Section 5.1, the full-tilt attitude strategy of Section 3.4.2, combined with an optimal method for producing the thrust setpoint, will converge to the optimal tilt and thrust setpoints that maximize the omni-directional acceleration in the presence of the external force* $\vec{F}_{ext}$.

*Proof.* If the devised position control method is optimal (the case for almost all popular methods), the desired thrust $\vec{F}_{des}$ calculated by the position controller will eventually converge to $-\vec{F}_{ext}$ if the external force is constant. Therefore, considering that the gravity changes are negligible, $\vec{F}_{des}$ will converge to $-\vec{F}_{ext}$ when the airspeed is constant.

From Section 4.2, we know that for the robots with the fixed rotor angles, the tilt of the robot will only rotate the thrust set around the external force point without any change to the shape. On the other hand, the third assumption of the problem (see Section 5.1) means that the robot will have the largest omni-directional acceleration when the desired acceleration point falls on the $\hat{Z}_{\mathcal{B}}$ line. This also means that the attitude resulting in zero consumed lateral thrust is optimal.

The zero consumed lateral thrust of the UAV happens when the total

generated thrust is normal to the body. Thus, during the equilibrium (when $\vec{F}_{des} = -\vec{F}_{ext}$), the full-tilt attitude strategy (Section 3.4.2) results in the optimal attitude setpoint. □

In reality, even when the UAV is hovering, there are changes to the external force (e.g., due to the air pressure changes and gusts). Therefore, the optimal tilt of the UAV will continuously be changing. However, in practice, assuming that the average wind or external force is constant, the optimal tilt can be averaged over some time and then locked using the fixed-tilt strategy (Section 3.4.4). This way, the UAV will keep the tilt required to oppose the external force (e.g., wind) effectively and will have the largest thrust left to accelerate in any direction and to reject the unpredicted disturbances.

We performed tests on several architectures to experimentally check the validity of the optimal tilt obtained from the full-tilt attitude strategy. Figure 5.5 illustrates such test performed on the architecture of our fixed-pitch hexarotors (see Figure 4.4(a)) that shows the relationship between the tilt and the omni-directional acceleration.

Another interesting observation is that when the UAV structure is symmetric, the motor inputs tend to converge to the same value at the optimal tilt when the accelerations are zero. Figure 5.6 illustrates how the motor inputs converge once the tilt estimation starts for our fixed-pitch hexarotor of Figure 3.18(a).

## 5.4 Planning Physical Interaction Tasks

The planner system planning for a task involving physical interaction of the robot with its environment requires to respect not only the environment constraints (such as obstacles) and robot kinematic and dynamic constraints (such as velocity limits and maximum turn rates) but also the constraints imposed due to the physical interaction. Such limits include the contact constraints as described in Section 3.6, as well as the limits on the possible wrenches than can be applied during that interaction.

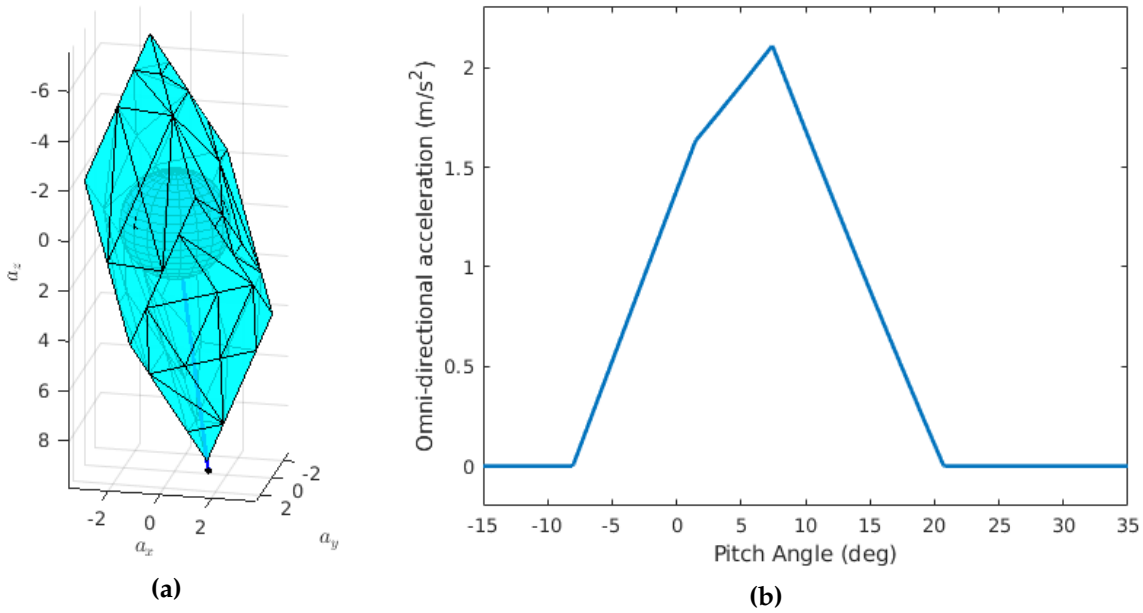



**Figure 5.5:** The relation of the omni-directional acceleration with the tilt in the presence of external forces. A 10 $[N]$ force is applied from south to north by the wind to the fixed-pitch hexarotor used in our project. (a) The optimal tilt is when the center of the omni-directional acceleration sphere (i.e., the desired acceleration) is placed on the UAV's $\hat{Z}_{\mathcal{B}}$ axis. (b) The omni-directional acceleration for different pitch angles.

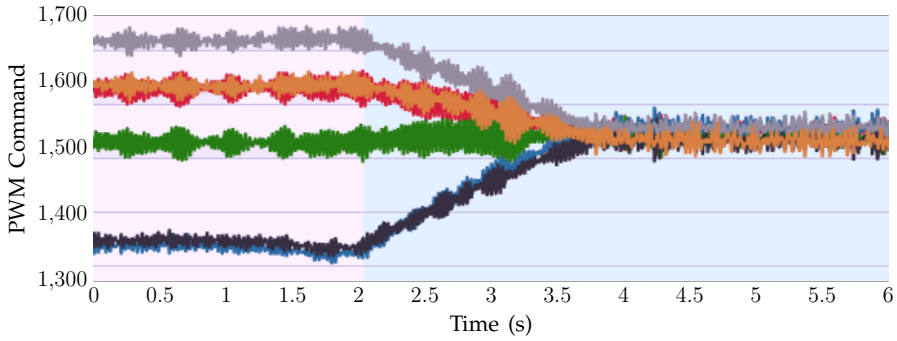
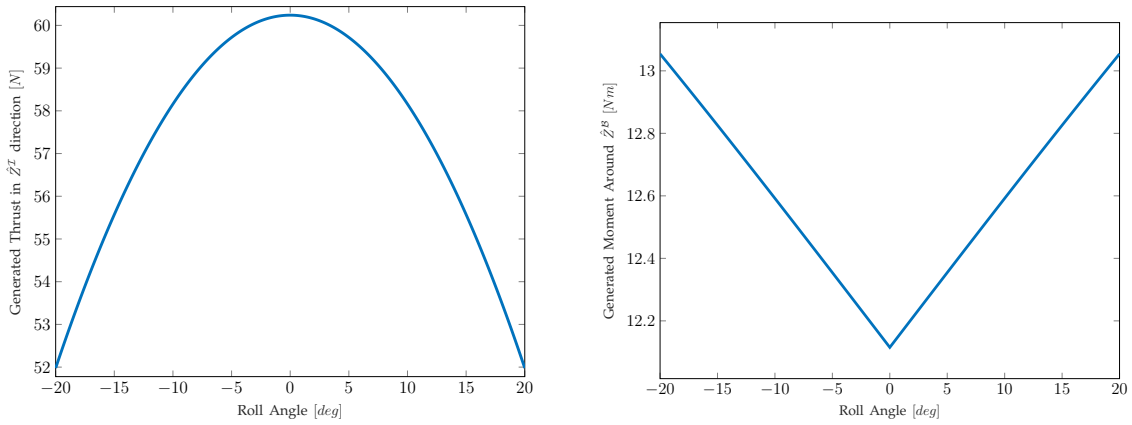



**Figure 5.6:** The motor inputs for the fixed-pitch symmetric hexarotor of Figure 3.18(a) converge over time when the symmetric hexarotor is in optimal tilt to oppose the wind force. The plot shows the PWM commands for the motors vs. time during hovering in an almost constant wind. The pink background shows the flight with the zero-tilt attitude strategy, while the blue background shows when the wind estimation optimizer runs during the full-tilt attitude strategy.

The study of the feasibility of applying desired motions and wrenches for a specific task is called *feasibility analysis* and was first introduced by Chiu




**(a)** **(b)**

**Figure 5.7:** The maximum generated vertical thrust and the moment around the $\hat{Z}_{\mathcal{B}}$ axis at different roll angles for the fixed-pitch hexarotor used in this project (see Figure 3.18(a)).

in 1987 [31, 32]. Feasibility and manipulability analysis have been used for ground manipulators, humanoid robots, and space manipulators in conjunction with task planning to provide feasible plans for physical interaction and manipulation of their environment and to optimize the manipulability of the plans [29, 54, 191, 202].

In addition to pure feasibility check on the tasks and instantaneously applied wrenches, manipulability analysis can optimize the robot's approach and pose for the specific task to achieve the best manipulation. Jaquier et al. [62] provides a planning system to optimize the robot's pose for the given task. Figure 5.7 shows the thrust and moments generated at different roll angles of our hexarotor platform during contact. The plots show that the optimal outputs do not always align, and depending on the task, different trajectories should be planned to achieve the optimal goal.

So far, planning methods have been using dynamic manipulability ellipsoids for online applications. These ellipsoids are computationally fast to compute, but they omit a significant portion of the wrench set, preventing the use of all the available wrenches.

On the other hand, the computation method for estimating the complete

wrench set (dynamic manipulability polytopes) has been computationally expensive, limiting its use only to offline task and trajectory planners [62].

Our real-time wrench set methods proposed in Chapter 4 allow the use of the entire wrench space in online planning methods. This extension can allow tasks that have been deemed infeasible with the limits given by dynamic manipulability ellipsoids and can further optimize the physical interaction tasks. In our previous work [6] we have shown an example of how our method can be used with the RRT*-Connect planner to plan a flight and manipulation task using our hexarotor with tilted arms.

## 5.5 Conclusion and Discussion

This chapter discussed several aerial robotics applications where our proposed real-time wrench set estimation method is beneficial. The method's benefits can be seen in all kinds of free flight and contact scenarios, but its impact on controlled physical interaction and manipulation is much more significant.

We first described how the control allocation module of aerial robots can utilize the real-time wrench set estimation methods to optimize the robot's behavior. Using this method ensures that the input of the control allocation is within the feasible limits and thus eliminates the need for optimization methods to refine the output of the dynamic inversion method (i.e., computed actuator commands). On the other hand, when the desired wrenches computed by the controller are infeasible, using the wrench set provides a flexible geometric method to decide the robot's behavior as opposed to the optimization-based post-processing of the output, which is difficult to formulate and implement.

Next, we used the wrench set estimation methods to prove that the full-tilt attitude strategy of our controller provides the maximum omni-directional acceleration when an external force is acting on the robot. Then we described how the optimal tilt and thrust could be computed in this scenario. The method is helpful for LBF robots where the available lateral force is limited, and the

external forces can consume all the lateral force resulting in the loss of trajectory tracking performance.

We discussed how planning for physical interaction tasks can use the estimated wrench set to find feasible ways to perform the desired tasks and compute optimal plans that utilize the complete wrench set. The information on the possible wrenches in each step allows the planner to explore the entire configuration space and plan for motions and wrenches in real-time that were not possible before due to conservative assumptions for the wrench set.

Finally, we briefly mentioned some other applications of the methods for aerial robots, such as optimizing the multirotor design and recovery from failures during the flight.

Having the powerful tool of real-time wrench set estimation in hand, many other applications and improvements are possible. Due to the scope of our work, we only covered example applications for aerial robots. However, the method is general and can also be extended to other manipulator robots to optimize their physical interaction.

The following chapters focus on deformable objects and explore how aerial robots can perform physical interaction tasks on these objects.

Chapter 6

# Deformable One-Dimensional Object Detection

Aerial robots are increasingly being used for automated inspection and maintenance of infrastructure. While recent advances have primarily addressed the UAV interaction with rigid external bodies, the work on automated interaction and manipulation of deformable objects such as wires and cables has been minimal. In order to enable applications such as autonomous inspection, maintenance, and interaction of the infrastructure containing such deformable objects (e.g., utility poles, network interface devices, cable boxes), an effective method is required to detect these objects in a suitable manner for robotic manipulation.

Various methods were developed across industrial and surgical robotics to segment the deformable objects. On the other hand, several methods have been proposed and implemented across academia and industry to allow the detection of power lines and bridge cables using UAVs for visual inspection or obstacle avoidance purposes. While the output of all these methods can be used for remote user operation, visual inspection, and obstacle avoidance, they are not suitable for manipulating these objects.

On the other hand, many methods exist to model and track deformable one-dimensional objects (e.g., cables, ropes, and threads) across a stream of

video frames. However, these methods depend on the existence of some initial conditions, and to the best of our knowledge, the topic of detection methods that can extract those initial conditions in non-trivial situations has hardly been addressed. The lack of detection methods limits the use of the tracking methods in real-world applications and is a bottleneck for fully autonomous applications that work with these objects.

This chapter proposes our approach for detecting deformable one-dimensional objects, which can be used for tasks such as routing and manipulation and automatically provides the initialization required by the tracking methods, effectively filling the gap between the segmentation methods and the tracking methods. It takes an image containing a deformable object and outputs a chain of fixed-length cylindrical segments connected with passive spherical joints. The chain follows the natural behavior of the deformable object and fills the gaps and occlusions in the original image. Our tests and experiments show that the method can correctly detect deformable one-dimensional objects in various complex conditions and can handle crossings and occlusions.

The proposed method moves us closer to enabling automated UAV inspection, maintenance, and manipulation of infrastructure containing cables and wires in the future. It was also a missing step for the complete automation of robotics applications involving planning and manipulation of such deformable objects in other fields as well, including industrial and surgical robotics settings.

## 6.1 Introduction and Related Work

Manipulating non-rigid objects using robots has long been the subject of research in various contexts [5]. A specific class of non-rigid objects is called Deformable One-dimensional Objects (DOOs) or Deformable Linear Objects (DLOs) and includes objects such as ropes, cables, threads, sutures, and wires.

In order to achieve full autonomy in physical interaction tasks using UAVs

involving DOOs (e.g., inspection and maintenance of utility poles), one of the essential and challenging parts is perception. Many such applications require complete knowledge of the object's initial conditions, even in the occluded parts. Moreover, routing and manipulation applications also require a representation model of the DOO that allows simulation and computation of its dynamics.

There are many methods proposed in the medical imaging community that can find the DOOs (known as tubular structures in that context) in the image frame [56, 94, 134, 196, 197]. These methods are considered segmentation methods and are perfected for low signal-to-noise images, and some can even work with self-crossings. However, they mainly only provide the region in the image containing the DOO, and their output is a collection of data points (e.g., pixels or point clouds). Such methods can be adapted for our robotics settings, such as physical UAV inspection. However, their output needs further processing to be useful for automated robotics tasks such as manipulation or routing.

The UAV community has also extensively worked on the segmentation of the cables and wires for visual inspection and obstacle avoidance purposes. These methods can effectively segment out the power lines, bridge cables, and other near-straight deformable objects [36, 92, 110, 139, 184, 210, 212, 214]. However, none of these approaches can handle true deformations that other cables and wires may have, such as turns and crossings, making these approaches unsuitable for other types of deformable one-dimensional objects used in utility poles and other robotics tasks.

On the other hand, various algorithms have been proposed for industrial settings to track a DOO across the video frames, even in the presence of occlusions and self-crossings. These methods may devise different tools such as registration methods (e.g., Coherent Point Drift [126]), learning, dynamics models, and simulation to predict and correct the prediction across the consecutive frames [63, 95, 138, 142, 152, 168, 182, 199]. While some of these methods can initialize the DOO in the first frame using trivial conditions (e.g., a straight rope in camera view), the other methods require even a simple DOO configuration

to be provided to them a priori. Most tracking methods will fail to initialize in even slightly complex initial conditions.

The initial conditions are not generally provided in real-world applications, and pure segmentation is insufficient and requires further processing. For example, our application of an aerial manipulator working with wires at the top of a utility pole needs to detect the desired wire, including its occluded parts and crossings and requires a model of the detected wire to perform any task on it.

The authors of [203] proposed a method that trains on rendered images and fine-tunes on real images to detect and track the state of a rope. The method requires tens of thousands of rendered and real images of the same rope for training and needs re-training and tuning for a new DOO. This approach may be acceptable for an industrial application focusing on DOOs of the exact same characteristics, but it may not be practical for many other applications, including ours.

We propose a method to detect the initial conditions of a deformable one-dimensional object [87, 88]. The method fills the occluded parts and works with crossings. The output is a single object represented as a discretized model useful for routing, manipulation, and simulation. The detected object can be used for initialization of other existing tracking methods to be used in applications such as manipulating DOOs into desired shapes and knots [102, 162, 183, 194] and is a step toward enabling aerial manipulation of wires and cables for inspection and maintenance tasks.

## 6.2 Problem Definition

This work addresses the detection of cable-like deformable shapes. Unlike rigid objects, the shape of deformable objects can change, and some form of a flexible model is required to represent the current state of their shape. On the other hand, to predict the reaction of the deformable objects to the applied forces

and moments, the representation model should facilitate the integration of a dynamics model.

In theory, a DOO represented by its pixels (or voxels) in the camera frame can be integrated with a dynamics model. However, the model would typically require finite element analysis and is computationally expensive, making it impractical for robotics applications. Simpler representations are commonly used in tasks such as manipulation, routing, and planning. Arriola-Rios et al. [5] provide an overview of the common representations for deformable objects.

A commonly-used approach for representing DOOs is to model them as a chain of fixed-length cylindrical segments connected by spherical joints. This simple model can easily integrate with an efficient dynamics model to simulate or predict the object's behavior. While the chosen model does not affect the ideas described in the proposed method, our method utilizes this model. In our application, the length of each segment is represented by $l_s$, and there is no gap between the segments (i.e., each segment starts precisely where the neighbor segment ends). Figure 6.1 illustrates the fixed-length cylinder chain model used in this work.

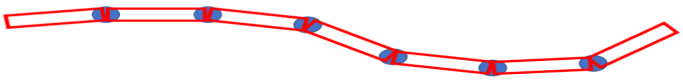


**Figure 6.1:** The representation of a DOO as a chain of fixed-length cylinders connected by spherical joints.

The focus of this work is to provide the cylinder chain representation of a deformable one-dimensional object seen in the camera frame (which can be RGB or RGB-D/3-D). The output chain model should predict and fill the path taken by the DOO under the occlusions and return a single chain object.

## 6.3 Proposed Method

The DOO detection method in this work takes the camera frame and performs a sequence of processing steps to output a single object in chain representation (described in Section 6.3). Figure 6.2 shows a high-level overview of the method.

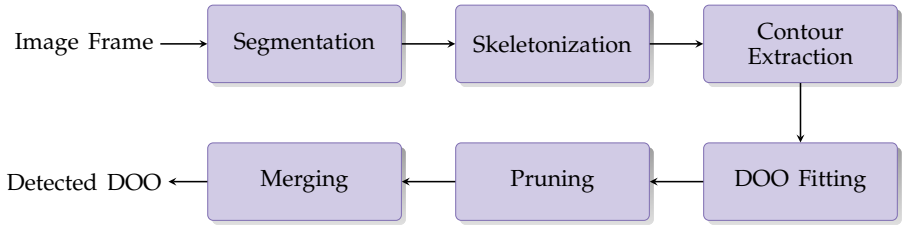



**Figure 6.2:** The high-level overview of the proposed method for detection of deformable one-dimensional objects.

The first three steps in the proposed method are well-known processes. Our algorithm can work with different segmentation, skeletonization, and contour extraction approaches. The choice depends on the task at hand. The last three processing steps are the contributions of our algorithm.

Algorithm 5 shows the pseudo-code of our approach. This section describes each of those steps in more detail.

### 6.3.1 Segmentation

A vital step in extracting the complete DOO from the input camera image is segmentation. This step aims to filter the image data to extract the DOO portions and exclude all other data. More formally, defining $P_{doo}$ as the collection of all image data points (pixels in 2-D case) belonging to the DOO, the segmentation output should ideally be the collection $P_{seg}$ where:

$$P_{seg} \subseteq P_{doo}, \quad \frac{|P_{seg}|}{|P_{doo}|} \approx 1, \tag{6.1}$$

**Algorithm 5** Deformable one-dimensional object detection.

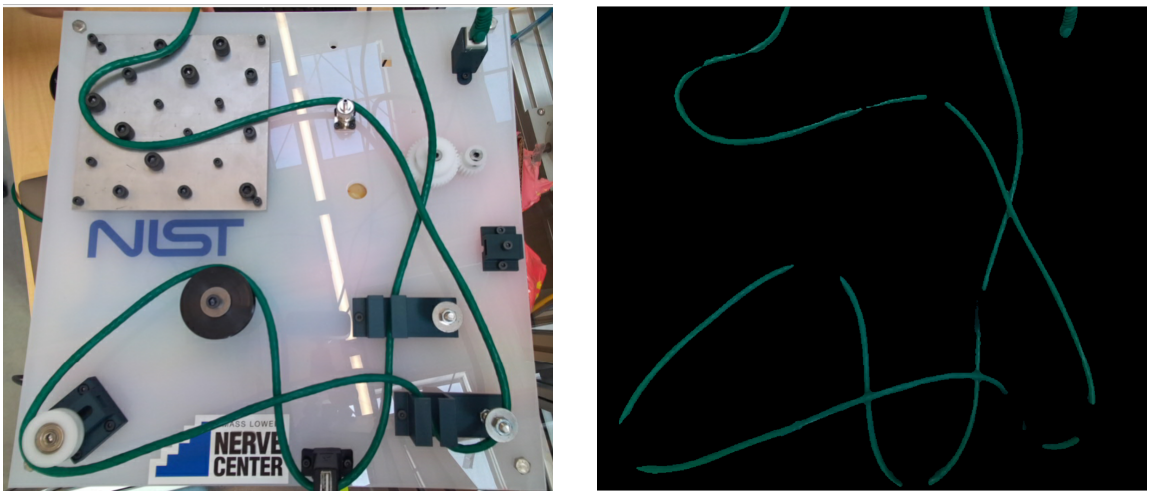
```
▷ Detects and extract DOOs from the input image frame.
function DETECTDOO(frame)
    ▷ Segment the image to extract the DOO region
    segmented_img ← SEGMENT(frame)
    ▷ Extract the skeletons of the segmented regions
    thinned_img ← SKELETONIZE(segmented_img)
    ▷ Extract the contours from the skeletons
    contours ← EXTRACTCONTOURS(thinned_img)
    ▷ Fit DOO chains to all contours
    Chains ← ∅
    for each c ∈ contours do
        fitted_chains ← FITDOO(c)
        Chains.insert(fitted_chains)
    end for
    ▷ Prune all the overlapping segments
    Chains ← PRUNE(Chains)
    ▷ Merge the DOO chains into a single DOO
    while Chains.length > 1 do
        C_1, C_2 ← FINDBESTMERGEMATCH(Chains)
        C_merged ← MERGECHAINS(C_1, C_2)
        Chains.remove({C_1, C_2})
        Chains.insert(C_merged)
    end while
    return Chains[0] ▷ Return the final DOO chain
end function
```

where $\|\cdot\|$ is the number of data points in the collection.

The simplest segmentation methods include color-based filtering and background subtraction, which can work in lab settings, but more complex methods are required for real-world applications.

The medical research community provides an extensive set of segmentation methods to address vein and vessel detection, which can be used here directly or with small modifications. Such methods include both model-based [48, 56, 94, 118, 134, 196] and learning-based [197, 198] approaches and are often robust to clutter in the input image and can work in low signal-to-noise conditions.

The requirement for the segmentation method for our work is to filter the DOO data conservatively, i.e., ideally, it should eliminate all the unrelated data even if it removes some of the DOO data. Figure 6.3 illustrates the segmentation of an example cable in the camera frame.




(a) (b)

**Figure 6.3:** The segmentation of a DOO in a camera frame. (a) The original image. (b) The segmentation result.

### 6.3.2 Topological Skeletonization

Skeletonization transforms each segmented connected component into a set of connected pixels with single-pixel width called a *skeleton*. It is commonly used in the pre-processing stage of various applications ranging from Optical Character Recognition (OCR) to human motion tracking, fingerprint analysis, and various medical imaging analysis [44, 80, 163].

Our algorithm has two requirements for choosing the skeletonization method: a) the skeleton of a connected component should remain a connected component; b) only one branch should be returned per actual branch (i.e., multi-branching of a single skeleton branch should be avoided). The skeletonization algorithm chosen for this step should inherently respect the two constraints. Figure 6.4

shows the skeletonization of the segmented example of Figure 6.3.

**Figure 6.4:** The skeletonization of a segmented deformable one-dimensional object.

### 6.3.3 Contour Extraction

A contour (a.k.a., boundary) is an ordered sequence of the pixels around a shape. Extracting contours from an image is utilized in many applications ranging from shape analysis to semantic segmentation and image classification [53, 85].

Ideally, the contour extraction method applied to the skeletons should result in one contour per skeleton branch. However, the contour extraction methods can result in several contours per branch and some contours containing multiple branches in practice. Moreover, a contour contains the closed boundary *around* the skeleton and not the actual skeleton pixels. Figure 6.5 shows different types of contours that can be extracted from a skeleton piece.

**Figure 6.5:** Contour types extracted from a skeletonized image. The black area around the gray skeleton is the contour.

Our DOO detection method can handle the above-mentioned common issues raised by the contour extraction methods. Therefore, many of the existing contour extraction methods can be used with our algorithm regardless of their output limitations. Figure 6.6 presents the result of contour extraction.

**Figure 6.6:** The contours extracted from the skeleton. Each contour is drawn with a different color.

Contours facilitate traversing points along the skeleton and simplify determining the connections in the branches. The contour extraction step can be skipped if an ordered set of pixels for each skeleton branch is obtained from the skeletonization method or other means.

### 6.3.4 Fitting DOO Segments

The next step is to fit a chain of fixed-length segments to each contour. A contour can be a single branch, or it may contain multiple branches (see Figure 6.6). The pixel sequence for a contour returned by a typical contour extraction method starts from one of the tips and ends with a sharp turn back at the start point.

Let us call the latest added segment as $s$, the current segment as $s'$, the first point in the contour sequence as the starting point $p_s$ of $s'$, and the point currently being traversed as $p_c$. Let us also define $\vec{s}$ as the vector in the direction of segment $s$, starting at its start point and pointing towards its end

point. Therefore, starting with an empty DOO chain, the points are traversed while the distance $\|p_c - p_s\|_2$ is less than $l_s$. Then a new segment of length $l_s$ is added to the DOO chain from point $p_s$ in the direction of $p_c$, the new segment's end-point $p_e$ is saved as the next segment's start point $p_s$, and the traversal continues.

Three conditions may happen during the traversal:

1. Vector $\vec{s'}$ is close to vector $\vec{s}$: The new segment is added to the current DOO chain in this case.
2. Vector $\vec{s}'$ is close to vector $-\vec{s}$: It means that the traversal has gone over a branch end. In this case, the current DOO is recorded without the new segment, and the new segment is discarded. The traversal continues from the branch tip with a new empty DOO chain.
3. The last point in the contour sequence is reached: it means that the traverse has returned to the start point, and the traverse can be terminated. There may be cases where the whole contour length is less than the segment size. In these cases, no new DOO chains will be generated.

Algorithm 6 shows the pseudo-code for the described steps.

### 6.3.5 Pruning

The segment fitting algorithm of Section 6.3.4 returns multiple overlapping DOO chains for each part of the object. It is desired to prune the overlapping segments to reduce the total number of segments and simplify the further steps by assuming that no two segments overlap.

To define the overlap of two segments, each segment can be assumed as a rotated rectangle in the 2-D case and a square cuboid (a cuboid with two square faces) for the 3-D case. The length of the rectangle and cuboid is the segment length $l_s$, and the rectangle's width is 3 pixels or higher. The reasoning for the choice of the width is that the width of the skeleton is generally 1 pixel with occasional width of 2 pixels (depends on the choice of the thinning

**Algorithm 6** Traversing contours for DOO chain creation.

▷ Traverses a contour and returns all created DOO chains.
**function** TRAVERSECONTOUR(*contour*)
  ▷ Initialize the collection of DOO chains
  $Collection \leftarrow \emptyset$
  ▷ Initialize the start point and the next DOO chain
  $p_s \leftarrow contours[0]$ , $chain \leftarrow \emptyset$
  ▷ Initialize the tip point
  $p_t \leftarrow p_s$ , $update\_tip \leftarrow True$
  ▷ Traverse over all the points in the contour
  **for each** $p_c \in contour$ **do**
    ▷ Create a new segment if $p_c$ is far enough
    **if** DIST$(p_c, p_s) \geq l_s$ **then**
      $p_e \leftarrow p_s + l_s \times (p_c - p_s)/\|p_c - p_s\|$
      $s' \leftarrow$ CREATESEGMENT$(p_s, p_e)$
      **if** $chain = \emptyset$ or ANGLE$(s, s')$ is small **then**
        ▷ Add the segment if it is the first in chain
        ▷ or if the direction has not changed much
        $chain$.Insert$(s')$
        $s \leftarrow s'$ , $p_s \leftarrow p_e$ , $p_t \leftarrow p_e$
      **else**
        ▷ Start a new chain if direction has changed
        $Collection$.Insert$(chain)$
        $chain \leftarrow \emptyset$ , $p_s \leftarrow p_t$
      **end if**
      $update\_tip \leftarrow True$ ▷ Start updating tip point
    **else if** DIST$(p_c, p_s) \geq$ DIST$(p_t, p_s)$ **then**
      **if** $update\_tip = True$ **then** $p_t \leftarrow p_c$ ▷ Update the tip point
    **else**
      $update\_tip \leftarrow False$ ▷ Stop updating tip point
    **end if**
  **end for**
  $Collection$.Insert$(chain)$ ▷ Add the last generated chain
  **return** $Collection$ ▷ Return all the chains at the end
**end function**

method). The contour is the boundary around the skeleton, and for the two rectangles on the two sides of the skeleton to overlap, they need to be at least 3 pixels wide. In practice, any small number greater than 3 pixels should work

well for pruning the overlapping segments. For the 3-D case, the width of the cuboid can be chosen similarly, i.e., it should be at least the total of the width of the contour layer and the maximum width of the skeleton. Once the segments are defined, the geometric intersection of two rotated rectangles or cuboids is used to find the overlap. Figure 6.7 shows how two segments can overlap for the same skeleton.

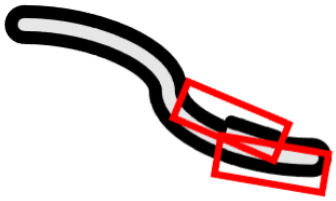


**Figure 6.7:** An illustration of overlapping segments around a skeletonized deformable one-dimensional object.

A heuristic that has shown performance improvements in the subsequent stages removes the segment from the shorter chain when two segments overlap. This heuristic will result in many chains being quickly emptied, which reduces the overall number of DOO chains in the collection.

Figure 6.8 shows the chains resulting from segment fitting and then pruning of the contours in Figure 6.6.

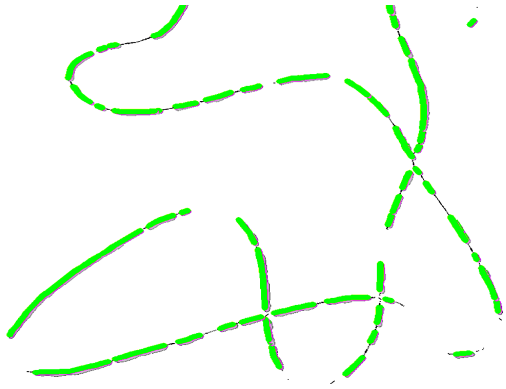


**Figure 6.8:** The result of segment fitting from contours and pruning for a deformable one-dimensional object of Figure 6.4.

### 6.3.6 Merging

Once we have a collection of DOO chains, they should be merged to fill the gaps and form a single object. A gap can result from an occlusion or an imperfect segmentation and should be filled in a way that follows the natural curve of the deformable object.

The merging process can be performed iteratively, connecting two chains at a time until all the chains are merged into a single deformable one-dimensional object. Each iteration can be broken down into two steps:

1. Choose the best two chains for merging
2. Connect the selected chains

The following subsections describe choosing the best chains and properly connecting them with their natural bend.

### 6.3.7 Merging: Choosing the Best Matches

To choose the best chains to connect, we define a new cost function $C_M(\cdot)$ that calculates the cost of connecting any two chain ends. Considering that there are two ends for each chain, there will be four cost values for connecting the two ends for any two chains. The lowest among the four values is the cost of connecting the two chains.

Given an end segment $s_1$ of the first chain and an end segment $s_2$ of the second chain, three separate partial costs are defined and then combined to create the total cost function $C_M(\cdot)$:

- Euclidean Cost $C_E$: This measure incurs costs to two chain ends based on their Euclidean distance to deter the early connection of far away ends (Figure 6.9(a)). Having the end segments $s_1$ and $s_2$, this cost can be calculated as:

$$C_E(s_1, s_2) = \|s_1.end - s_2.end\|_2, \tag{6.2}$$

where $\|\cdot\|_2$ is the norm of the resulting vector and $s.end$ is the end point of the segment (end point of the chain).

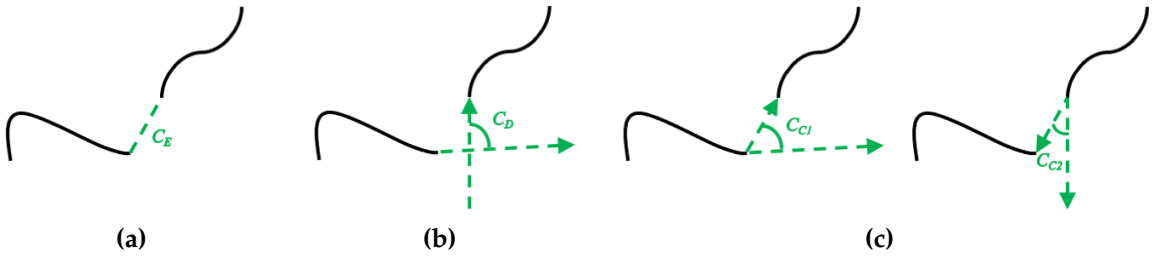



**Figure 6.9:** Illustration of different partial costs of merging two chain ends. (a) Euclidean cost. (b) Direction cost. (c) Curvature costs from first chain to the second and from the second chain to the first.

- Direction Cost $C_D$: This measure incurs costs to two chain ends based on the difference between the direction of the first with the opposite direction of the second (Figure 6.9(b)). This cost discourages the connection of chains that are not facing each other. Having the end segments $s_1$ and $s_2$, this cost can be calculated as:

$$C_D\left(s_1, s_2\right) = \left|\arccos\left(\frac{-\overrightarrow{s_1}\cdot\overrightarrow{s_2}}{\|s_1\|\|s_2\|}\right)\right|, \tag{6.3}$$

  where $|\cdot|$ is the absolute value, $\|\cdot\|$ is the norm of the vector (size of the segment), $\vec{s}$ is the vector along the segment $s$ starting from the start of the segment and ending at the end of the segment (i.e., the end of the chain), and $\cdot$ is the inner product operator.

- Curvature Cost $C_C$: This measure incurs costs to two chain ends based on how much curvature is needed to connect them. It is calculated as the higher cost of bending the first chain's end segment towards the second chain's end segment and vice versa (Figure 6.9(c)). The measure is defined to discourage connections requiring excessive bending and to encourage smooth connections. Having the end segments $s_1$ and $s_2$, this cost can be calculated

as:

$$\begin{aligned} C_{C1}\left(s_1, s_2\right) &= \left|\arccos\left(\frac{\overrightarrow{s_1} \cdot \overrightarrow{s_{21}}}{\|s_1\|\|s_{21}\|}\right)\right| \\ C_{C2}\left(s_1, s_2\right) &= \left|\arccos\left(\frac{\overrightarrow{s_2} \cdot \overrightarrow{s_{12}}}{\|s_2\|\|s_{12}\|}\right)\right| \\ C_C\left(s_1, s_2\right) &= \max\left(C_{C1}, C_{C2}\right), \end{aligned} \tag{6.4}$$

where $s_{nm}$ is a shorthand for $s_n.end - s_m.end$.

Having the three cost values for the ends of two chains, the total cost of these ends is computed as:

$$C_M(s_1, s_2) = \mathscr{F}\Big(C_E\left(s_1, s_2\right), C_D\left(s_1, s_2\right), C_C\left(s_1, s_2\right)\Big), \tag{6.5}$$

where $\mathscr{F}$ is the function combining the three values. In practice, we learned that the weighted sum of the values works well, and even after manually choosing a simple weight set, the algorithm works for almost all kinds of situations (see Section 6.4 for our test values). With the weighted sum, the Equation 6.5 reduces to:

$$C_M(s_1, s_2) = w_e \cdot C_E\left(s_1, s_2\right) + w_d \cdot C_D\left(s_1, s_2\right) + w_c \cdot C_C\left(s_1, s_2\right) \tag{6.6}$$

After calculating the four costs of all end combinations of the two chains, the minimum of those costs is the cost of merging the two DOO chains. Once the costs for all pairs of chains are calculated, the two chains with the lowest total merging cost are chosen for merging.

Note that the choice for the cost function of Equation 6.5 is to encourage the connection of closer chains that align well and can connect smoothly. Choosing a single measure such as minimum curvature would result in unwanted connections of farther chains that align perfectly over closer chains that are slightly misaligned.

### 6.3.8 Merging: Connecting Two Chains

Once two chains $C_1$ and $C_2$ are selected for connection (see Section 6.3.7), the gap between the two chains should be filled with a new chain $C_{new}$ in a way that it follows the expected curve of the deformable object. Our experiments show that any deformable object can take almost any curve given different pressure points, forces, tensions, and the object's condition. However, it is possible to have an educated *guess* on how the object behaves. For this purpose, we calculate the "natural" curvature required for the new chain $C_{new}$, which connects the desired end of $C_1$ to the desired end of $C_2$.

To compute the "natural" curvature, we assume that the new chain $C_{new}$ starts in the same direction as the two desired chain ends. In other words, at each end, $C_{new}$ initially follows the direction of the last segment of the chain to which it is connected. On the other hand, we assume that when it is possible, the deformable one-dimensional object will follow a curve with a constant turn rate (i.e., constant radius). With these assumptions, we can find two circles tangent to the lines passing through the two chain ends, each passing through one of the chain end-points. Based on triangle similarity theorems, the radii of the two circles are proportional to the distances of the chain ends from the intersection point. Figure 6.10 illustrates this idea.

Let us define the end-points we desire to connect on chains $C_1$ and $C_2$ as $e_1$ and $e_2$, respectively. We call the lines passing through $e_1$ and $e_2$ in the direction of $C_1$ and $C_2$ ends as $l_1$ and $l_2$, respectively. Finally, we define the circles passing through $e_1$ and $e_2$ as $c_1$ and $c_2$, and the points they touch on the other line as $t_1$ and $t_2$, respectively. Note that points $e_1$ and $t_2$ will be lying on line $l_1$, while points $e_2$ and $t_1$ are on line $l_2$.

Without loss of generality, let us assume that in Figure 6.10, circle $c_1$ is the red circle, the blue circle is $c_2$, the red dot is $t_1$, the blue dot is $t_2$, the red arrow's end point (arrow side) is $e_1$, the blue arrow's end point is $e_2$, the line passing $e_1$ is $l_1$ and the line passing $e_2$ is $l_2$.

It can be proven that the distance between $e_2$ and $t_1$ is equal to the distance

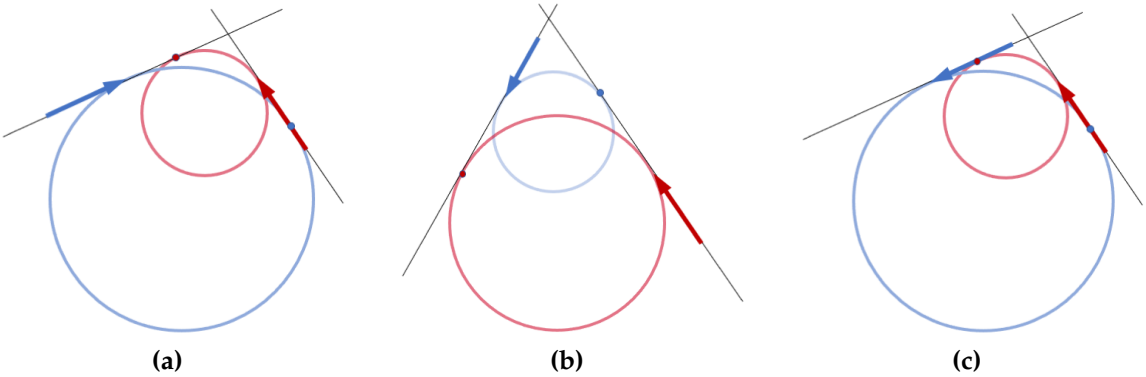



**Figure 6.10:** Illustration of different merging scenarios with the two circles tangent to the line passing the end points of the two chains, each circle passing through one of the end points. Arrow ends and directions represent the end points and end directions of the chains. (a) Exactly one of the circles passing through the end point of a chain is touching the other line ahead of the other chain. (b) Both the circles passing through the end points of the chains are touching the other line ahead of the other chain. (c) None of the circles passing through the end point of a chain are touching the other line ahead of the other chain.

between $e_1$ and $t_2$. However, each $t_1$ and $t_2$ can be lying on lines $l_2$ and $l_1$ ahead or behind $e_2$ and $e_1$, creating three different situations:

- Either $t_1$ is ahead of $e_2$ or $t_2$ is ahead of $e_1$, but not both (Figure 6.10(a)). Not surprisingly, a majority of connections in a typical application would be of this type. In this case the blue circle $c_2$ that touches $l_1$ at point $t_2$ behind $e_1$ is discarded and we use the radius of the red circle $c_1$ for the turn radius of the new chain $C_{new}$. This chain will be composed of the arc of $c_1$ from $e_1$ to $t_1$ and the line from $t_1$ to $e_2$. Figure 6.11(a) shows this scenario's solution.

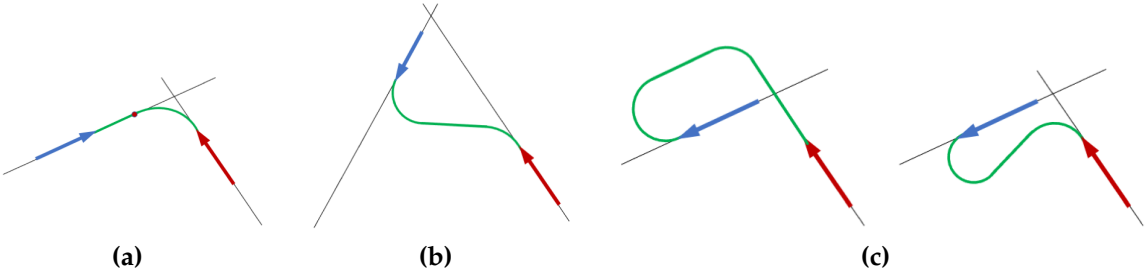



**Figure 6.11:** Suggested solutions for different merging cases illustrated in Figure 6.10.

- Both $t_1$ and $t_2$ are ahead of $e_2$ and $e_1$ (Figure 6.10(a)). In this case, the new chain $C_{new}$ is composed of an arc on each end ($e_1$ and $e_2$) and a line tangent to the arcs. The turn radius is as desired or can be experimentally determined for the DOO, and it should be large enough to allow the "natural-looking turn." However, the radius should be small enough so that the direction of the line tangent to the two arcs is close to the direction of the line connecting $e_1$ to $e_2$. Finally, we suggest the same turn radius for both ends. Figure 6.11(b) shows this scenario's solution.
- Both $t_1$ and $t_2$ are behind of $e_2$ and $e_1$ (Figure 6.10(c)). This case has two suggested solutions that depend on the conditions. In both solutions, similar to the previous case, the new chain $C_{new}$ is composed of an arc on each end ($e_1$ and $e_2$) and a line tangent to the arcs. The turn radius is as desired or can be experimentally determined for the DOO. However, depending on external conditions, $C_{new}$ can fill the gap from outside or inside the region between the two chains. Figure 6.11(c) shows this scenario's solutions.

Note that, in all scenarios, there can be infinite correct solutions, which depend on the conditions. In practice, the suggested solutions result in good fits with the ground truth and can be used in most conditions without any modification.

Once the new chain $C_{new}$ is obtained to fill the gap between the two chain ends $C_1$ and $C_2$, it can be added to the ends of the chains to connect them. $C_{new}$ is a combination of constant-radius arcs and lines. Adding a DOO segment for the line sections is trivial and will not be explained. To add a segment that follows the desired turn, we use the last segment $s$ on the chain. Knowing the start and end-points of this segment $s$, we can calculate two circles with the desired radius that pass through these points. Knowing the direction of the turn, one circle is eliminated, and the new point on the remaining circle at the segment distance $l_s$ of the segment's end-point is used to create the new segment $s'$ that is added to the end of the chain. Figure 6.12 shows how the new segment can be added with the desired turn radius.

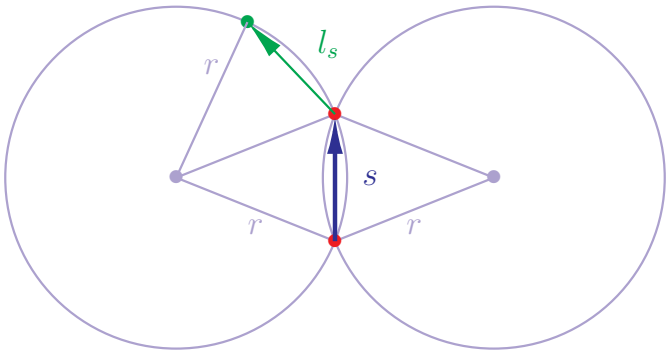



**Figure 6.12:** Adding a new DOO segment with length $l_s$ with a desired constant turn radius of $r$ to the end of segment $s$.

### 6.3.9 Notes on the Proposed Method

The merging process continues connecting the chains, two at a time until all the chains are merged into one single chain, which is the detected DOO represented by the chain of fixed-length cylindrical segments connected by passive spherical joints. This representation can be used as the input to routing and manipulation systems for the desired application.

Each segmented image will be processed into a single DOO. When there are multiple DOOs present in the camera frame, the easiest way to detect them as separate DOOs is to have separate segmentations for them. For example, if there are multiple cables with different colors in the image, a color-based segmentation can have two segmented images. In case there are multiple DOOs that cannot be segmented separately, the proposed algorithm can still help process them into separate DOO outputs. Note that the algorithm is greedy in the sense that it first goes for the best-matched chains. With multiple DOOs, there is a high chance that the chains related to separate DOOs do not give a good fit. As a result, for example, when there are only two chains left, there is a high chance that the two chains are the two separate DOOs. Therefore, it is enough to stop the merging process when the desired number of chains is left.

The proposed method is general and can be used with both 2-D and 3-D image data to provide the deformable object's representation in 2-D or 3-D.

Finally, the final output of the proposed method is a single chain of segments that does not keep track of the parts seen in the frame vs. the occluded parts. There are two ways to mark the parts of the chain that are related to the occlusions:

1. Map the segmented parts on the final detection to determine the occluded parts of the DOO chain.
2. During the process, when merging two chains, if the gap is equal to or longer than the segment length $l_s$, the newly added chain $C_{new}$ is marked as occluded. The reason for skipping smaller gaps is that many gaps shorter than $l_s$ are created during the pruning process.

Both approaches ultimately depend on the accuracy of the segmentation. The first approach is simple but may mislabel an occluded part as visible in a multilayer setup. On the other hand, the second approach tends to be more accurate in multilayer settings but may skip more minor occlusions.

## 6.4 Experiments and Results

The proposed was implemented for 2-D images in Python 3. We used the color-based segmentation of the DOO region. This approach generally tends to include extra areas around the DOO and other regions with similar color hues to the DOO. We chose conservative thresholds to exclude any non-DOO regions. This results in some DOO data being excluded; However, our experiments have shown the DOO detection to have challenges when extra regions are included but to work when some data is lost in segmentation. The same principle is advised for other segmentation methods choices, and those methods' parameters should be chosen conservatively to remove the irrelevant regions.

We used a well-known morphological thinning method for skeletonization [111]. The algorithm proposed by Suzuki and Abe [177] and provided in the OpenCV library is used to extract contours. All our tests use $w_e = 1$, $w_d = 100$ and $w_c = 100$ values for the cost function of Equation 6.6 and the

segment length $l_s$ is chosen as 10 pixels. The weights are chosen manually to focus on shorter distances while heavily discouraging non-matching segment directions and excessive bending. Different weights result in some types of incorrect connections increasing while the number of other types decreases. A better set can be found using a more methodical approach and optimization for the desired applications.

Similarly, the segment length was not chosen optimally. In general, a shorter segment length can capture the contour ends better and create segments from smaller contours, leading to better curves in filling gaps; however, it increases the number of segments and the average number of incorrect connections. On the other hand, a longer segment length has the potential of not following the curves well and ignoring small contours. However, it tends to reduce the number of incorrect connections and improve the detection speed by reducing the number of segments.

Figure 6.13 shows results of the detection on inputs with heavy occlusion and several crossings.

We have used the method for cable routing and manipulation tasks [86]. The viability is tested on 7 video sequences with a total of 4,230 frames of size 1280×720. Table 6.1 shows the quantitative results for the algorithm's accuracy on the whole cable in an image, for the occlusions filled, and for the merges performed. Mengyuan et al. [203] have used the root mean square of the Euclidean distance between their estimated and the ground-truth point positions on the DOO, which they reported as around 23 mm. Note that due to the lack of ground truth for the occluded areas and to focus on testing the key contributions of our proposed approach, we used a stricter measure that even when a single connection is incorrect, we counted the frame as incorrect detection. The occlusions are counted as incorrect when either a wrong connection is made or when the filled connection does not follow the actual cable's path. Finally, we noticed that the incorrect merges only rarely happen in places other than at occlusions, with only 148 cases, almost all of which happened at self-crossings.

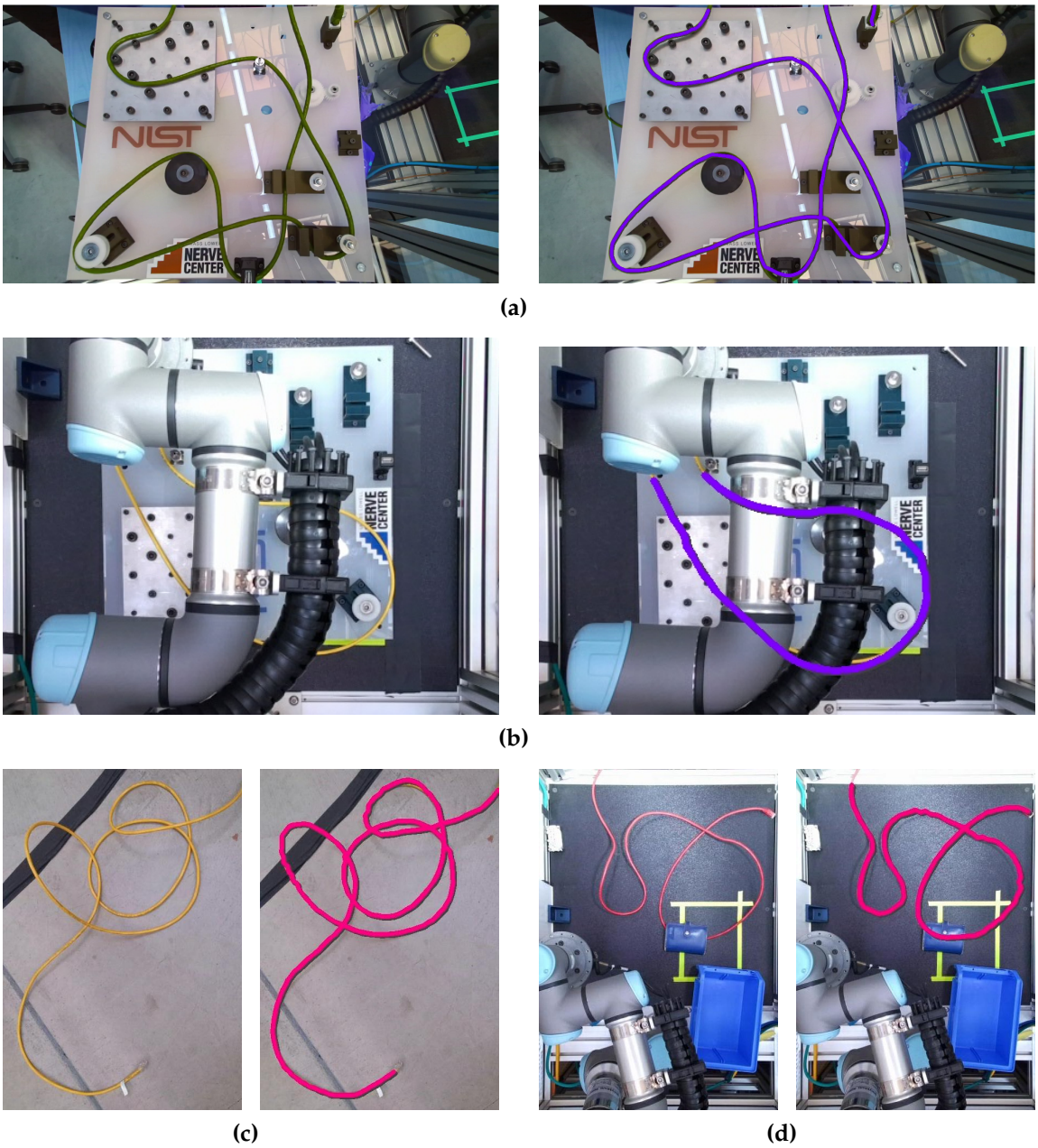


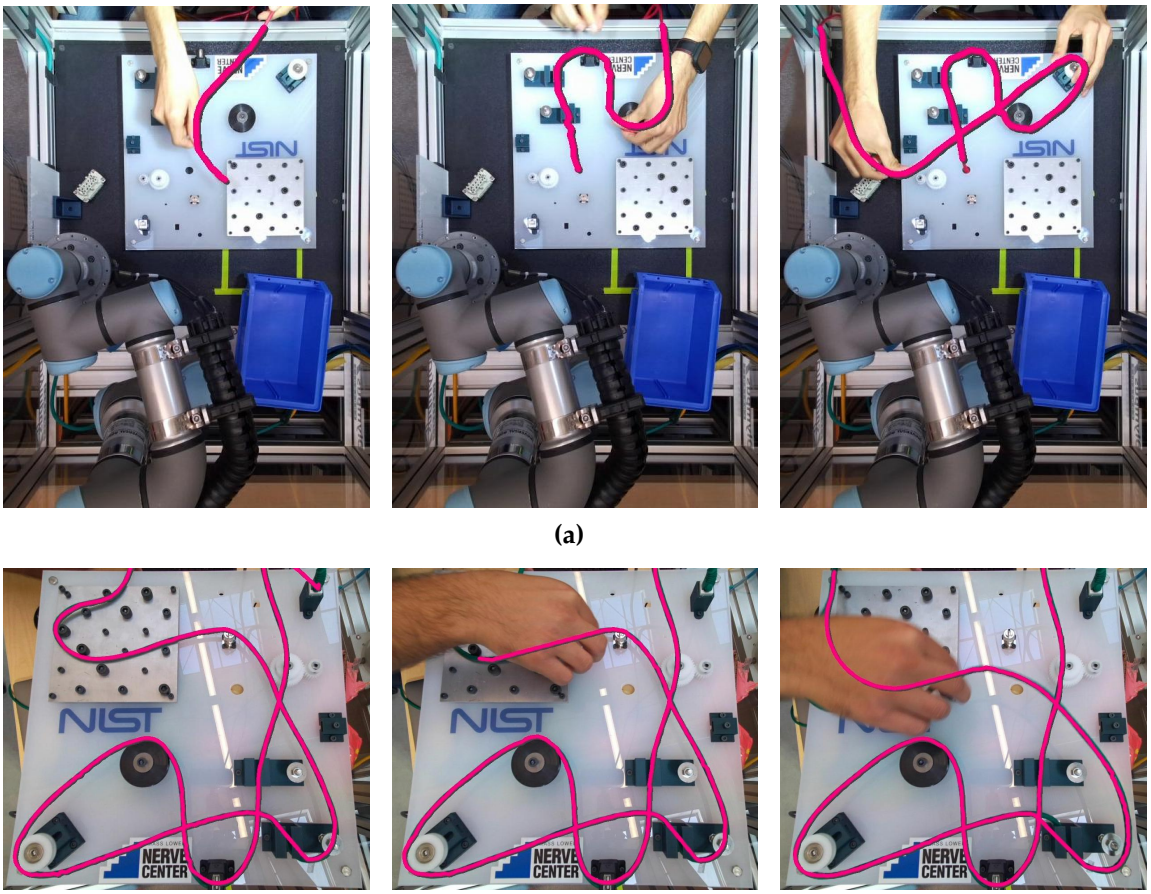

**Figure 6.13:** The result of the proposed detection method on example inputs with crossings and occlusions. The detected cable (purple and magenta) overlaid on the frames on the right.

Our method's average detection time per frame across all the sequences is 0.537 seconds on a system with Intel® Core™ i9-10885H CPU and 64 GB DDR4 RAM. Figure 6.14 shows snapshots of some video sequences and the detection

Table 6.1: Detection results on 7 video sequences.

| | Total | Correct | Incorrect | Accuracy |
|---|---|---|---|---|
| Frames | 4,230 | 3,542 | 688 | 83.7% |
| Occlusions | 26,456 | 23,991 | 2,465 | 90.7% |
| Merges | 583,743 | 581,130 | 2,613 | 99.6% |

results.




**(a)**

**(b)**

**Figure 6.14:** Screenshots of example sequences with the overlaid detected cable (in magenta). The third row includes the original frame for comparison.

## 6.5 Conclusion and Discussion

We presented a novel method for detecting deformable one-dimensional objects (e.g., wires and cables) for robotics applications and showed the results. Our implementation is only 2-D and not tuned towards a specific condition. Choices other than the weighted sum for the total cost function were not researched, and our selection of weights was not made optimally. Nevertheless, the results show promise with an almost 2 Hz detection rate on an HD image input, which is enough to initialize the DOO trackers for inspection and maintenance with UAVs. A more optimized implementation can take advantage of special data structures and parallelization to increase the method's speed by several orders of magnitude.

Furthermore, the proposed method is very general and flexible and can be tuned for specific 2-D and 3-D applications to provide near-perfect results in other settings as well, including surgical and industrial robots.

The cable segmentation method proposed by Li [103] provides a cost function for choosing the best two chains for merging. While we developed our cost function elements independently, the overall cost functions between our work and Li have similar structures, only differing in details. However, the Li method only uses merging for neighbor image patches, and the intention is not to fill the occlusions and gaps but is pure segmentation. We go further by providing different merging solutions for different conditions to deal with occlusions and imperfect segmentation. Our final output is a DOO representation suitable for manipulation instead of the segmentation mask provided by the Li method.

Note that it is not hard to find unstructured or adversarial situations with entanglements, occlusions, multiple close and parallel DOOs, and other complex scenarios that can easily confuse the proposed algorithm. This work is the first effort to solve the DOO detection problem and was aimed to provide a method that can assist in semi-structured situations rather than addressing those "crazy" scenarios.

The considerations for the 3-D case are provided for each step. However, we did not implement the 3-D case, and there may be unpredicted implementation challenges. In the future, the ideas of the method can be integrated with tracking methods to improve tracking accuracy. Its integration in a robotics pipeline can eventually enable full autonomy in real-world robotics applications working with DOOs such as cables, surgical sutures, and ropes and is a step toward realizing aerial manipulation of these objects.

The following chapter introduces our routing solution for the manipulation of DOOs and explores how aerial manipulators can have physical interaction with the detected DOOs.

Chapter 7

# Deformable One-Dimensional Object Routing and Manipulation

With the field of rigid-body robotics having matured in the last fifty years, routing, planning, and manipulation of deformable objects have recently emerged as a more untouched research area in aerial robotics and many other fields ranging from surgical robotics to industrial assembly and construction.

Routing approaches for deformable objects which rely on learned implicit spatial representations (e.g., Learning-from-Demonstration methods) make them vulnerable to changes in the environment and the specific setup. On the other hand, algorithms that entirely separate the spatial representation of the deformable object from the routing and manipulation (often by using a general representation approach independent of planning) result in slow planning in high-dimensional spaces.

This chapter proposes a novel approach to routing deformable one-dimensional objects (e.g., wires, cables, ropes, sutures, threads) [86]. This approach utilizes a compact representation for the object, allowing efficient and fast online routing. The spatial representation is based on the geometrical decomposition of the space into convex subspaces, resulting in a discrete coding of the deformable object configuration as a sequence. With such a configuration, the routing

problem can be solved using a fast dynamic programming sequence matching method that calculates the next routing move. The proposed method couples the routing and efficient configuration for improved planning time. Our simulation and real experiments show the method correctly computing the next manipulation action in sub-millisecond time and accomplishing various routing and manipulation tasks.

Then, analyze the requirements of aerial manipulation of deformable one-dimensional objects [88]. We study the feasibility of the physical interaction of UAVs with these objects from the perspective of their end-effector precision and their available wrenches for an example wire manipulation task.

## 7.1 Introduction and Related Work

Objects such as wires, cables, ropes, threads, and surgical sutures can be found in many industrial, surgical, construction, and everyday settings. In the literature, they are commonly called Deformable One-dimensional Objects (DOOs) or Deformable Linear Objects (DLOs). Automation of tasks involving rigid bodies had been extensively studied in robotics; however, the need for further automation of manual tasks is forcing robotics applications to move towards working with DOOs, raising research interest in aerial as well as other robotics areas [5, 165].

A crucial part of many robotics applications involving DOOs is routing [55, 74, 165]. *Routing*, also known as *route planning*, finds a viable *path* to change the initial state of a DOO to a goal state. Path in this context depends on the application and may mean a path in Euclidean space, a trajectory in the configuration space of the DOO, or a series of actions performed on the DOO.

Several representation methods have been used to capture the state of a DOO in route planning and manipulation problems. These representations range from spring-mass models [108] to linear combination of curves [59, 122], and fixed-length segments [87].

Planning methods for routing DOOs for manipulation can be categorized into computational algorithms and approaches based on learning.

Computational routing methods are generally sampling-based approaches. These methods sample the space and use search methods such as Probabilistic Roadmaps (PRM) and Rapidly-exploring Random Trees (RRTs) to find the route from the initial DOO state to the final state. Guo et al. [55] propose the RRT-BwC (Bi-direction with Constrain) planning algorithm to plan for aircraft cable assembly in narrow cabins with obstacles. Their method is based on the geometric formulation of the objects and the bi-directional RRT search to route in the high dimensional planning space. Amato et al. [74] find an approximate route by pre-computing a global roadmap using a variant of PRM, then refine the route by constrained sampling and applying adaptive forward dynamics. Moll and Kavraki [122] propose DOO planning using minimal-energy curves and a sampling-based planning method such as PRM. Roussel et al. [155] use quasi-static and dynamic models coupled with sampling-based methods to plan for an elastic rod manipulation. Koo et al. [90] and Ma et al. [109] apply RRT search-based approaches for routing and manipulation of DOOs. For all these methods, the solution can be statistically guaranteed, but they suffer from the curse of dimensionality, and in complex routing cases, the time and space complexity of the algorithms may make the algorithms slow and infeasible for many practical scenarios.

On the other hand, the learning-based methods primarily consist of learning-from-demonstration approaches, potentially working with any DOO type for any task. However, they are not generalizable and quickly fail when the experiment setup conditions even slightly deviate from the learning data [58, 147, 195].

We propose a novel approach to solving the DOO routing problem that is suitable for both offline and online routing due to its efficiency and speed [86]. This approach relies on the geometrical decomposition of the task space into convex regions. It uses this discretized space to describe the DOO's configuration using a compact sequence, which is simplified from the original 3-D continuous space description. Unlike the existing routing methods that have

to find a solution by exploring a high-dimensional space, our new spatial representation allows utilizing a high-speed dynamic programming sequence matching method that reduces the planning delay to near zero, making it suitable for online planning for routing and manipulation tasks.

As far as we know, there have not been any studies on the requirements for physical interaction and manipulation of deformable objects using aerial robots. In this chapter, we perform a basic analysis of the requirements of aerial manipulation of deformable one-dimensional objects [88]. We study the feasibility of the physical interaction of UAVs with these objects from the perspective of precision of their end-effectors and their available wrenches for a wire manipulation task.

## 7.2 Problem Definition

In the rest of this chapter, we refer to the area where the task is taking place as *work region*. We assume that the exact positioning of a deformable one-dimensional object is only important inside the work region, and the details of its positioning outside this region are ignored. Additionally, we presume that the work region falls within the workspace of the aerial manipulator, and the robot can access all of the work regions. Moreover, we assume that the work region is a "free space" with different *components* occupying some of its space. All the components' positions, shapes, and dimensions are presumed to be known either beforehand or through processing the perception input. Figure 7.1 illustrates an example utility box that satisfies all the mentioned assumptions.

The exact positioning of a DOO in the work region is called its *state*, which contains the exact places in space occupied by a DOO.

Given the initial and desired states of a DOO in a work region, *routing* is finding the sequence of actions required to perform on the DOO to change its state from the initial state to the final state.

The actual DOO state is continuous and needs to be converted to a discrete

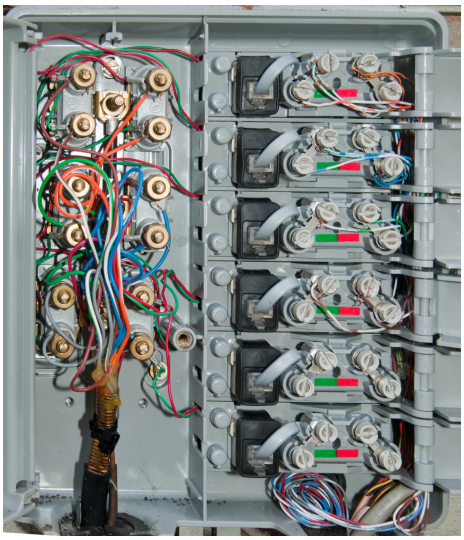


**Figure 7.1:** An example demarcation box or Network Interface Device (NID) [144].

spatial representation for robotics and computer applications. Common different ways of such representation include sampling equidistant points along the DOO's medial axis, fixed-length B-splines, and fixed-length cylinders connected by spherical joints [55, 87].

We propose to use a more efficient spatial representation based on the convex decomposition of the work region, combined with a fast sequence matching algorithm to solve the routing problem. Our proposed method is completely independent of the DOO dynamics and tries to embed the dynamics effects in the state representation. We relax the problem assuming that the *slack* of the DOO is not important. In other words, we assume that the application working with the DOO is not affected if the DOO has some extra slack in any area of the work region. For example, suppose a cable is not laying straight in a region and has a rather significant bending in a region. In that case, the extra bending is considered slack, and our algorithm does not consider it. We define the slack as extra curves in DOO that do not pass around any components or anchor points; the slack can virtually be eliminated by creating tension in the DOO.

The following sections describe our proposed spatial representation and the

routing method for DOOs.

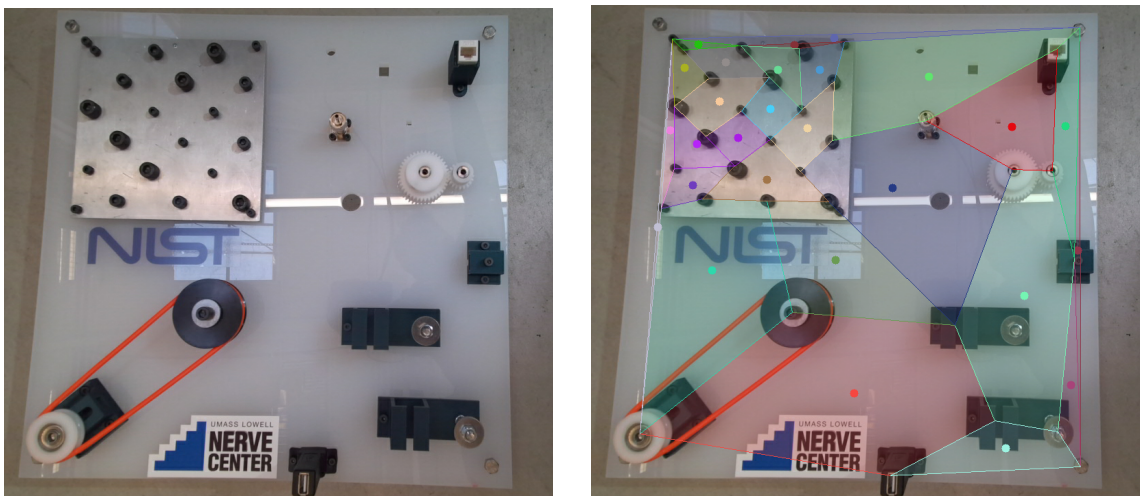
## 7.3 Spatial Representation

Let us introduce a graph $G_s$ (called a *spatial representation graph*) to model the spatial representation of a deformable one-dimensional object passing through a work region.

The work region should be decomposed into convex subspaces to generate the vertices $V_s$ of the spatial representation graph $G_s$. These subspaces are called *convex polygons* in 2-D and *convex polytopes* in 3-D spaces. Each of these subspaces is a vertex in the graph $G_s$. If the work region is not enclosed (i.e., if a portion of DOO can lie outside the work region), a new vertex is added to $V_s$ to represent the "outside" region. There are many exact and approximate approaches for convex decomposition, and each can be used for this work [11, 39, 105, 106]. Figure 7.2 illustrates the convex decomposition on an example circuit board. Each of the components on the board is a node of the convex polygons generated from the board's layout.

The generated convex regions allow efficiently defining subspaces in both 2-D and 3-D. It is desirable to represent the subspace only in 2-D when possible for simplicity. Much of the workspace in the finish line of industrial robotics is on a tabletop which can be approximated as a 2.5-D space. Meaning two dimensions are far more significant than the third dimension. There are specific scenarios where a 3-D work region can be simplified as a 2-D region with additional 3-D "tunnel"-like components such as bridges, passes, and tunnels. To allow the 2-D representation for these work regions, we can add a vertex to $V_s$ for each of the entrances of these components. Figure 7.3 shows all the vertices constructed from the example board of Figure 7.2(a) with the yellow dots representing the vertices of the entrances of the tunnel components.

Once all the vertices $V_s$ are defined, the edges $E_s$ for the spatial representation graph $G_s$ can be computed using the following rules:




**(a)** **(b)**

**Figure 7.2:** Convex decomposition of an example circuit board. (a) The original board. (b) The result of convex decomposition. Each component on the board is used for defining the convex region vertices. The centroid of each convex region is marked with a dot.

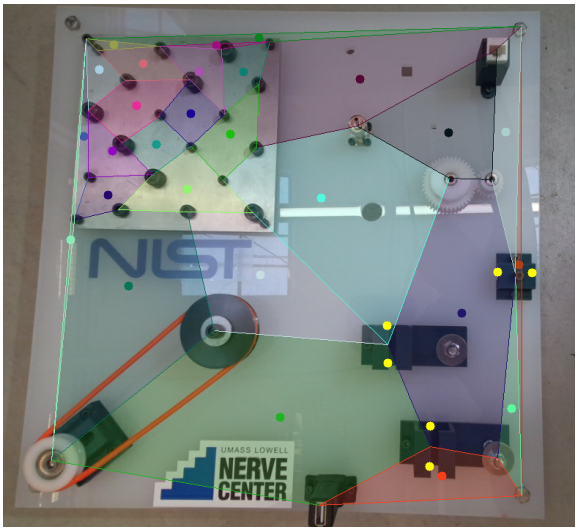



**Figure 7.3:** The vertices $V_s$ of the spatial representation graph $G_s$ constructed from the example board in Figure 7.2(a). The tunnel entrance vertices are depicted by yellow dots. Note that the outside region vertex is omitted in the illustration.

- Vertices from the neighbor convex regions (convex regions sharing a side) are connected with an edge.
- Convex regions with a side not shared with any other convex region (i.e., convex regions surrounding the work region) are connected to the outside

vertex.

- Vertices for entrances of a tunnel component are connected to each other.
- Each vertex for the tunnel entrances is connected to the vertex of the convex (or outside) region that it is lying on.

Figure 7.4 shows the spatial representation graph $G_s$ constructed for the circuit board of Figure 7.2(a).

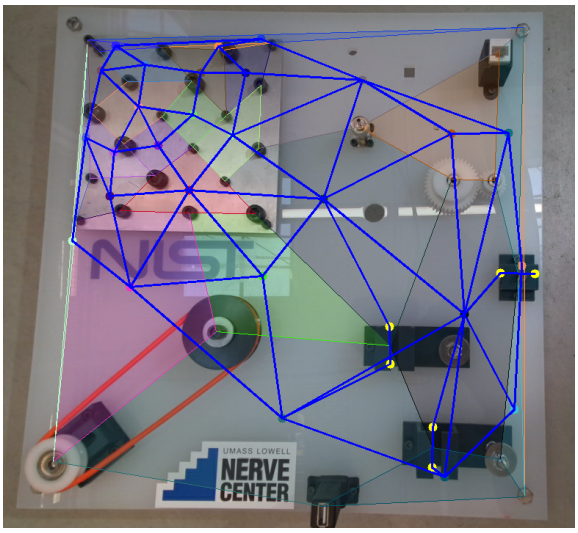



**Figure 7.4:** The spatial representation graph $G_s$ computed from the example board of Figure 7.2(a). For simplicity, the edges connected to the outside vertex are not depicted here.

Without the loss of generality, we assume that the size of $V_s$ is $n+1$, with the vertices numbered from $-1$ to $n-1$, and $-1$ reserved for the outside vertex.

Having computed the graph $G_s$, a DOO lying in the work region or passing through it can be represented by an ordered sequence $C$ of the vertex numbers it is passing through. We call this sequence representing the DOO as *configuration* of DOO. Note that if the DOO is bidirectional (does not have a pre-assigned head and tail), it can have two sequences for the same configuration that are reverse of each other. For example, the configuration of the DOO drawn on the circuit board in Figure 7.5 is $C = (-1, 1, 27, 28, 11, 4, 1, 6, 4, 15, 6, 9, -1)$ or its

reverse. Note that if the graph $G_s$ is computed correctly, every two consecutive vertices in $C$ should have an edge in $E_s$.


**Figure 7.5:** An example DOO passing through the spatial representation graph vertices of Figure 7.4.

Based on the assumptions of the problem (see Section 7.2), the extra slack of a DOO in each region is not encoded into its configuration. However, the extra slack is encoded if the DOO goes through some neighboring regions and comes back (e.g., if the DOO "touches" the neighbor convex region while passing through a convex region). Such instances are encoded as palindrome subsequences (i.e., subsequences that are the same if read backward or forward). Removing such subsequences may be desirable depending on the application and simplifies the configuration $C$ of a DOO in the work region.

We should note that the idea of convex decomposition in planning has been explored in other contexts before [57, 104, 131]. However, it is used differently here to define the DOO configuration rather than planning itself.

## 7.4 Routing Approach

Assume that the current configuration $C_0$ of a deformable one-dimensional object in a work region is provided along with the desired goal configuration $C_g$ of the DOO.

The problem is to route the DOO in the work region from the current configuration $C_0$ to the goal configuration $C_g$. A naive solution to the routing problem is to completely undo $C_0$ into a "free" DOO, then apply $C_g$ configuration by passing through all the vertices in $C_g$. However, this solution is inefficient and requires the maximum number of manipulative actions. A more efficient approach is to keep the matching areas between the current and goal configurations and only manipulate what is necessary to reduce the number of manipulative actions.

We propose utilizing the sequence matching algorithms to minimize the number of actions required to change from $C_0$ into $C_g$. Let us assume that the manipulator supports two motion primitives: 1) pick a DOO at a specific point, and 2) place the picked DOO at a specific point in the work region. Then the following actions on the configuration sequences can be applied:

- Replacing the $i^{\text{th}}$ element $s_i$ in $C_0$ with the $j^{\text{th}}$ element $g_j$ in $C_g$: Pick the DOO where it is passing through vertex $s_i$ and place it at vertex $g_j$.
- Removing the $i^{\text{th}}$ element $s_i$ in $C_0$ that does not correspond to an element in $C_g$: Remove the DOO from region $s_i$.
- Inserting the $j^{\text{th}}$ element $g_j$ in $C_g$ that does not correspond to an element in $C_0$: Adding (i.e., stretching) the DOO to region $g_j$.

With these three actions, we propose modifying the well-known Levenshtein sequence distance algorithm [100] to obtain the manipulation actions required for routing.

The original Levenshtein algorithm computes the minimum required edits (i.e., replacement, deletion, and insertion) to convert the initial sequence to

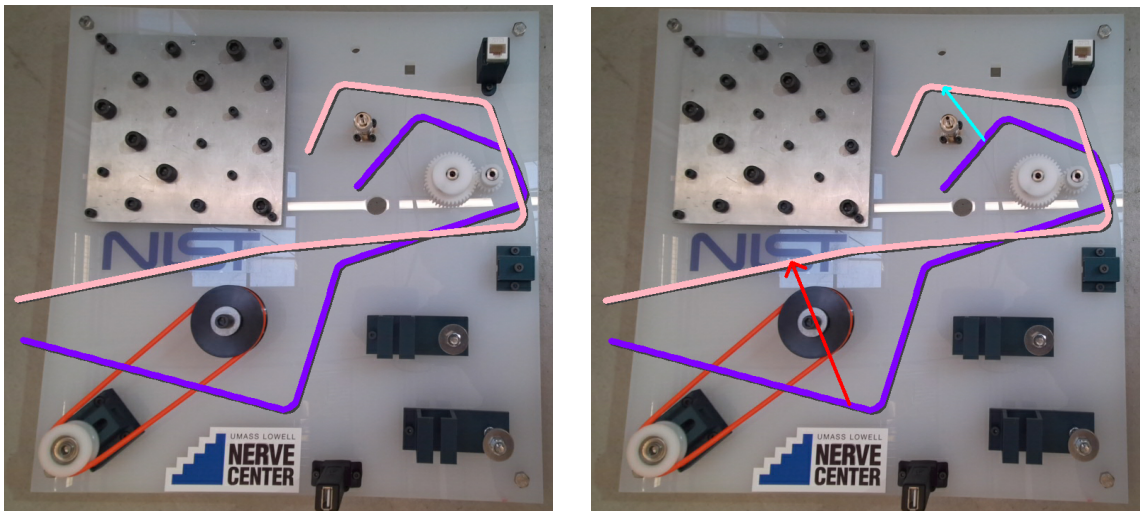
the final sequence. To return this minimum distance, Levenshtein's dynamic programming method computes a matrix that retains the minimum number of edits required for converting the first $i$ elements of the initial sequence to the first $j$ elements of the final sequence. While the algorithm itself only computes the minimum number of edits, the types of edits can be extracted by backtracing this matrix once the algorithm is finished. Note that these edits are not unique, and backtracing will only output only one of the feasible solutions with the minimum number of actions.

To use the Levenshtein algorithm for the routing problem, the following modifications are required:

1. When comparing two elements $s_i$ and $g_i$, they match if they are the same vertex number (i.e., $s_i = g_i$). However, if they are both $-1$ (the "outside" vertex), they only match if there is a common neighbor for them in the sequence (e.g., if $s_{i-1} = g_{i+1}$). In other words, the two outside regions are considered the same only if they are next to the same vertices. That same vertex may occur before or after $-1$.
2. The cost for each action is set to $1$. However, for a tunnel-like component, the action cost of either of the operations depends on how many vertices come before and after it. In other words, for the $i^{\text{th}}$ element in the sequence of size $n$, the cost will be $2 \times \min(i-1, n-i) + 1$. For example, to remove the DOO from a tunnel-like region, it must free either the start of the DOO or the end of the DOO and put everything back again, bypassing the tunnel.

If the DOO is bi-directional, the algorithm should be repeated with one of the sequences reversed to get the least number of actions. Then, backtracing can give the actions needed to perform on the DOO to change its configuration from $C_0$ to $C_g$. The time and space complexities of the algorithm are $\mathcal{O}(nm)$, where $n$ and $m$ are the lengths of the current configuration ($|C_0|$) and goal ($|C_g|$) configuration sequences. Figure 7.6 shows the routing actions for a DOO to get from its current configuration to the goal configuration.

To realize the computed actions, a single manipulator can act as below:




(a) (b)

**Figure 7.6:** Routing of a deformable one-dimensional object from the current configuration to a goal configuration. (a) The current (purple) and goal (pink) deformable objects drawn on the example board of Figure 7.2(a). (b) The result of the routing from current to final configuration. Red arrow shows a removal action and cyan arrow shows a move (replacement) action.

- For replacement of element $s_i$ in $C_0$ with element $g_j$ in $C_g$: Pick the DOO where it is passing through vertex $s_i$ and place it at vertex $g_j$.
- Removing element $s_i$ in $C_0$: Pick the DOO where it is passing through vertex $s_i$, and place it at the point where the goal DOO crosses from $g_{j-1}$ to $g_j$.
- Inserting element $g_j$ in $C_g$: Pick the DOO where it is crossing $s_{i-1}$ into $s_i$, and place it at (i.e., stretch it to) vertex $g_j$.

The details of how these actions are implemented depend on the motion primitives of the robot and the environment.

We call the point on the DOO where the picking happens as *picking point* and the points where the two sequences match (i.e., no action is required) as *fixed points*. If more than one manipulator is available, the second manipulator can grab the closest fixed point before the picking point, and the third manipulator can grab the closest fixed point after the picking point to prevent these points from moving.

The proposed routing algorithm does not incorporate the DOO dynamics and, therefore, cannot understand the result of the actions taken by the manipulator on the whole DOO configuration. To mitigate the lack of dynamics knowledge, routing and manipulation action can be performed iteratively until the current configuration matches the goal configuration. A single routing and then a manipulative action is performed at each iteration to get the DOO closer to the goal configuration.

## 7.5 Experiments and Results

We implemented it for a routing and manipulation task to test the proposed method. The task includes a single-arm manipulating a cable on the circuit board of Figure 7.2(a) to change its current configuration to a goal configuration. This board was originally designed for task #3 of the Assembly Performance Metrics and Test Methods by the National Institute of Standards and Technology (NIST) to measure the capability of robotics systems for performing advanced manipulation on cables [133]. The board was also adopted for the Robotic Grasping and Manipulation Competition in IROS 2020.

We performed many simulation experiments and several experiments with different settings on our robot. Each experiment included several iterations of routing and a manipulative action (i.e., pick and place actions) until the cable configuration had matched the given goal configuration.

We performed 200 simulation experiments, where we randomly placed a 0.3-0.5 $[m]$ cable on the 0.38 $[m]$ NIST board and randomly (in 170 tests) or manually (in 30 tests) placed a cable of the same length on the board as the goal configuration. The average number of actions over all the experiments was 4.34, and the maximum number of actions was 9. The processing time for each routing step was less than 1 $[ms]$ for all the experiments. Figure 7.6(b) illustrates an iteration of our simulated routing experiments.

One of the fastest DOO routing methods applicable in our scenario is

provided by Guo et al. [55]. They propose a bi-directional RRT-based method called RRT-BwC (Bi-direction with Constrain) to route DOOs. They report the execution time of 14.7 $[s]$ for a DOO routing task, which is a significant improvement over the other existing routing methods but several orders of magnitude slower than our method. Compared to our solution, their method uses a higher number of anchor (sampled) points on the cable and a higher dimension for the space, leading to a much higher dimensionality of the configuration space and a much slower planning problem. However, note that most routing solutions are designed for more general scenarios than ours and can work for scenarios where our solution would either fail or needs to be extended.

We manually placed the cables on the NIST board for the robot experiments. At each step of the planning, we utilized the DOO detection algorithm proposed in Chapter 6 to automatically detect the cable and extract its configuration on the circuit board. Then the goal configuration was manually given to the system. Our experiments showed that the method could also extend to real systems. Figures 7.7 and 7.8 show the routing and manipulation experiments using Universal Robots UR3 arm robot.

## 7.6 Analysis for Aerial Robot Manipulation

In general, ground manipulators have higher position precision compared to aerial robots. During a free flight or even physical interaction tasks such as painting a wall or pushing a box, the lower precision of the aerial robots may not affect the performance or the task's feasibility. However, many physical interaction tasks require more position precision, such as maintenance and cable manipulation at the top of utility poles with many wires and components around. For such tasks, the lower precision of the aerial manipulator end-effector can become a significant issue.

We analyzed the feasibility of manipulating the detected cable in both Gazebo and MATLAB simulation environments for our fully-actuated hexarotor with

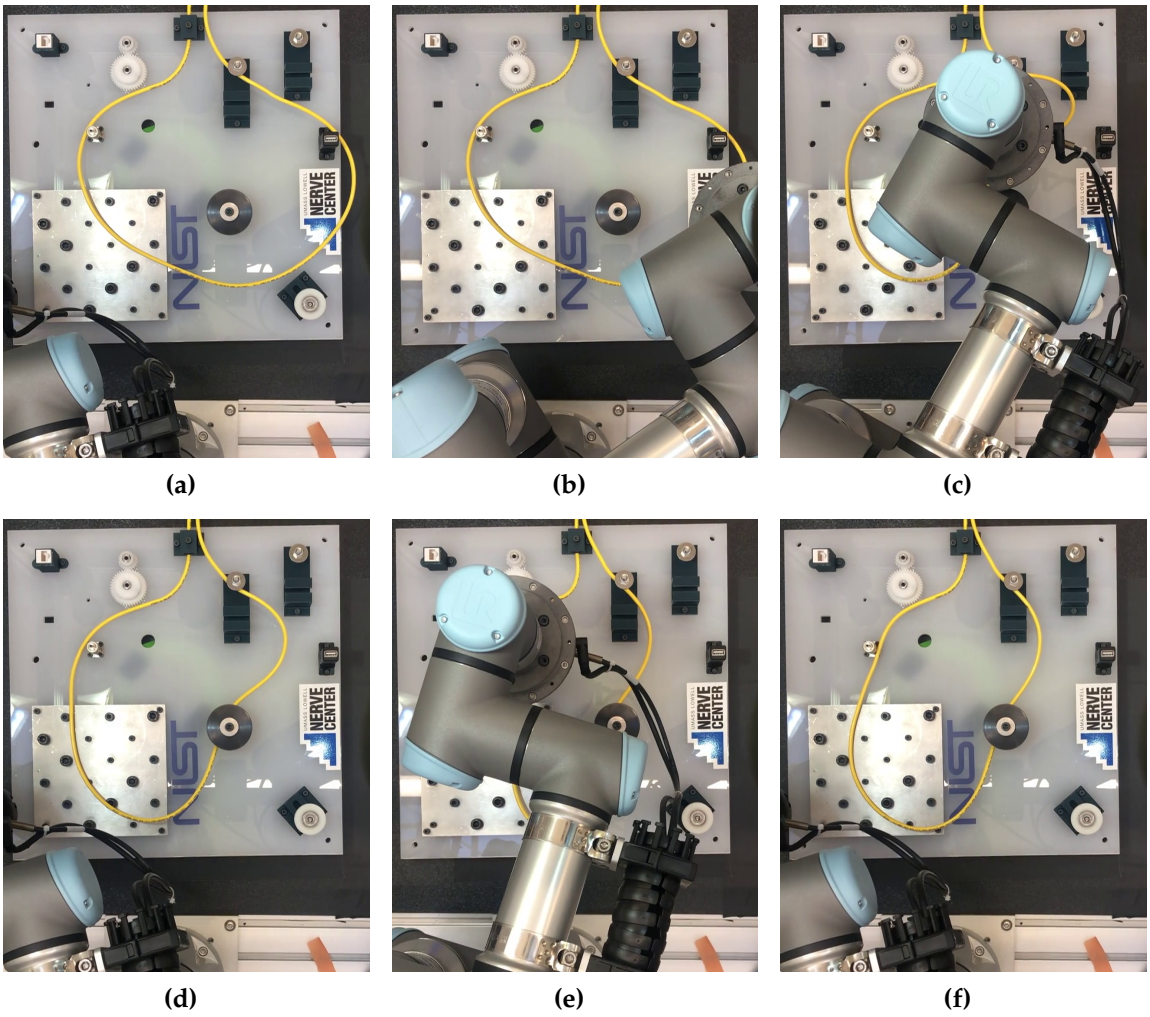
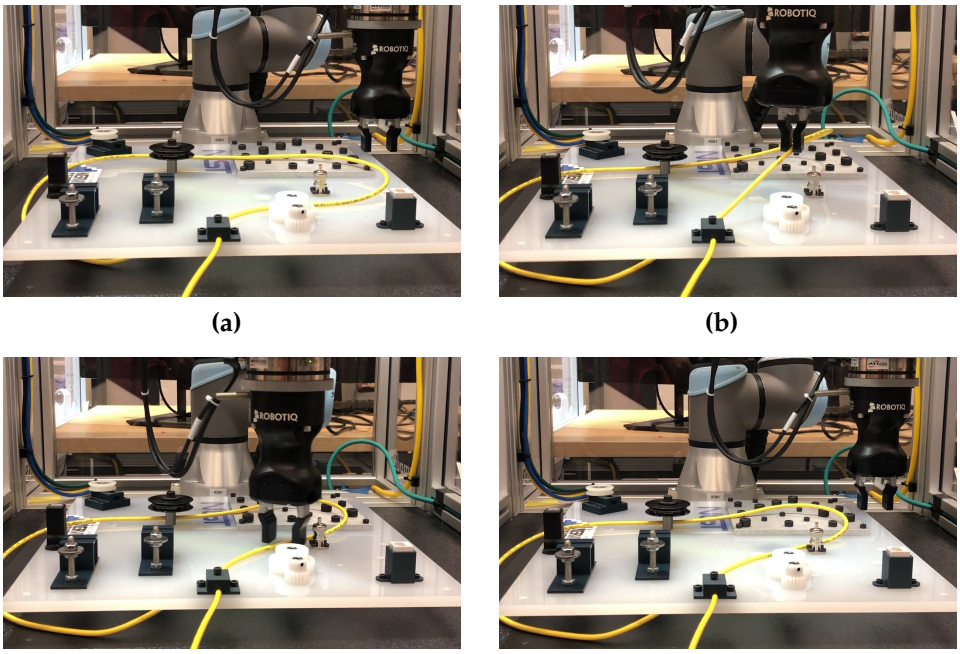



**Figure 7.7:** Routing and manipulation of a deformable one-dimensional object from the initial configuration shown in figure (a) to the goal configuration shown in figure (f). (a) Initial configuration. (b-c) Manipulating the cable based on the first iteration of routing. (d) The result configuration after the first manipulative task. (e) Manipulating the cable based on the second iteration of routing.

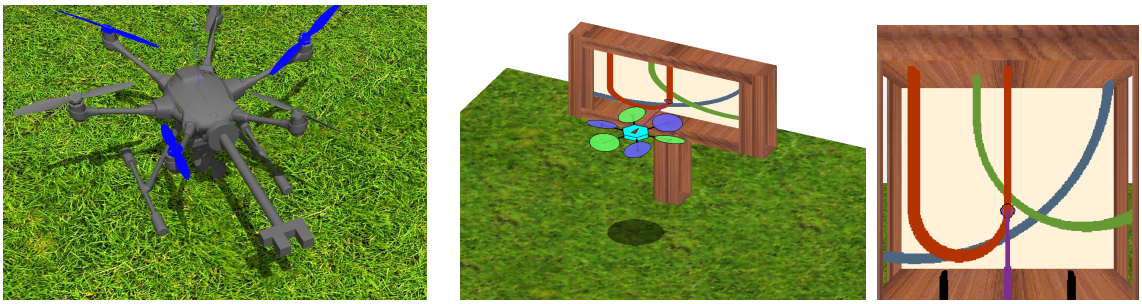
tilted arms (see Section 3.7.1) controlled with the controller system developed in Chapter 3. We measured the position error for grasping a specific point on the cable. Figure 7.9 shows our setup for testing the feasibility of the task.

For each experiment, the robot first flies to around 0.5 $[m]$ distance from the cable, then moves forward to grasp the cable segment. We measure the distance of the end-effector from the desired point on the cable at the moment



(a) (b) (c) (d)

**Figure 7.8:** Routing and manipulation of a deformable one-dimensional object. (a - d) Screenshots from a single iteration of routing and manipulation sequence for a cable on the example circuit board of Figure 7.2(a).



(a) (b)

**Figure 7.9:** The setup used for feasibility tests of DOO manipulation using aerial robots in simulation. (a) Gazebo model of our robot with a gripper for cable manipulation. (b) A screenshot of the MATLAB simulator used for analyzing the feasibility of aerial DOO manipulation and the accuracy of the end-effector for grasping and manipulating the cables. (Left) An external view of grasping the red wire using our hexarotor with tilted arms. (Right) First-person view of the moment of grasping the red wire.

when it is the closest to it.

The results from the MATLAB simulation were unrealistically perfect, and

since the Gazebo simulator tends to give more realistic results, we only report the Gazebo experiments.

Figure 7.10 illustrates how our end-effector's position can reach the target cable point. Table 7.1 shows the viability of the physical interaction with the perceived DOOs in the simulation if at least 13 $[mm]$ position error in grasping can be tolerated in the application. The next future step would be to perform the analysis on the real robot.

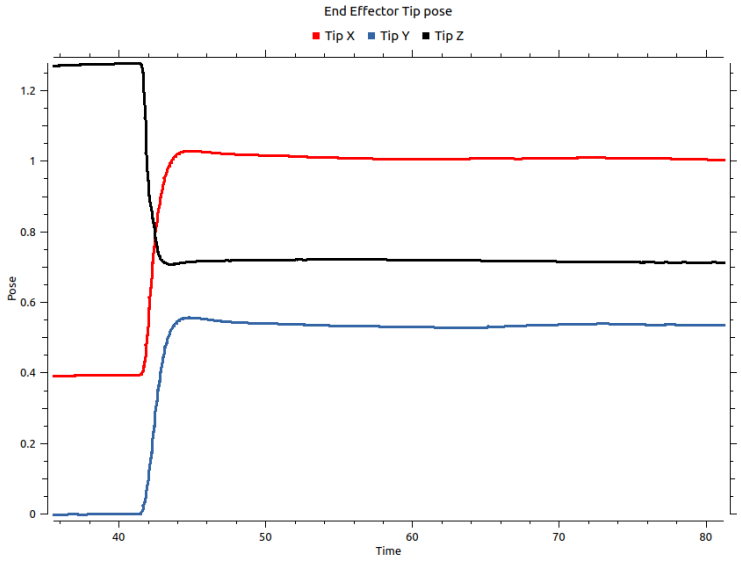



**Figure 7.10:** Feasibility tests of aerial DOO manipulation in Gazebo: End-effector's position to grasp the desired cable segment at $\begin{bmatrix}1.0 & 0.5 & 0.7\end{bmatrix}^\top$.

Table 7.1: The end-effector's position error (in $[mm]$) for grasping a cable segment. Trials done in the Gazebo simulator.

| # of Tests | Max. Error | Mean Error | Std. Dev. |
|---|---|---|---|
| 20 | 12.92 | 7.84 | 2.91 |

At the same time, aerial robots have limited wrenches compared to ground robots. It is imperative to analyze the feasibility of the physical interaction

tasks from the manipulability perspective as well.

We measured the forces required for simple cable-related tasks, such as plugging and unplugging cables in the slots on a board. Figure 7.11 shows the example forces measured for unplugging a USB Type-A cable. In this specific experiment, the maximum measured required force is 15.84 $[N]$ at the peak.

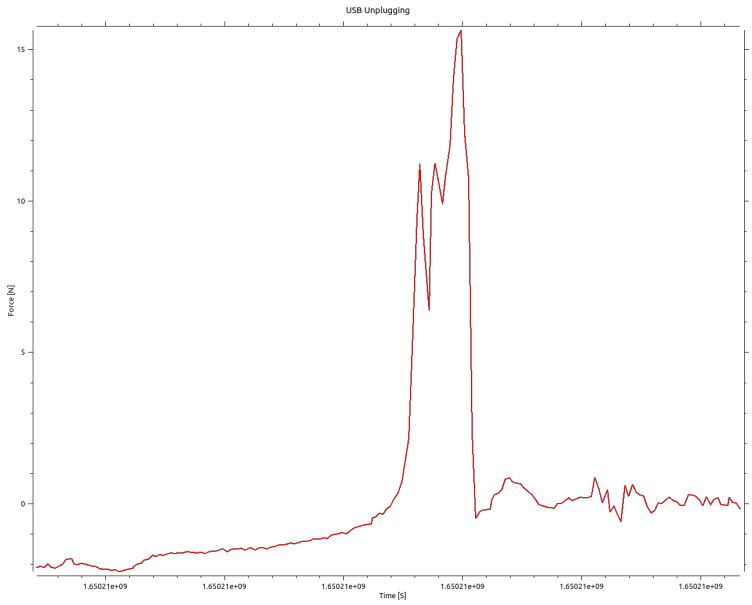
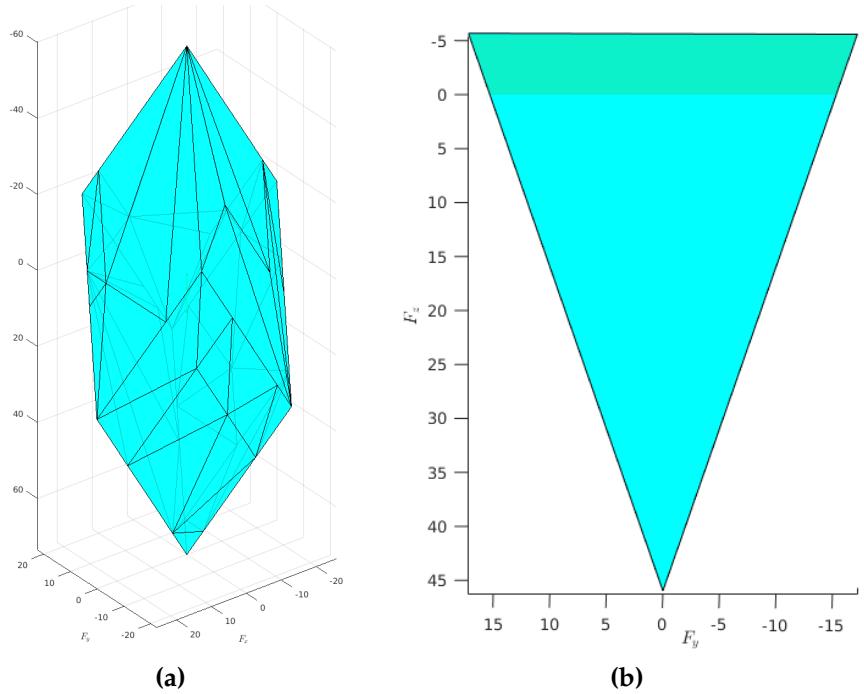



**Figure 7.11:** Feasibility tests of aerial DOO manipulation in Gazebo: Forces measured for the task of unplugging a USB Type-A cable.

Figure 7.12 compares the available thrust set for our UAV during hovering (Figure 7.12(a)) vs. when it is pulling the aforementioned USB Type-A cable directly in its backward direction (i.e., $-\hat{X}_{\mathcal{B}}$ direction) at the peak moment when the required force to unplug the cable is 15.84 $[N]$ (Figure 7.12(b)). The sets are computed using Algorithms 1 and 4 proposed in Chapter 4.

The green region in Figure 7.12(b) shows the remaining forces that still allow the robot to keep its altitude during the maximum force required for the unplugging task. This analysis shows that our aerial robot would be able to unplug the cable in this case, but it is very close to its limits. For example, if the goal is to perform the task while keeping the altitude, it may not be able




(a) (b)

**Figure 7.12:** Feasibility tests of DOO manipulation forces. (a) Our aerial platform. (b) Force polytopes for our platform. (c) Remaining $y$ and $z$ forces when unplugging a USB cable.

to achieve a more demanding task.

## 7.7 Conclusion and Discussion

We presented a novel method for the spatial representation and routing of a deformable one-dimensional object that is efficient and fast. The proposed method decomposes the work region into convex polygons and polytopes. Then it uses the decomposition to encode the configuration of a DOO in this work region. The resulting configuration is a sequence of the regions the DOO passes from, effectively simplifying the routing algorithm to a modified sequence matching method with a quadratic time and space complexity. The iteration of our routing algorithm with manipulative actions can accomplish a desired routing and manipulation task. The low planning time and overhead make it ideal for offline and online planning problems for routing and manipulation.

Our experiments showed that the proposed method could correctly plan the manipulation actions and achieve goal configurations of the DOO from various initial configurations.

The proposed approach is still in its infancy and can be extended further to cover many real-world tasks that are currently being addressed using slower and less efficient methods such as sampling-based planners. In its current iteration, this method can be used in tasks where the environment can be divided to separate convex regions and for a manipulator with two primitive actions: pick a point on the DOO, and place it in a specific point. The algorithm can be easily extended to include more manipulator motion primitives, such as wrapping the DOO around a component or passing through loops to create knots.

The proposed routing and manipulation algorithm ignores the dynamics of a cable. Although the routing algorithm itself can work well independent from the dynamics, in our real-world tests, we realized that considering the dynamics during manipulative tasks can help the system with performing the pick/place tasks. Additionally, further incorporating simple dynamics into the routing method's cost calculation in the future can reduce the number of actions (i.e., iterations) required for performing a routing/manipulation task by more accurately predicting the result of the manipulation task at each iteration.

We further analyzed the feasibility of DOO manipulation tasks using aerial robots. The analysis demonstrated our achieved precision and the required wrenches for a sample wire manipulation task. It provides a good stepping stone to having a more comprehensive and powerful study to identify the challenges and achieve helpful insights on deformable object manipulation using aerial robots.

Chapter 8

# Conclusion and Future Work

This work aimed to improve the state-of-the-art in physical interaction and manipulation of the environment using aerial robots and further extend such interactions to deformable objects.

We introduced a novel controller design that can extend most existing designs to provide faster integration of the new fully-actuated multirotors into existing flight stacks and allow them to work with commonly-available software and hardware tools without any modification.

We further extended the controller design to control both the positions and forces applied to the contact point during physical interactions. In addition to extensive simulation tests in different simulators, we showed the viability of our design for real-world free-flight and physical interaction tasks, such as contact inspection and drawing on the whiteboard, using our fully-actuated multirotor.

A possible next step for taking this Hybrid Position-Force controller further would be to extend it into a full Hybrid Motion-Wrench controller (HMWC), allowing full control of both translational and rotational motions and wrenches at the point of contact. A new Moment Controller module can be added to control the moment during the contact, similar to the Force Controller module. Figure 8.1 illustrates a possible architecture for the hybrid motion-wrench

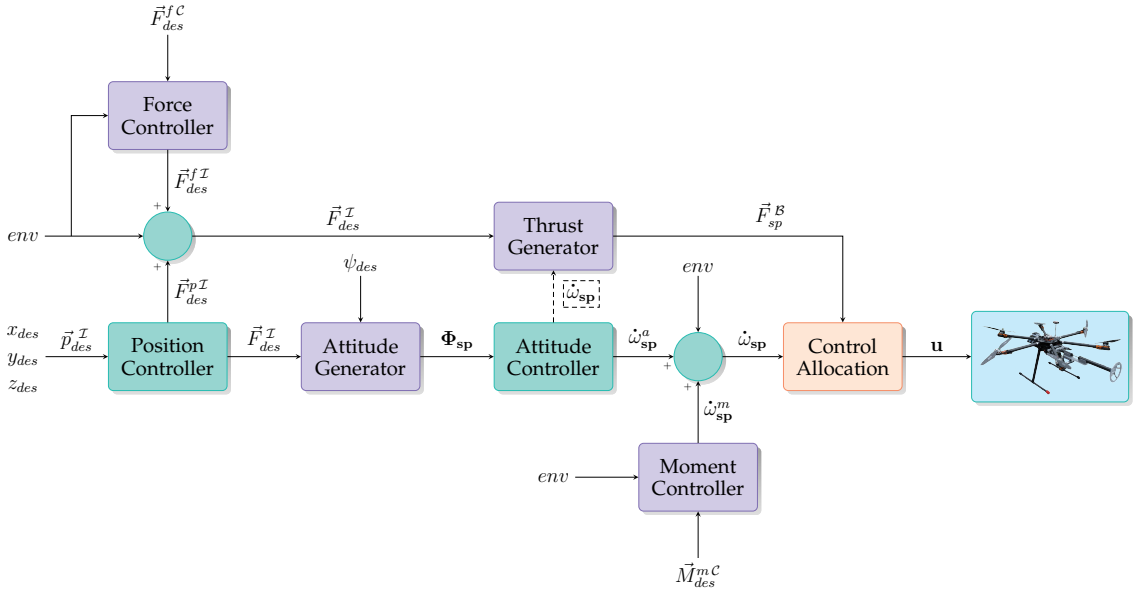



**Figure 8.1:** A possible extension of our proposed controller architecture of Chapter 3 into a Hybrid Motion-Wrench controller. The position, attitude, force, and moment control modules independently calculate the necessary linear and angular accelerations to achieve the desired inputs, then are combined based on their respective subspaces. All the modules also receive the state feedback $\mathbf{x}$ from the multirotor, which is omitted here for better illustration.

controller architecture.

This thesis provided a real-time method for estimating the dynamic manipulability polytopes (i.e., complete wrench set) for multirotors. It further extended the estimation to controlled physical interaction scenarios, allowing accurate estimation of the wrenches when some of the desired wrenches are already known. Extensive analysis and experiments illustrated how the methods work and can be used in different tasks.

We also showed how a rough estimate of the wrench set could be computed for the variable-pitch multirotors by sampling different pitches. The next step to improve the wrench-set estimation methods for aerial robots would be to solve the problem for variable-pitch rotors analytically instead of sampling. This can result in a real-time solution that can expand the benefits of the wrench estimation methods to a new group of aerial manipulators. Additionally, the

proposed methods can be extended to arm manipulators to provide the real-time estimation of the wrench polytopes for an even more extensive set of robots.

We illustrated the benefits of the real-time wrench set estimation methods in planning the physical interaction tasks, enhancing the control allocation module making it more flexible and more accurate, and computing the optimal tilt and thrust setpoint in the presence of external forces. We also enumerated other applications such as failure recovery and optimization of multirotor designs.

We mostly only explored the mentioned applications, and the future possible research directions would be developing and illustrating these applications on real UAVs in real-world tasks.

We presented a novel method for detecting deformable one-dimensional objects (e.g., wires and cables) for robotics applications and illustrated its effectiveness using real experiments.

A possible future enhancement of the method would be exploring choices other than the weighted sum for the total cost function. A more optimized implementation would take advantage of special data structures and parallelization to increase the method's speed by several orders of magnitude. Furthermore, while the considerations for the 3-D case are provided for each step, we did not implement the 3-D case, and there may be unpredicted implementation challenges. In the future, the ideas of the method can be integrated with tracking methods to improve tracking accuracy. Finally, the method's handling of the cable crossings does not detect which cable was on top. Another future step would be detecting the order of the cables in the crossings.

We presented a novel method for the spatial representation and routing of a deformable one-dimensional object that is efficient and fast. The low planning time and overhead make it ideal for offline and online planning problems for routing and manipulation. Our experiments showed its effectiveness for environments such as a wire board.

The proposed routing and manipulation algorithm ignores the dynamics of a cable. Although the routing algorithm itself can work well independent

from the dynamics, in our real-world tests, we realized that considering the dynamics during manipulative tasks can help the system with performing the pick/place tasks. Additionally, further incorporating simple dynamics into the routing method's cost calculation in the future can reduce the number of actions (i.e., iterations) required for performing a routing/manipulation task by more accurately predicting the result of the manipulation task at each iteration.

Finally, we presented the initial feasibility analysis for manipulating the deformable one-dimensional objects using aerial robots. We measured the accuracy of the end-effector in simulation and measured the real forces required for some everyday wire manipulation tasks.

With the steps we took in this work and the future advances, we believe one day it will be possible to have aerial manipulation applications involving deformable objects (such as Figure 8.2) that can help bring operational and maintenance costs while reducing the risks associated with these tasks.

**Figure 8.2:** Illustration of a possible future aerial manipulation task performed on deformable objects.

The next step would be integrating the different parts of this thesis into a real aerial manipulator to perform fully-automated cable manipulation in

realistic scenarios. However, to achieve this goal, there is a need to study the aerial manipulation of deformable objects in more realistic scenarios. Some challenges include understanding the effects of gravity and the applied forces on the manipulation task and analyzing the optimal grasping strategy of these objects in 3-D space. The planner and controller for the tasks involving deformable objects should consider the dynamics of the objects and account for the precision of the end-effector and the required wrenches. While grasping any part of an object may work for a rigid object, the physical interaction with a deformable object should be carefully planned to contact the right spot on the object with correct poses and wrenches.

Appendix A

# Symbols and Notation

The symbols in the document that have special meanings are listed in Table A.1.

| Symbol(s) | Meaning |
|---|---|
| $\mathcal{F}^{\mathcal{I}}, \mathcal{F}^{\mathcal{B}}, \mathcal{F}^{\mathcal{R}_i}$ | Inertial, body-fixed and $i^{\text{th}}$ rotor coordinate frames |
| $O_{\mathcal{I}}, O_{\mathcal{B}}, O_{\mathcal{R}_i}$ | Inertial, body-fixed and $i^{\text{th}}$ rotor frame origins |
| ${}_{\mathcal{B}}O_{\mathcal{R}_i}$ | Origin of $i^{\text{th}}$ rotor described in body-fixed frame |
| $\hat{X}_{\mathcal{I}}, \hat{Y}_{\mathcal{I}}, \hat{Z}_{\mathcal{I}}$ | Axes of the inertial frame |
| $\hat{X}_{\mathcal{B}}, \hat{Y}_{\mathcal{B}}, \hat{Z}_{\mathcal{B}}$ | Axes of the body-fixed frame |
| $\hat{X}_{\mathcal{R}_i}, \hat{Y}_{\mathcal{R}_i}, \hat{Z}_{\mathcal{R}_i}$ | Axes of the $i^{\text{th}}$ rotor frame |
| $\mathbf{R}_{\mathcal{BI}}, \mathbf{R}_{\mathcal{IB}}$ | Rotation from inertial frame to body-fixed frame and vice-versa |
| $\mathbf{R}_{\mathcal{BR}_i}, \mathbf{R}_{\mathcal{R}_i\mathcal{B}}$ | Rotation from body-fixed frame to $i^{\text{th}}$ rotor frame and vice-versa |
| $\vec{p}^{\mathcal{I}}, \vec{q}$ | The position and attitude of the robot in the inertial frame |
| $x, y, z$ | The robot position elements in the inertial frame |
| $\mathbf{\Phi}$ | The set of Euler angles of the robot |
| $\phi, \theta, \psi$ | Roll, pitch and yaw of the robot |
| $\omega$ | Body angular velocity |

| | |
|---|---|
| $p, q, r$ | Elements of the angular velocity |
| $\vec{r}_i$ | The vector from the robot's origin to the origin of the $i^{\text{th}}$ rotor |
| $r_{i_x}, r_{i_y}, r_{i_z}$ | The $i^{\text{th}}$ rotor position elements in the body-fixed frame |
| $\alpha_i$ | The angle between $\vec{r}_i$ and $\vec{r}_{i+1}$ projections on $\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$ plane |
| $\phi_{dih_i}$ | The angle between $\vec{r}_i$ and its projection on $\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$ plane |
| $\mu_i$ | The rotation angle of $\vec{r}_i$ projection on $\hat{X}_{\mathcal{B}}\hat{Y}_{\mathcal{B}}$ plane |
| $\phi_{xi}, \phi_{yi}$ | Inward and sideward angles of $i^{\text{th}}$ rotor frame |
| $\ell_i$ | The length of the $i^{\text{th}}$ rotor arm |
| $n_r$ | The number of rotors in the multirotor |
| $m, m_{rotor_i}, m_{leg_i}$ | The total mass of the multirotor, $i^{\text{th}}$ rotor and $i^{\text{th}}$ rotor arm |
| $\mathbf{I}_{\mathcal{B}}$ | The body-frame inertia tensor of the multirotor |
| $g$ | Gravitational acceleration |
| $c_{Fi}, c_{\tau i}$ | Thrust and moment constants of the $i^{\text{th}}$ rotor |
| $\Omega_i$ | Rotational (angular) velocity of the $i^{\text{th}}$ rotor |
| $d_i$ | Spinning direction of the $i^{\text{th}}$ rotor |
| $\overrightarrow{F}, \overrightarrow{M}$ | Total force and moment applied to robot |
| $\overrightarrow{F}_{grav}, \overrightarrow{F}_{thr}$ | Gravitational and the total rotor thrust forces applied to robot |
| $\overrightarrow{M}_{grav}, \overrightarrow{M}_{thr}$ | Robot moments resulted from weight distribution and rotor thrusts |
| $\overrightarrow{M}_{reac}$ | Reaction moment as a result of the rotors spinning |
| $\overrightarrow{F}_{app}, \overrightarrow{M}_{app}$ | External force and moment applied by the end-effector |
| $\overrightarrow{F}_{des}, \overrightarrow{M}_{des}$ | Desired force and moment applied by the end-effector |
| $\mathbf{x}$ | The state of the system |
| $\mathbf{u}$ | Vector of squared rotor angular velocities (robot system input) |
| $\mathbf{u}'$ | The inputs to the extended robot/end-effector system |
| $\mathbf{y}$ | The system output vector |
| $\mathbf{y_d}, \mathbf{y_m}$ | The set of the desired and the measured variables of the system |

| | |
|---|---|
| $\overrightarrow{p_d}, \overrightarrow{q_d}$ | The desired end-effector position and attitude |
| $\overrightarrow{p_e}, \overrightarrow{q_e}$ | The position and attitude of the end-effector |
| $\mathbf{f}(\mathbf{x})$ | System state drift due to gravity and rotational inertia |
| $\mathbf{J}(\mathbf{x})$ | Decoupling matrix mapping the input $\mathbf{u}$ to the state space |
| $\eta(\mathbf{\Phi})$ | Matrix mapping angular velocity to Euler angular rates |
| $\mathbf{L}$ | Matrix mapping input $\mathbf{u}$ to total thrust force $\overrightarrow{F}^{\mathcal{B}}_{thr}$ |
| $\mathbf{G}$ | Matrix mapping input $\mathbf{u}$ to the reaction moment $\overrightarrow{M}^{\mathcal{B}}_{reac}$ |
| $\mathbf{F}$ | Matrix mapping input $\mathbf{u}$ to the thrust moment $\overrightarrow{M}^{\mathcal{B}}_{thr}$ |
| $\mathbf{M}$ | The sum of $\mathbf{F}$ and $\mathbf{G}$ matrices |

Table A.1: List of the symbols

A summary of notation used in this document is as shown in Table A.2.

| Symbol | Quantity | Comments |
|---|---|---|
| $\mathbb{R}^{n\times m}$ | Real space of dimension $n \times m$ | |
| $SE(3)$ | Special Euclidean group | |
| $SO(3)$ | Special orthogonal group | |
| $s, S$ | scalar | |
| $\vec{v}, \overrightarrow{V}$ | Geometric vector | can also be shown as matrix $\mathbf{v}$ and $\mathbf{V}$ |
| $\dot{\vec{v}}, \ddot{\vec{v}}$ | First and second time derivatives of vector $\vec{v}$ | |
| ${}_xV$, ${}_yV$, ${}_zV$ | The $x$, $y$ and $z$ components of vector $\overrightarrow{V}$ | |
| $\mathbf{m}, \mathbf{M}$ | Matrix | |
| $\mathbf{m}^\top, \mathbf{M}^\top$ | Transpose of matrices $\mathbf{m}$ and $\mathbf{M}$ | |
| $[\mathbf{M}]_{ij}$ | Matrix element at row $i$ and column $j$ | |
| $\mathcal{F}$ | Reference frame | |
| $\hat{X}$ | Axis or unit vector | |
| $\left[\hat{X}\right]_i$ | $i^{\text{th}}$ element of the unit vector | |
| $\vec{v}^{\mathcal{F}}$ | Vector $\vec{v}$ described in frame $\mathcal{F}$ | |
| $[U]$ | Unit of measurement | |
| $\mathrm{s}(\cdot), \mathrm{c}(\cdot)$ | Abbreviations for $\sin(\cdot)$ and $\cos(\cdot)$ | |

Table A.2: List of the notations

# Bibliography


[1] Pramod Abichandani, Deepan Lobo, Gabriel Ford, Donald Bucci, and Moshe Kam. Wind measurement and simulation techniques in multi-rotor small unmanned aerial vehicles. *IEEE Access*, 8:54910–54927, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2977693. URL https://ieeexplore.ieee.org/document/9020170/. 5.3

[2] Albert Albers, Simon Trautmann, Thomas Howard, Trong Anh Nguyen, Markus Frietsch, and Christian Sauter. Semi-autonomous flying robot for physical interaction with environment. In *2010 IEEE Conference on Robotics, Automation and Mechatronics*, pages 441–446. IEEE, jun 2010. ISBN 978-1-4244-6503-3. doi: 10.1109/RAMECH.2010.5513152. URL http://ieeexplore.ieee.org/document/5513152/. 3.1

[3] Mike Allenspach, Karen Bodie, Maximilian Brunner, Luca Rinsoz, Zachary Taylor, Mina Kamel, Roland Siegwart, and Juan Nieto. Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight. *The International Journal of Robotics Research*, 39(10-11):1305–1325, sep 2020. ISSN 0278-3649. doi: 10.1177/0278364920943654. URL https://doi.org/10.1177/0278364920943654. 3.1

[4] Jorge M Arizaga, Herman Castaneda, and Pedro Castillo. Adaptive control for a tilted-motors hexacopter UAS flying on a perturbed environment. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 171–177, Atlanta, GA, USA, jun 2019. IEEE. ISBN 978-1-7281-0333-4. doi: 10.1109/ICUAS.2019.8798048. URL https://ieeexplore.ieee.org/document/8798048/. 2.7

[5] Veronica Arriola-Rios, Puren Guler, Fanny Ficuciello, Danica Kragic, Bruno Siciliano, and Jeremy Wyatt. Modeling of deformable objects for robotic manipulation: A tutorial and review. *Frontiers in Robotics and AI*, 7:82,

2020. ISSN 2296-9144. doi: 10.3389/frobt.2020.00082. 6.1, 6.2, 7.1

[6] Andrew Ashley, Azarakhsh Keipour, and Sebastian Scherer. Trajectory planning for a UAV wrench task considering vehicle dynamics and force output capabilities. *Carnegie Mellon Robotics Institute Summer Scholars Working Papers Journal*, 9(1):42–45, 2021. URL https://riss.ri.cmu.edu/research_showcase/working-papers-journals/. 8, 5.4

[7] ATIGamma. Ati gamma f/t sensor, 2022. URL https://www.ati-ia.com/products/ft/ft_models.aspx?id=gamma. 3.7.1, 3.19

[8] David Avis and Komei Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete & Computational Geometry*, 8(3):295–313, 1992. doi: 10.1007/BF02293050. URL https://doi.org/10.1007/BF02293050. 4.1

[9] Khelifa Baizid, Gerardo Giglio, Francesco Pierri, Miguel Ángel Trujillo Soto, Gianluca Antonelli, Fabrizio Caccavale, Antidio Viguria Jimenez, Stefano Chiaverini, and Aníbal Ollero. Experiments on behavioral coordinated control of an unmanned aerial vehicle manipulator system. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4680–4685. IEEE, may 2015. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139848. URL http://ieeexplore.ieee.org/document/7139848/. 3.1

[10] Khelifa Baizid, Gerardo Giglio, Francesco Pierri, Miguel Ángel Trujillo Soto, Gianluca Antonelli, Fabrizio Caccavale, Antidio Viguria Jimenez, Stefano Chiaverini, and Aníbal Ollero. Behavioral control of unmanned aerial vehicle manipulator systems. *Autonomous Robots*, 41(5):1203–1220, jun 2017. ISSN 0929-5593. doi: 10.1007/s10514-016-9590-0. URL https://doi.org/10.1007/s10514-016-9590-0. 3.1

[11] Chanderjit L. Bajaj and Tamal K. Dey. Convex decomposition of polyhedra and robustness. *SIAM Journal on Computing*, 21(2):339–364, 1992. doi: 10.1137/0221025. 7.3

[12] Teun Bartelds, Alex Capra, Salua Hamaza, Stefano Stramigioli, and Matteo Fumagalli. Compliant aerial manipulators: Toward a new generation of aerial robotic workers. *IEEE Robotics and Automation Letters*, 1(1):477–483, jan 2016. ISSN 2377-3766. doi: 10.1109/LRA.2016.2519948. URL http://ieeexplore.ieee.org/document/7387723/. 3.1

[13] Randal W Beard. Quadrotor dynamics and control. Technical report, Brigham Young University, 2008. URL http://hdl.lib.byu.edu/1877/624. 2.4

[14] Dmitry Bershadsky, Stephen Haviland, and Eric N Johnson. The semi-coaxial multirotor. In *74th American Helicopter Society International Annual*

*Forum and Technology Display 2018: The Future of Vertical Flight*, volume 2018-May, Georgia Institute of Technology, Atlanta, GA, United States, 2018. American Helicopter Society. ISBN 15522938 (ISSN). URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85054537479{&}partnerID=40{&}md5=72f3eb11daa191217cb56c6be06c0798. 3.1

[15] José Agnelo Bezerra and Davi A. Santos. On the guidance of fully-actuated multirotor aerial vehicles under control allocation constraints using the receding-horizon strategy. *ISA Transactions*, 2021. ISSN 0019-0578. doi: 10.1016/j.isatra.2021.07.019. URL https://www.sciencedirect.com/science/article/pii/S0019057821003864. 5.2

[16] Davide Bicego. *Design and Control of Multi-Directional Thrust Multi-Rotor Aerial Vehicles with applications to Aerial Physical Interaction Tasks*. Theses, INSA de Toulouse, sep 2019. URL https://hal.laas.fr/tel-02433940. 2.7, 3.1, 3.1

[17] Davide Bicego, Jacopo Mazzetto, Ruggero Carli, Marcello Farina, and Antonio Franchi. Nonlinear model predictive control with actuator constraints for multi-rotor aerial vehicles. *ArXiv*, nov 2019. URL http://arxiv.org/abs/1911.08183. 3.1

[18] Karen Bodie, Zachary Taylor, Mina Kamel, and Roland Siegwart. Towards efficient full pose omnidirectionality with overactuated mavs. In *16th International Symposium on Experimental Robotics (ISER 2018)*, volume 2105, pages 28–36, oct 2018. ISBN 8610828378018. URL https://doi.org/10.3929/ethz-b-000319975. 3.1

[19] Karen Bodie, Maximilian Brunner, Michael Pantic, Stefan Walser, Patrick Pfändler, Ueli Angst, Roland Siegwart, and Juan Nieto. An omnidirectional aerial manipulation platform for contact-based inspection. In *Robotics: Science and Systems XV*, pages 1–9. Robotics: Science and Systems Foundation, jun 2019. ISBN 978-0-9923747-5-4. doi: 10.15607/RSS.2019.XV.019. URL http://dx.doi.org/10.15607/RSS.2019.XV.019. 3.1

[20] Karen Bodie, Maximilian Brunner, Michael Pantic, Stefan Walser, Patrick Pfändler, Ueli Angst, Roland Siegwart, and Juan Nieto. Active interaction force control for omnidirectional aerial contact-based inspection. *ArXiv*, pages 1–12, mar 2020. URL http://arxiv.org/abs/2003.09516. 3.1

[21] Rogerio Bonatti, Cherie Ho, Wenshan Wang, Sanjiban Choudhury, and Sebastian Scherer. Towards a robust aerial cinematography platform: Localizing and tracking moving targets in unstructured environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 229–236, 2019. doi: 10.1109/IROS40897.2019.8968163. 2.1

[22] Hossein Bonyan Khamseh, Farrokh Janabi-Sharifi, and Abdelkader Abdessameud. Aerial manipulation—a literature survey. *Robotics and Autonomous Systems*, 107:221–235, sep 2018. ISSN 09218890. doi: 10.1016/j.robot.2018.06.012. URL https://www.sciencedirect.com/science/article/pii/S0921889017305535. 2.1, 3.1

[23] Dario Brescianini and Raffaello D'Andrea. Design, modeling and control of an omni-directional aerial vehicle. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3261–3266. IEEE, may 2016. ISBN 978-1-4673-8026-3. doi: 10.1109/ICRA.2016.7487497. URL http://ieeexplore.ieee.org/document/7487497/. 3.1, 5.2

[24] Dario Brescianini and Raffaello D'Andrea. An omni-directional multirotor vehicle. *Mechatronics*, 55:76–93, nov 2018. ISSN 09574158. doi: 10.1016/j.mechatronics.2018.08.005. URL https://www.sciencedirect.com/science/article/pii/S0957415818301314. 2.2, 3.1

[25] Dario Brescianini and Raffaello D'Andrea. Tilt-prioritized quadrocopter attitude control. *IEEE Transactions on Control Systems Technology*, 28(2): 376–387, mar 2020. ISSN 1063-6536. doi: 10.1109/TCST.2018.2873224. URL https://ieeexplore.ieee.org/document/8556372/. 3.3

[26] Elisabetta Cataldi, Giuseppe Muscio, Miguel Ángel Trujillo Soto, Yamnia Rodríguez Esteves, Francesco Pierri, Gianluca Antonelli, Fabrizio Caccavale, Antidio Viguria Jimenez, Stefano Chiaverini, and Aníbal Ollero. Impedance control of an aerial-manipulator: Preliminary results. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3848–3853. IEEE, oct 2016. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7759566. URL http://ieeexplore.ieee.org/document/7759566/. 3.1

[27] Zhenrong Jeremy Chen, Karl A Stol, and Peter J Richards. Preliminary design of multirotor UAVs with tilted-rotors for improved disturbance rejection capability. *Aerospace Science and Technology*, 92:635–643, sep 2019. ISSN 1270-9638. doi: 10.1016/J.AST.2019.06.038. URL https://www.sciencedirect.com/science/article/pii/S1270963818312367. 3.1

[28] P. Chiacchio, Y. Bouffard-Vercelli, and F. Pierrot. Evaluation of force capabilities for redundant manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3520–3525 vol.4, 1996. doi: 10.1109/ROBOT.1996.509249. 4.1

[29] Pasquale Chiacchio. Exploiting redundancy in minimum-time path following robot control. In *1990 American Control Conference*, pages 2313–2318, 1990. doi: 10.23919/ACC.1990.4791142. 5.4

[30] Pasquale Chiacchio, Yann Bouffard-Vercelli, and François Pierrot. Force polytope and force ellipsoid for redundant manipulators. *Journal of Robotic Systems*, 14(8):613–620, 1997. URL https://doi.org/10.1002/(SICI)1097-4563(199708)14:8<613::AID-ROB3>3.0.CO;2-P. 4.1

[31] S. Chiu. Control of redundant manipulators for task compatibility. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 1718–1724, 1987. doi: 10.1109/ROBOT.1987.1087795. 5.4

[32] Stephen L. Chiu. Task compatibility of manipulator postures. *The International Journal of Robotics Research*, 7(5):13–21, 1988. doi: 10.1177/027836498800700502. URL https://doi.org/10.1177/027836498800700502. 5.4

[33] Bryan Convens, Kelly Merckaert, Marco M Nicotra, Roberto Naldi, and Emanuele Garone. Control of fully actuated unmanned aerial vehicles with actuator saturation. *IFAC-PapersOnLine*, 50(1):12715–12720, jul 2017. ISSN 24058963. doi: 10.1016/j.ifacol.2017.08.1823. URL https://www.sciencedirect.com/science/article/pii/S240589631732445X. 2.7

[34] Bill Crowther, Alexander Lanzon, Martin Maya-Gonzalez, and David Langkamp. Kinematic analysis and control design for a nonplanar multirotor vehicle. *Journal of Guidance, Control, and Dynamics*, 34(4):1157–1171, jul 2011. ISSN 0731-5090. doi: 10.2514/1.51186. URL https://doi.org/10.2514/1.51186. 3.1, 5.2

[35] CyPhy. Cyphy lvl 1 drone: Reinvented for performance and control, 2018. URL https://www.kickstarter.com/projects/1719668770/cyphy-lvl-1-drone-reinvented-for-performance-and-c. 3.1, 3.1

[36] Zhiyong Dai, Jianjun Yi, Yajun Zhang, Bo Zhou, and Liang He. Fast and accurate cable detection using CNN. *Applied Intelligence*, 50(12):4688–4707, Dec 2020. ISSN 1573-7497. doi: 10.1007/s10489-020-01746-9. URL https://doi.org/10.1007/s10489-020-01746-9. 6.1

[37] Hemjyoti Das. A comparative study between a cant angle hexacopter and a conventional hexacopter. In *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pages 501–506, Kumaracoil, India, dec 2016. IEEE. ISBN 978-1-5090-5240-0. doi: 10.1109/ICCICCT.2016.7988002. URL http://ieeexplore.ieee.org/document/7988002/. 3.1

[38] Carlos Canudas de Wit, Bruno Siciliano, and Georges Bastin. Motion and force control. In Carlos Canudas de Wit, Bruno Siciliano, and Georges Bastin, editors, *Theory of Robot Control*, pages 141–175. Springer London,

London, 1996. ISBN 978-1-4471-1501-4. doi: 10.1007/978-1-4471-1501-4_4. URL https://doi.org/10.1007/978-1-4471-1501-4{_}4. 3.1

[39] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7.3

[40] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.110.5134. 2.3.2

[41] Xilun Ding, Pin Guo, Kun Xu, and Yushu Yu. A review of aerial manipulation of small-scale rotorcraft unmanned robotic systems. *Chinese Journal of Aeronautics*, 32(1):200–214, jan 2019. ISSN 10009361. doi: 10.1016/j.cja.2018.05.012. URL https://www.sciencedirect.com/science/article/pii/S1000936118301894. 3.1

[42] Davi Antônio dos Santos, Osamu Saotome, and Arben Cela. Trajectory control of multirotor helicopters with thrust vector constraints. In *21st Mediterranean Conference on Control and Automation*, pages 375–379, 2013. doi: 10.1109/MED.2013.6608749. 5.2

[43] Eric Dyer, Shahin Sirouspour, and Mohammad Jafarinasab. Energy optimal control allocation in a redundantly actuated omnidirectional UAV. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5316–5322, Montreal, QC, Canada, may 2019. IEEE. ISBN 978-1-5386-6027-0. doi: 10.1109/ICRA.2019.8793549. URL https://ieeexplore.ieee.org/document/8793549/. 5.2

[44] Shahab Ensafi, M Miremadi, Mohammad Eshghi, M Naseri, and Azarakhsh Keipour. Recognition of separate and adjoint Persian letters in less than three letter subwords using primitives. In *Iran 17th Electrical Engineering Conference*, pages 1–6, 2009. 6.3.2

[45] Matthias Faessler, Davide Falanga, and Davide Scaramuzza. Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight. *IEEE Robotics and Automation Letters*, 2(2):476–482, apr 2017. ISSN 2377-3766. doi: 10.1109/LRA.2016.2640362. URL http://ieeexplore.ieee.org/document/7784809/. 3.3

[46] Antonio Franchi. Platforms with multi-directional total thrust. In Anibal Ollero and Bruno Siciliano, editors, *Aerial Robotic Manipulation: Research, Development and Applications*, pages 53–65. Springer International Publishing, Cham, 2019. ISBN 978-3-030-12945-3. doi: 10.1007/978-3-030-12945-3_4. URL https://doi.org/10.1007/978-3-030-12945-3{_}4.

2.1, 3.1

[47] Antonio Franchi, Ruggero Carli, Davide Bicego, and Markus Ryll. Full-pose tracking control for aerial robotic systems with laterally bounded input force. *IEEE Transactions on Robotics*, 34(2):534–541, apr 2018. ISSN 1552-3098. doi: 10.1109/TRO.2017.2786734. URL http://ieeexplore.ieee.org/document/8291488/. 2.7, 3.1, 3.5, 4.1, 4.2, 5.2

[48] Alejandro F. Frangi, Wiro J. Niessen, Koen L. Vincken, and Max A. Viergever. Multiscale vessel enhancement filtering. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 130–137, 1998. ISBN 978-3-540-49563-5. 6.3.1

[49] Guido Gioioso, Markus Ryll, Domenico Prattichizzo, Heinrich H Bülthoff, and Antonio Franchi. Turning a near-hovering controlled quadrotor into a 3d force effector. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6278–6284, Hong Kong, China, may 2014. IEEE. ISBN 978-1-4799-3685-4. doi: 10.1109/ICRA.2014.6907785. URL http://ieeexplore.ieee.org/document/6907785/. 3.1

[50] Juan I Giribet, Ricardo S Sanchez-Pena, and Alejandro S Ghersin. Analysis and design of a tilted rotor hexacopter for fault tolerance. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1555–1567, aug 2016. ISSN 0018-9251. doi: 10.1109/TAES.2016.140885. URL http://ieeexplore.ieee.org/document/7738337/. 3.1

[51] Juan I Giribet, Claudio D Pose, Alejandro S Ghersin, and Ignacio Mas. Experimental validation of a fault tolerant hexacopter with tilted rotors. *International Journal of Electrical and Electronic Engineering & Telecommunications.*, 7(2):58–65, 2018. ISSN 23192518. doi: 10.18178/ijeetc.7.2.58-65. URL http://www.ijeetc.com/index.php?m=content{&}c=index{&}a=show{&}catid=180{&}id=1142. 3.1

[52] Juan I Giribet, Claudio D Pose, and Ignacio Mas. Fault tolerance analysis of a multirotor with 6dof. In *2019 4th Conference on Control and Fault Tolerant Systems (SysTol)*, pages 32–37, Casablanca, Morocco, sep 2019. IEEE. ISBN 978-1-7281-0380-8. doi: 10.1109/SYSTOL.2019.8864792. URL https://ieeexplore.ieee.org/document/8864792/. 3.1

[53] Xin-Yi Gong, Hu Su, De Xu, Zheng-Tao Zhang, Fei Shen, and Hua-Bin Yang. An overview of contour detection approaches. *International Journal of Automation and Computing*, 15(6):656–672, Dec 2018. ISSN 1751-8520. doi: 10.1007/s11633-018-1117-z. 6.3.3

[54] L. Guilamo, J. Kuffner, K. Nishiwaki, and S. Kagami. Manipulability optimization for trajectory generation. In *Proceedings 2006 IEEE International*

*Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2017–2022, 2006. doi: 10.1109/ROBOT.2006.1642001. 5.4

[55] Jiuming Guo, Jiwen Zhang, Dan Wu, Yuhang Gai, and Ken Chen. An algorithm based on bidirectional searching and geometric constrained sampling for automatic manipulation planning in aircraft cable assembly. *Journal of Manufacturing Systems*, 57:158–168, 2020. ISSN 0278-6125. doi: 10.1016/j.jmsy.2020.08.015. URL https://doi.org/10.1016/j.jmsy.2020.08.015. 7.1, 7.2, 7.5

[56] Tomasz Hachaj and Marek R. Ogiela. Segmentation and visualization of tubular structures in computed tomography angiography. In *Intelligent Information and Database Systems*, pages 495–503, 2012. ISBN 978-3-642-28493-9. 6.1, 6.3.1

[57] D. Hunter Hale, G. Michael Youngblood, and Priyesh N. Dixit. Automatically-generated convex region decomposition for real-time spatial agent navigation in virtual worlds. In *Proceedings of the Fourth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'08, page 173–178. AAAI Press, 2008. 7.3

[58] Haifeng Han, Gavin Paul, and Takamitsu Matsubara. Model-based reinforcement learning approach for deformable linear object manipulation. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 750–755, 2017. doi: 10.1109/COASE.2017.8256194. 7.1

[59] S. Hirai. *Energy-Based Modeling of Deformable Linear Objects*, pages 11–27. Springer London, London, 2000. ISBN 978-1-4471-0749-1. doi: 10.1007/978-1-4471-0749-1_3. 7.1

[60] Dino Hüllmann, Niels Paul, Harald Kohlhoff, Patrick P Neumann, and Achim J Lilienthal. Measuring rotor speed for wind vector estimation on multirotor aircraft. *Materials Today: Proceedings*, 5(13):26703–26708, 2018. ISSN 22147853. doi: 10.1016/j.matpr.2018.08.139. URL http://www.sciencedirect.com/science/article/pii/S2214785318321114. 5.3

[61] Davide Invernizzi and Marco Lovera. Geometric tracking control of a quadcopter tiltrotor UAV. *IFAC-PapersOnLine*, 50(1):11565–11570, 2017. ISSN 2405-8963. doi: 10.1016/j.ifacol.2017.08.1645. URL https://www.sciencedirect.com/science/article/pii/S2405896317322486. 20th IFAC World Congress. 5.2

[62] Noémie Jaquier, Leonel Rozo, Darwin G Caldwell, and Sylvain Calinon. Geometry-aware manipulability learning, tracking, and transfer. *The International Journal of Robotics Research*, 40(2-3):624–650, 2021. doi: 10.117

7/0278364920946815. URL https://doi.org/10.1177/0278364920946815. PMID: 33994629. 5.4, 5.4

[63] Shervin Javdani, Sameep Tandon, Jie Tang, James F. O'Brien, and Pieter Abbeel. Modeling and perception of deformable one-dimensional objects. In *2011 IEEE International Conference on Robotics and Automation*, pages 1607–1614, 2011. doi: 10.1109/ICRA.2011.5980431. 6.1

[64] Guangying Jiang. Dexterous hexrotor UAV platform. M.S. Thesis, University of Denver, 2013. URL https://digitalcommons.du.edu/etd/321/. 3.1

[65] Guangying Jiang and Richard M Voyles. Hexrotor UAV platform enabling dextrous interaction with structures-flight test. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6. IEEE, oct 2013. ISBN 978-1-4799-0880-6. doi: 10.1109/SSRR.2013.6719377. URL http://ieeexplore.ieee.org/document/6719377/. 3.1

[66] Guangying Jiang and Richard M Voyles. Hexrotor UAV platform enabling dextrous aerial mobile manipulation. In *Proceedings of the International Micro Air Vehicle Conference and Flight Competition IMAV 2013*, pages 122–131, Toulouse, France, sep 2013. URL http://www.imavs.org/papers/2013/122{_}IMAV2013{_}Proceedings.pdf. 3.1

[67] Guangying Jiang and Richard M Voyles. A nonparallel hexrotor UAV with faster response to disturbances for precision position keeping. In *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, pages 1–5. IEEE, oct 2014. ISBN 978-1-4799-4199-5. doi: 10.1109/SSRR.2014.7017669. URL http://ieeexplore.ieee.org/document/7017669/. 3.1

[68] Guangying Jiang and Richard M Voyles. Dexterous UAVs for precision low-altitude flight. In Kimon P Valavanis and George J Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, chapter 12, pages 207–237. Springer Netherlands, Dordrecht, 2015. ISBN 978-90-481-9707-1. doi: 10.1007/978-90-481-9707-1_130. URL https://doi.org/10.1007/978-90-481-9707-1{_}130. 3.1

[69] Guangying Jiang, Richard M Voyles, Kenneth Sebesta, and Helen Greiner. Mock-up of the exhaust shaft inspection by dexterous hexrotor at the doe wipp site. In *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–2, Philadelphia, PA, USA, oct 2015. IEEE. ISBN 978-1-5090-1959-5. doi: 10.1109/SSRR.2015.7443006. URL http://ieeexplore.ieee.org/document/7443006/. 3.1

[70] Guangying Jiang, Richard M Voyles, David Cappelleri, Daniel McArthur,

Shoushuai Mou, Alibek Yertay, Robert Bean, Praveen Abbaraju, and Arindam Chowdhury. Purpose-built UAVs for physical sampling of trace contamination at the portsmouth gaseous diffusion plant. In *Waste Management (WM 2017), 44th International Symposium*, pages 1–15, Phoenix, AZ, USA, 2017. URL https://inis.iaea.org/search/search.aspx?orig{_}q=RN:50046752. 3.1

[71] Guangying Jiang, Richard M Voyles, Kenneth Sebesta, and Helen Greiner. Estimation and optimization of fully-actuated multirotor platform with nonparallel actuation mechanism. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6843–6848. IEEE, sep 2017. ISBN 978-1-5386-2682-5. doi: 10.1109/IROS.2017.8206605. URL http://ieeexplore.ieee.org/document/8206605/. 3.1

[72] Guangying Jiang, Richard M Voyles, and Jae Jung Choi. Precision fully-actuated UAV for visual and physical inspection of structures for nuclear decommissioning and search and rescue. In *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–7, Philadelphia, PA, USA, aug 2018. IEEE. ISBN 978-1-5386-5572-6. doi: 10.1109/SSRR.2018.8468628. URL https://ieeexplore.ieee.org/document/8468628/. 3.1

[73] Tor A Johansen and Thor I Fossen. Control allocation—a survey. *Automatica*, 49(5):1087–1103, may 2013. ISSN 00051098. doi: 10.1016/j.automatica.2013.01.035. URL https://www.sciencedirect.com/science/article/pii/S0005109813000368. 2.7

[74] Ilknur Kabul, Russell Gayle, and Ming C. Lin. Cable route planning in complex environments using constrained sampling. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, SPM '07, page 395–402, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936660. doi: 10.1145/1236246.1236303. 7.1

[75] Mina Kamel, Michael Burri, and Roland Siegwart. Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles. *IFAC-PapersOnLine*, 50(1):3463–3469, jul 2017. ISSN 24058963. doi: 10.1016/j.ifacol.2017.08.849. URL https://www.sciencedirect.com/science/article/pii/S2405896317313083. 2.7

[76] Mina Kamel, Sebastian Verling, Omar Elkhatib, Christian Sprecher, Paula Wulkop, Zachary Taylor, Roland Siegwart, and Igor Gilitschenski. The voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle. *IEEE Robotics & Automation Magazine*, 25(4):34–44, dec 2018. ISSN 1070-9932. doi: 10.1109/MRA.2018.2866758. URL https://ieeexplore.ieee.org/document/8485627/. 3.1

[77] Mina Kamel, Sebastian Verling, Omar Elkhatib, Christian Sprecher, Paula Wulkop, Zachary Taylor, Roland Siegwart, and Igor Gilitschenski. Voliro: An omnidirectional hexacopter with tiltable rotors. *ArXiv*, pages 1–8, jan 2018. doi: 10.1109/MRA.2018.2866758. URL http://dx.doi.org/10.1109/MRA.2018.2866758. 2.3, 3.1

[78] Evan Kaufman, Kiren Caldwell, Daewon Lee, and Taeyoung Lee. Design and development of a free-floating hexrotor UAV for 6-dof maneuvers. In *2014 IEEE Aerospace Conference*, pages 1–10. IEEE, mar 2014. ISBN 978-1-4799-1622-1. doi: 10.1109/AERO.2014.6836427. URL http://ieeexplore.ieee.org/document/6836427/. 3.1, 4.1, 4.1

[79] Azarakhsh Keipour. Physical interaction and manipulation of the environment using aerial robots. Ph.D. Thesis Proposal, Dec 2020. 4.1

[80] Azarakhsh Keipour, Mohammad Eshghi, Sina Mohammadzadeh Ghadikolaei, Negin Mohammadi, and Shahab Ensafi. Omnifont Persian OCR system using primitives. *arXiv:2202.06371*, pages 1–5, 2013. doi: 10.48550/ARXIV.2202.06371. URL https://arxiv.org/abs/2202.06371. 6.3.2

[81] Azarakhsh Keipour, Mohammadreza Mousaei, and Sebastian Scherer. Automatic real-time anomaly detection for autonomous aerial vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5679–5685, Montreal, QC, Canada, Canada, may 2019. IEEE. ISBN 978-1-5386-6027-0. doi: 10.1109/ICRA.2019.8794286. URL https://ieeexplore.ieee.org/document/8794286/. 1

[82] Azarakhsh Keipour, Mohammadreza Mousaei, Andrew T Ashley, and Sebastian Scherer. Integration of fully-actuated multirotors into real-world applications. *arXiv:2011.06666*, pages 1–5, 2020. doi: 10.48550/ARXIV.2011.06666. URL https://arxiv.org/abs/2011.06666. 1.5, 2.2

[83] Azarakhsh Keipour, Mohammadreza Mousaei, and Sebastian Scherer. ALFA: A dataset for UAV fault and anomaly detection. *The International Journal of Robotics Research*, 40(2-3):515–520, 2021. doi: 10.1177/0278364920966642. URL https://journals.sagepub.com/doi/10.1177/0278364920966642. 2

[84] Azarakhsh Keipour, Guilherme A. S. Pereira, Rogerio Bonatti, Rohit Garg, Puru Rastogi, Geetesh Dubey, and Sebastian A. Scherer. Visual servoing approach for autonomous UAV landing on a moving vehicle. *arXiv:2104.01272*, 2021. doi: 10.48550/ARXIV.2104.01272. URL https://arxiv.org/abs/2104.01272. 5, 2.1

[85] Azarakhsh Keipour, Guilherme A.S. Pereira, and Sebastian Scherer. Real-

time ellipse detection for robotics applications. *IEEE Robotics and Automation Letters*, 6(4):7009–7016, 2021. doi: 10.1109/LRA.2021.3097057. URL https://ieeexplore.ieee.org/document/9484730. 4, 6.3.3

[86] Azarakhsh Keipour, Maryam Bandari, and Stefan Schaal. Efficient spatial representation and routing of deformable one-dimensional objects for manipulation. *arXiv:2202.06172*, pages 1–5, 2022. doi: 10.48550/ARXIV.2202.06172. URL https://arxiv.org/abs/2202.06172. 1.5, 6.4, 7, 7.1

[87] Azarakhsh Keipour, Maryam Bandari, and Stefan Schaal. Deformable one-dimensional object detection for routing and manipulation. *IEEE Robotics and Automation Letters*, 7(2):4329–4336, 2022. doi: 10.1109/LRA.2022.3146920. URL https://ieeexplore.ieee.org/document/9697357. 1.5, 6.1, 7.1, 7.2

[88] Azarakhsh Keipour, Mohammadreza Mousaei, Maryam Bandari, Stefan Schaal, and Sebastian Scherer. Detection and aerial manipulation of deformable linear objects. In *2nd Workshop on Representing and Manipulating Deformable Objects*, 2022. 1.5, 6.1, 7, 7.1

[89] Katsuyuki Kiso, Tatsuya Ibuki, Masahiro Yasuda, and Mitsuji Sampei. Structural optimization of hexrotors based on dynamic manipulability and the maximum translational acceleration. In *2015 IEEE Conference on Control Applications (CCA)*, pages 774–779, 2015. doi: 10.1109/CCA.2015.7320711. URL https://ieeexplore.ieee.org/document/7320711. 4.1, 5.1

[90] Kyongmo Koo, Xin Jiang, Atsushi Konno, and Masaru Uchiyama. Development of a wire harness assembly motion planner for redundant multiple manipulators. *Journal of Robotics and Mechatronics*, 23(6):907–918, 2011. doi: 10.20965/jrm.2011.p0907. 7.1

[91] Christopher Korpela, Matko Orsag, Miles Pekala, and Paul Oh. Dynamic stability of a mobile manipulating unmanned aerial vehicle. In *2013 IEEE International Conference on Robotics and Automation*, pages 4922–4927. IEEE, may 2013. ISBN 978-1-4673-5643-5. doi: 10.1109/ICRA.2013.6631280. URL http://ieeexplore.ieee.org/document/6631280/. 3.1

[92] V. I. Koshelev and D. N. Kozlov. Wire recognition in image within aerial inspection application. In *2015 4th Mediterranean Conference on Embedded Computing (MECO)*, pages 159–162, 2015. doi: 10.1109/MECO.2015.7181891. 6.1

[93] Denis Kotarski, Petar Piljek, Hrvoje Brezak, and Josip Kasać. Design of a fully actuated passively tilted multirotor UAV with decoupling control system. In *2017 8th International Conference on Mechanical and Aerospace Engineering (ICMAE)*, pages 385–390. IEEE, jul 2017. ISBN

978-1-5386-3305-2. doi: 10.1109/ICMAE.2017.8038677. URL http://ieeexplore.ieee.org/document/8038677/. 3.1

[94] Karl Krissian, Grégoire Malandain, Nicholas Ayache, Régis Vaillant, and Yves Trousset. Model-based detection of tubular structures in 3d images. *Computer Vision and Image Understanding*, 80(2):130–171, 2000. ISSN 1077-3142. 6.1, 6.3.1

[95] Yujun Lai, James Poon, Gavin Paul, Haifeng Han, and Takamitsu Matsubara. Probabilistic pose estimation of deformable linear objects. In *International Conference on Automation Science and Engineering (CASE)*, pages 471–476, 2018. 6.1

[96] David Langkamp, Gareth Roberts, Ashley Scillitoe, Alberto Llopis-Pascual, Juraj Zamecnik, Proctor Sam, Myrna Rodriguez-Frias, Martin Turner, Alexander Lanzon, and William Crowther. An engineering development of a novel hexrotor vehicle for 3d applications. In Guido de Croon and Matthijs Amelink, editors, *Proceedings of the International Micro Air Vehicle Conference and Flight Competition 2011 Summer Edition*, pages 35–42, 't Harde, the Netherlands, sep 2011. doi: 10.4233/uuid:eadf2fe7-7e5a-4cf8-88e5-6c247f5b6fa9. URL https://doi.org/10.4233/uuid:d7bdec21-938d-426b-9553-59cf834e8061. 3.1

[97] Dongjun Lee and Changsu Ha. Mechanics and control of quadrotors for tool operation. In *Volume 1: Adaptive Control; Advanced Vehicle Propulsion Systems; Aerospace Systems; Autonomous Systems; Battery Modeling; Biochemical Systems; Control Over Networks; Control Systems Design; Cooperativ*, pages 177–184, Lauderdale, Florida, USA, oct 2012. ASME. ISBN 978-0-7918-4529-5. doi: 10.1115/DSCC2012-MOVIC2012-8781. URL https://doi.org/10.1115/DSCC2012-MOVIC2012-8781. 3.1

[98] Jameson Yau Sung Lee. *Design and Control of a Fully-Actuated Hexrotor for Aerial Manipulation Applications*. Doctor of philosophy (phd) dissertation, University of Nevada, Las Vegas, 2018. URL https://digitalscholarship.unlv.edu/thesesdissertations/3279. 3.1

[99] Jameson Yau Sung Lee, Kam K Leang, and Woosoon Yim. Design and control of a fully-actuated hexrotor for aerial manipulation applications. *Journal of Mechanisms and Robotics*, 10(4):041007, apr 2018. ISSN 1942-4302. doi: 10.1115/1.4039854. URL http://mechanismsrobotics.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4039854. 3.1

[100] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. 7.4

[101] Chuanzheng Li, Chuang Xue, and Yue Bai. Experimental investigation on aerodynamics of nonplanar rotor pairs in a multi-rotor UAV. In *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 911–915, Xi'an, China, jun 2019. IEEE. ISBN 978-1-5386-9490-9. doi: 10.1109/ICIEA.2019.8834124. URL https://ieeexplore.ieee.org/document/8834124/. 3.1

[102] Xiang Li, Zerui Wang, and Yun-Hui Liu. Sequential robotic manipulation for active shape control of deformable linear objects. In *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 840–845, 2019. doi: 10.1109/RCAR47638.2019.9044123. 6.1

[103] Yingyu Li. Object detection and instance segmentation of cables. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2019. 6.5

[104] Zhaoying Li, Zhao Zhang, Hao Liu, and Liang Yang. A new path planning method based on concave polygon convex decomposition and artificial bee colony algorithm. *International Journal of Advanced Robotic Systems*, 17 (1):1729881419894787, 2020. doi: 10.1177/1729881419894787. 7.3

[105] Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polygons. *Computational Geometry*, 35(1):100–123, 2006. ISSN 0925-7721. doi: 10.1016/j.comgeo.2005.10.005. Special Issue on the 20th ACM Symposium on Computational Geometry. 7.3

[106] Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, SPM '07, page 121–131, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936660. doi: 10.1145/1236246.1236265. 7.3

[107] Dario Lunni, Angel Santamaria-Navarro, Roberto Rossi, Paolo Rocco, Luca Bascetta, and Juan Andrade-Cetto. Nonlinear model predictive control for aerial manipulation. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 87–93, Miami, FL, USA, USA, jun 2017. IEEE. ISBN 978-1-5090-4495-5. doi: 10.1109/ICUAS.2017.7991347. URL http://ieeexplore.ieee.org/document/7991347/. 3.1

[108] Naijing Lv, Jianhua Liu, Xiaoyu Ding, Jiashun Liu, Haili Lin, and Jiangtao Ma. Physically based real-time interactive assembly simulation of cable harness. *Journal of Manufacturing Systems*, 43:385–399, 2017. ISSN 0278-6125. doi: 10.1016/j.jmsy.2017.02.001. URL https://doi.org/10.1016/j.jmsy.2017.02.001. Special Issue on the 12th International Conference on Frontiers of Design and Manufacturing. 7.1

[109] Jiangtao Ma, Jianhua Liu, Xiaoyu Ding, and Naijing Lv. Motion planning for deformable linear objects under multiple constraints. *Robotica*, 38(5): 819–830, 2020. doi: 10.1017/S0263574719001103. 7.1

[110] Ratnesh Madaan, Daniel Maturana, and Sebastian Scherer. Wire detection using synthetic data and dilated convolutional networks for unmanned aerial vehicles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3487–3494, 2017. doi: 10.1109/IROS.2017.8206190. 6.1

[111] P. Maragos and R. Schafer. Morphological skeleton representation and coding of binary images. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1228–1244, 1986. doi: 10.1109/TASSP.1986.1164959. 6.4

[112] Hamza Mehmood and Eric N Johnson. A daisy-chain control design for a multirotor UAV with direct force capabilities. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–19, Reston, Virginia, jan 2017. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-450-3. doi: 10.2514/6.2017-1043. URL http://arc.aiaa.org/doi/10.2514/6.2017-1043. 2.2, 2.7, 2.7, 3.1, 3.1, 3.4.3, 3.5.1, 3.7.1, 5.2

[113] Hamza Mehmood, Takuma Nakamura, and Eric N Johnson. A maneuverability analysis of a novel hexarotor UAV concept. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 437–446. IEEE, jun 2016. ISBN 978-1-4673-9334-8. doi: 10.1109/ICUAS.2016.7502576. URL http://ieeexplore.ieee.org/document/7502576/. 3.1

[114] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240. IEEE, may 2015. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7140074. URL http://ieeexplore.ieee.org/document/7140074/. 3.3, 5.2

[115] Xiangdong Meng, Yuqing He, and Jianda Han. Survey on aerial manipulator: System, modeling, and control. *Robotica*, pages 1–30, oct 2019. ISSN 0263-5747. doi: 10.1017/S0263574719001450. URL https://www.cambridge.org/core/product/identifier/S0263574719001450/type/journal{_}article. 3.1

[116] Xiangdong Meng, Yuqing He, and Jianda Han. Hybrid force/motion control and implementation of an aerial manipulator towards sustained contact operations. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3678–3683, Macau, China, nov 2019. IEEE. ISBN 978-1-7281-4004-9. doi: 10.1109/IROS40897.2019.8967808. URL https://ieeexplore.ieee.org/document/8967808/. 3.1

[117] Abeje Y Mersha, Stefano Stramigioli, and Raffaella Carloni. Exploiting the dynamics of a robotic manipulator for control of UAVs. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1741–1746. IEEE, may 2014. ISBN 978-1-4799-3685-4. doi: 10.1109/ICRA.2014.6907086. URL http://ieeexplore.ieee.org/document/6907086/. 3.1

[118] Odyssée Merveille, Hugues Talbot, Laurent Najman, and Nicolas Passat. Tubular structure filtering by ranking orientation responses of path operators. In *Computer Vision (ECCV)*, pages 203–218, 2014. ISBN 978-3-319-10605-2. 6.3.1

[119] Shunsuke Mochida, Remma Matsuda, Tatsuya Ibuki, and Mitsuji Sampei. Development and design optimization of 2y hexarotor with robustness against rotor failure. *IFAC-PapersOnLine*, 53(2):9334–9339, 2020. ISSN 2405-8963. doi: 10.1016/j.ifacol.2020.12.2389. URL https://www.sciencedirect.com/science/article/pii/S2405896320330603. 21st IFAC World Congress. 5.1

[120] Shunsuke Mochida, Remma Matsuda, Tatsuya Ibuki, and Mitsuji Sampei. A geometric method of hoverability analysis for multirotor UAVs with upward-oriented rotors. *IEEE Transactions on Robotics*, 37(5):1765–1779, 2021. doi: 10.1109/TRO.2021.3064101. 5.1

[121] Abdullah Mohiuddin, Taha Tarek, Yahya Zweiri, and Dongming Gan. A survey of single and multi-UAV aerial manipulation. *Unmanned Systems*, 08(02):119–147, apr 2020. ISSN 2301-3850. doi: 10.1142/S2301385020500089. URL https://doi.org/10.1142/S2301385020500089. 3.1

[122] M. Moll and L.E. Kavraki. Path planning for deformable linear objects. *IEEE Transactions on Robotics*, 22(4):625–636, 2006. doi: 10.1109/TRO.2006.878933. 7.1

[123] João C. Monteiro, Fernando Lizarralde, and Liu Hsu. Optimal control allocation of quadrotor UAVs subject to actuator constraints. In *2016 American Control Conference (ACC)*, pages 500–505, 2016. doi: 10.1109/ACC.2016.7524963. 5.2

[124] Mohammadreza Mousaei, Junyi Geng, Azarakhsh Keipour, Dongwei Bai, and Sebastian Scherer. Design, modeling and control for a tilt-rotor VTOL UAV in the presence of actuator failure. in press, 2022. 6, 4.6, 4.9, 5.1, 5.1

[125] Mohammadreza Mousaei, Azarakhsh Keipour, Junyi Geng, and Sebastian Scherer. VTOL failure detection and recovery by utilizing redundancy. In *Workshop on Intelligent Aerial Robotics: From Autonomous Micro Aerial Vehicles to Sustainable Urban Air Mobility and Operations*, 2022. 7, 4.6, 4.9, 5.1, 5.1

[126] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12): 2262–2275, 2010. doi: 10.1109/TPAMI.2010.46. 6.1

[127] Gabriele Nava, Quentin Sable, Marco Tognon, Daniele Pucci, and Antonio Franchi. Direct force feedback control and online multi-task optimization for aerial manipulators. *IEEE Robotics and Automation Letters*, 5(2):331–338, apr 2020. ISSN 2377-3766. doi: 10.1109/LRA.2019.2958473. URL https://ieeexplore.ieee.org/document/8928943/. 3.1

[128] Hai-Nguyen Nguyen and Dongjun Lee. 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, Tokyo, Japan, nov 2013. IEEE. ISBN 978-1-4673-6358-7. doi: 10.1109/IROS.2013.6696849. URL http://ieeexplore.ieee.org/document/6696849/. 3.1

[129] Hai-Nguyen Nguyen, Sangyul Park, and Dongjun Lee. Aerial tool operation system using quadrotors as rotating thrust generators. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1285–1291, Hamburg, Germany, sep 2015. IEEE. ISBN 978-1-4799-9994-1. doi: 10.1109/IROS.2015.7353534. URL http://ieeexplore.ieee.org/document/7353534/. 3.1

[130] Ngo Phong Nguyen, Wonhee Kim, and Jun Moon. Super-twisting observer-based sliding mode control with fuzzy variable gains and its applications to fully-actuated hexarotors. *Journal of the Franklin Institute*, 356(8):4270–4303, may 2019. ISSN 00160032. doi: 10.1016/j.jfranklin.2019.03.005. URL https://www.sciencedirect.com/science/article/pii/S0016003219302005. 2.7

[131] Lasse Damtoft Nielsen, Inkyung Sung, and Peter Nielsen. Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges. *Sensors*, 19(19), 2019. ISSN 1424-8220. doi: 10.3390/s19194165. 7.3

[132] Alexandros Nikou, Georgios C Gavridis, and Kostas J Kyriakopoulos. Mechanical design, modelling and control of a novel aerial manipulator. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4698–4703. IEEE, may 2015. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139851. URL http://ieeexplore.ieee.org/document/7139851/. 3.1

[133] NIST 2018. Assembly performance metrics and test methods, 2018. URL https://www.nist.gov/el/intelligent-systems-division-73500/robotic-grasping-and-manipulation-assembly/assemb

ly. 7.5

[134] Jack H. Noble and Benoit M. Dawant. A new approach for tubular structure modeling and segmentation using graph-based techniques. *Medical image computing and computer-assisted intervention (MICCAI)*, 14(Pt 3): 305–312, 2011. 6.1, 6.3.1

[135] Aníbal Ollero, Juan Cortes, Angel Santamaria-Navarro, Miguel Ángel Trujillo Soto, Ribin Balachandran, Juan Andrade-Cetto, Angel Rodriguez, Guillermo Heredia, Antonio Franchi, Gianluca Antonelli, Konstantin Kondak, Alberto Sanfeliu, Antidio Viguria Jimenez, J. Ramiro Martinez-de Dios, and Francesco Pierri. The aeroarms project: Aerial robots with advanced manipulation capabilities for inspection and maintenance. *IEEE Robotics & Automation Magazine*, 25(4):12–23, dec 2018. ISSN 1070-9932. doi: 10.1109/MRA.2018.2852789. URL https://ieeexplore.ieee.org/document/8435987/. 3.1

[136] Matko Orsag, Christopher Korpela, Stjepan Bogdan, and Paul Oh. Lyapunov based model reference adaptive control for aerial manipulation. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 966–973. IEEE, may 2013. ISBN 978-1-4799-0817-2. doi: 10.1109/ICUAS.2013.6564783. URL http://ieeexplore.ieee.org/document/6564783/. 3.1

[137] Valerio Ortenzi, Rustam Stolkin, Jeffrey A Kuo, and Michael N Mistry. Hybrid motion/force control: a review. *Advanced Robotics*, 31(19-20):1102–1113, oct 2017. ISSN 0169-1864. doi: 10.1080/01691864.2017.1364168. URL https://doi.org/10.1080/01691864.2017.1364168. 3.1

[138] Nicolas Padoy and Gregory Hager. Deformable tracking of textured curvilinear objects. In *2012 23rd British Machine Vision Conference, BMVC 2012*, 2012. doi: 10.5244/C.26.5. 6.1

[139] A. Pagnano, M. Höpf, and R. Teti. A roadmap for automated power line inspection. maintenance and repair. In *Eighth Conference on Intelligent Computation in Manufacturing Engineering (CIRP)*, volume 12, pages 234–239, 2013. doi: 10.1016/j.procir.2013.09.041. URL https://www.sciencedirect.com/science/article/pii/S2212827113006823. 6.1

[140] Sangyul Park, Jongbeom Her, Juhyeok Kim, and Dongjun Lee. Design, modeling and control of omni-directional aerial robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2016-Novem, pages 1570–1575. IEEE, oct 2016. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7759254. URL http://ieeexplore.ieee.org/document/7759254/. 2.2, 3.1, 5.1

[141] Sangyul Park, Jeongseob Lee, Joonmo Ahn, Myungsin Kim, Jongbeom Her, Gi-Hun Yang, and Dongjun Lee. Odar: Aerial manipulation platform enabling omnidirectional wrench generation. *IEEE/ASME Transactions on Mechatronics*, 23(4):1907–1918, aug 2018. ISSN 1083-4435. doi: 10.1109/TMECH.2018.2848255. URL https://ieeexplore.ieee.org/document/8401328/. 3.1

[142] Olivier Pauly, Hauke Heibel, and Nassir Navab. A machine learning approach for deformable guide-wire tracking in fluoroscopic sequences. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 343–350, 2010. ISBN 978-3-642-15711-0. 6.1

[143] Patrick Pfändler, Karen Bodie, Ueli Angst, and Roland Siegwart. Flying corrosion inspection robot for corrosion monitoring of civil structures – first results. In *Fifth Conference on Smart Monitoring, Assessment and Rehabilitation of Civil Structures (SMAR 2019)*, pages 1–8, Potsdam, Germany, 2019. doi: 10.3929/ETHZ-B-000365572. URL https://www.ndt.net/search/docs.php3?id=24863. 3.1

[144] PRC68a. Telephone power & catv poles, 2022. URL https://www.prc68.com/I/TelephonePoles.shtml. 7.1

[145] Sujit Rajappa, Markus Ryll, Heinrich H Bülthoff, and Antonio Franchi. 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. IEEE, may 2015. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139759. URL http://ieeexplore.ieee.org/document/7139759/. 2.2, 2.2, 2.7, 3.1, 3.2, 3.5, 5.2

[146] Sujit Rajappa, Heinrich H Bülthoff, and Paolo Stegagno. Design and implementation of a novel architecture for physical human-UAV interaction. *The International Journal of Robotics Research*, 36(5-7):800–819, jun 2017. ISSN 0278-3649. doi: 10.1177/0278364917708038. URL https://doi.org/10.1177/0278364917708038. 2.7

[147] Matthias Rambow, Thomas Schauß, Martin Buss, and Sandra Hirche. Autonomous manipulation of deformable objects based on teleoperated demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2809–2814, 2012. doi: 10.1109/IROS.2012.6386002. 7.1

[148] Ramy Rashad, Petra Kuipers, Johan B C Engelen, and Stefano Stramigioli. Design, modeling, and geometric control on SE(3) of a fully-actuated hexarotor for aerial interaction. *ArXiv*, pages 1–9, sep 2017. URL http://arxiv.org/abs/1709.05398. 3.1, 5.1

[149] Ramy Rashad, Federico Califano, and Stefano Stramigioli. Port-hamiltonian passivity-based control on SE(3) of a fully actuated UAV for aerial physical interaction near-hovering. *IEEE Robotics and Automation Letters*, 4(4):4378–4385, oct 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2932864. URL https://ieeexplore.ieee.org/document/8786163/. 3.1

[150] Ramy Rashad, Johan B C Engelen, and Stefano Stramigioli. Energy tank-based wrench/impedance control of a fully-actuated hexarotor: A geometric port-hamiltonian approach. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6418–6424, Montreal, QC, Canada, may 2019. IEEE. ISBN 978-1-5386-6027-0. doi: 10.1109/ICRA.2019.8793939. URL https://ieeexplore.ieee.org/document/8793939/. 3.1

[151] Ramy Rashad, Jelmer Goerres, Ronald Aarts, Johan B C Engelen, and Stefano Stramigioli. Fully actuated multirotor UAVs: A literature review. *IEEE Robotics & Automation Magazine*, 27(3):97–107, sep 2020. ISSN 1070-9932. doi: 10.1109/MRA.2019.2955964. URL https://ieeexplore.ieee.org/document/8978486/. 2.1, 3.1

[152] Alireza Rastegarpanah, Rhys Howard, and Rustam Stolkin. Tracking linear deformable objects using slicing method. *Robotica*, page 1–19, 2021. doi: 10.1017/S0263574721001065. 6.1

[153] Máximo A. Roa and Raúl Suárez. Grasp quality measures: review and performance. *Autonomous Robots*, 38(1):65–88, 2015. doi: 10.1007/s10514-014-9402-3. URL https://doi.org/10.1007/s10514-014-9402-3. 4.1

[154] Hugo Romero, Sergio Salazar, Anand Sanchez, and Rogelio Lozano. A new UAV configuration having eight rotors: Dynamical model and real-time control. In *2007 46th IEEE Conference on Decision and Control*, pages 6418–6423, New Orleans, LA, USA, 2007. IEEE. ISBN 978-1-4244-1497-0. doi: 10.1109/CDC.2007.4434776. URL http://ieeexplore.ieee.org/document/4434776/. 2.2, 3.1

[155] Olivier Roussel, Andy Borum, Michel Taïx, and Timothy Bretl. Manipulation planning with contacts for an extensible elastic rod by sampling on the submanifold of static equilibrium configurations. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3116–3121, 2015. doi: 10.1109/ICRA.2015.7139627. 7.1

[156] Fabio Ruggiero, Miguel Ángel Trujillo Soto, Raul Cano, Hector Ascorbe, Antidio Viguria Jimenez, Corrêa Perez, Vincenzo Lippiello, Aníbal Ollero, and Bruno Siciliano. A multilayer control for multirotor UAVs equipped with a servo robot arm. In *2015 IEEE International Conference on Robotics*

*and Automation (ICRA)*, volume 2015-June, pages 4014–4020, Seattle, Washington, may 2015. IEEE. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139760. URL http://ieeexplore.ieee.org/document/7139760/. 3.1

[157] Fabio Ruggiero, Vincenzo Lippiello, and Aníbal Ollero. Aerial manipulation: A literature review. *IEEE Robotics and Automation Letters*, 3(3):1957–1964, jul 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2808541. URL https://ieeexplore.ieee.org/document/8299552/. 2.1, 3.1

[158] Markus Ryll, Heinrich H Bülthoff, and Paolo Robuffo Giordano. A novel overactuated quadrotor unmanned aerial vehicle: Modeling, control, and experimental validation. *IEEE Transactions on Control Systems Technology*, 23(2):540–556, mar 2015. ISSN 1063-6536. doi: 10.1109/TCST.2014.2330999. URL http://ieeexplore.ieee.org/document/6868215/. 2.3, 3.1

[159] Markus Ryll, Davide Bicego, and Antonio Franchi. 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. IEEE, oct 2016. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7759271. URL http://ieeexplore.ieee.org/document/7759271/. 2.3, 3.1

[160] Markus Ryll, Giuseppe Muscio, Francesco Pierri, Elisabetta Cataldi, Gianluca Antonelli, Fabrizio Caccavale, and Antonio Franchi. 6d physical interaction with a fully actuated aerial robot. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5190–5195. IEEE, may 2017. ISBN 978-1-5090-4633-1. doi: 10.1109/ICRA.2017.7989608. URL http://ieeexplore.ieee.org/document/7989608/. 3.1

[161] Markus Ryll, Giuseppe Muscio, Francesco Pierri, Elisabetta Cataldi, Gianluca Antonelli, Fabrizio Caccavale, Davide Bicego, and Antonio Franchi. 6d interaction control with aerial robots: The flying end-effector paradigm. *The International Journal of Robotics Research*, 38(9):1045–1062, aug 2019. ISSN 0278-3649. doi: 10.1177/0278364919856694. URL https://doi.org/10.1177/0278364919856694. 3.1

[162] Mitul Saha and Pekka Isto. Manipulation planning for deformable linear objects. *IEEE Transactions on Robotics*, 23(6):1141–1150, 2007. doi: 10.1109/TRO.2007.907486. 6.1

[163] Punam K. Saha, Gunilla Borgefors, and Gabriella Sanniti di Baja. Chapter 1 - skeletonization and its applications – a review. In *Skeletonization: Theory, Methods and Applications*, pages 3–42. Academic Press, 2017. ISBN 978-0-08-101291-8. 6.3.2

[164] S Salazar, H Romero, R Lozano, and P Castillo. Modeling and real-time stabilization of an aircraft having eight rotors. *Journal of Intelligent and Robotic Systems*, 54(1):455–470, 2009. ISSN 1573-0409. doi: 10.1007/s10846-008-9274-x. URL https://doi.org/10.1007/s10846-008-9274-x. 3.1

[165] Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. *The International Journal of Robotics Research*, 37(7):688–716, 2018. doi: 10.1177/0278364918779698. 7.1

[166] Pedro Sanchez-Cuevas, Guillermo Heredia, and Aníbal Ollero. Multirotor aerodynamic effects in aerial manipulation. In Anibal Ollero and Bruno Siciliano, editors, *Aerial Robotic Manipulation: Research, Development and Applications*, pages 67–82. Springer International Publishing, Cham, 2019. ISBN 978-3-030-12945-3. doi: 10.1007/978-3-030-12945-3_5. URL https://doi.org/10.1007/978-3-030-12945-3{_}5. 3.1

[167] Simon Schopferer, Julian Soren Lorenz, Azarakhsh Keipour, and Sebastian Scherer. Path planning for unmanned fixed-wing aircraft in uncertain wind conditions using trochoids. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 503–512, Dallas, TX, USA, jun 2018. IEEE. ISBN 978-1-5386-1354-2. doi: 10.1109/ICUAS.2018.8453391. URL https://ieeexplore.ieee.org/document/8453391/. 3, 5.3

[168] John Schulman, Alex Lee, Jonathan Ho, and Pieter Abbeel. Tracking deformable objects with point clouds. In *2013 IEEE International Conference on Robotics and Automation*, pages 1130–1137, 2013. 6.1

[169] Micha Schuster, David Bernstein, Chao Yao, Klaus Janschek, and Michael Beitelschmidt. Comparison of design approaches of fully actuated aerial robots based on maximum wrench generation and minimum energy consumption. *IFAC-PapersOnLine*, 52(15):603–608, jan 2019. ISSN 24058963. doi: 10.1016/j.ifacol.2019.11.742. URL https://www.sciencedirect.com/science/article/pii/S2405896319317379. 3.1

[170] Yu Sheng. *Adaptive Control Techniques for Multirotor Aircrafts*. Phd (doctor of philosophy) dissertation, University of Virginia, 2019. URL https://search.lib.virginia.edu/catalog/pn89d694n. 2.7

[171] Yu Sheng, Gang Tao, and Peter Beling. Dynamic mutation and adaptive tracking control of omni-directional multirotor systems. In *AIAA Scitech 2019 Forum*, AIAA SciTech Forum, pages 1–15, San Diego, CA, USA, jan 2019. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-578-4. doi: 10.2514/6.2019-1562. URL https://doi.org/10.251

4/6.2019-1562. 2.7

[172] Taku Shimizu, Satoshi Suzuki, Takashi Kawamura, Hikaru Ueno, and Hiroki Murakami. Proposal of 6dof multi-copter and verification of its controllability. In *2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 810–815. IEEE, jul 2015. ISBN 978-4-9077-6448-7. doi: 10.1109/SICE.2015.7285456. URL http://ieeexplore.ieee.org/document/7285456/. 3.1

[173] Antun Skuric, Vincent Padois, and David Daney. On-line force capability evaluation based on efficient polytope vertex search. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1700–1706, 2021. doi: 10.1109/ICRA48506.2021.9562050. 4.1

[174] Skygauge. Skygauge, the drone for any inspection, 2020. URL https://skygauge.co/. 3.1, 3.1

[175] Ewoud Smeur, Daan Höppener, and Christophe De Wagter. Prioritized control allocation for quadrotors subject to saturation. In H. de Plinval J.-M. Moschetta, G. Hattenberger, editor, *International Micro Air Vehicle Conference and Flight Competition 2017*, pages 37–43, Toulouse, France, Sep 2017. 3.3

[176] Alejandro Suarez, Victor M Vega, Manuel Fernandez, Guillermo Heredia, and Anibal Ollero. Benchmarks for aerial manipulation. *IEEE Robotics and Automation Letters*, 5(2):2650–2657, apr 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.2972870. URL https://ieeexplore.ieee.org/document/8990026/. 3.1

[177] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985. 6.4

[178] Yuichi Tadokoro, Tatsuya Ibuki, and Mitsuji Sampei. Maneuverability analysis of a fully-actuated hexrotor UAV considering tilt angles and arrangement of rotors. *IFAC-PapersOnLine*, 50(1):8981–8986, jul 2017. ISSN 24058963. doi: 10.1016/j.ifacol.2017.08.1325. URL https://www.sciencedirect.com/science/article/pii/S2405896317318505. 3.1, 4.1, 4.2

[179] Yuichi Tadokoro, Tatsuya Ibuki, and Mitsuji Sampei. Joint optimization of geometric control and structure of a fully-actuated hexrotor based on an analytic hjbe solution. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1186–1191. IEEE, dec 2018. ISBN 978-1-5386-1395-5. doi: 10.1109/CDC.2018.8618982. URL https://ieeexplore.ieee.org/document/8618982/. 3.1, 4.1, 5.1

[180] Yuichi Tadokoro, Tatsuya Ibuki, and Mitsuji Sampei. Classification and structural evaluation of fully-actuated hexrotor UAVs. In *2018 Annual American Control Conference (ACC)*, pages 1945–1950. IEEE, jun 2018. ISBN 978-1-5386-5428-6. doi: 10.23919/ACC.2018.8431700. URL https://ieeexplore.ieee.org/document/8431700/. 3.1, 4.1, 4.1, 4.2

[181] Yuichi Tadokoro, Tatsuya Ibuki, and Mitsuji Sampei. Nonlinear model predictive control of a fully-actuated UAV on SE(3) using acceleration characteristics of the structure. In *2019 12th Asian Control Conference (ASCC)*, pages 283–288, Kitakyushu-shi, Japan, 2019. ISBN 978-4-88898-300-6. URL https://ieeexplore.ieee.org/abstract/document/8765156. 2.7

[182] Te Tang, Yongxiang Fan, Hsien-Chung Lin, and Masayoshi Tomizuka. State estimation for deformable objects by point registration and dynamic simulation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2427–2433, 2017. doi: 10.1109/IROS.2017.8206058. 6.1

[183] Te Tang, Changhao Wang, and Masayoshi Tomizuka. A framework for manipulating deformable linear objects by coherent point drift. *IEEE Robotics and Automation Letters*, 3(4):3426–3433, 2018. doi: 10.1109/LRA.2018.2852770. 6.1

[184] Feng Tian, Yaping Wang, and Linlin Zhu. Power line recognition and tracking method for UAVs inspection. In *2015 IEEE International Conference on Information and Automation*, pages 2136–2141, 2015. doi: 10.1109/ICInfA.2015.7279641. 6.1

[185] Marco Tognon. *Theory and Applications for Control and Motion Planning of Aerial Robots in Physical Interaction with particular focus on Tethered Aerial Vehicles*. Theses, Institut national des sciences appliquées de Toulouse, jul 2018. URL https://hal.laas.fr/tel-02003048. 3.1

[186] Marco Tognon and Antonio Franchi. Omnidirectional aerial vehicles with unidirectional thrusters: Theory, optimal design, and control. *IEEE Robotics and Automation Letters*, 3(3):2277–2282, jul 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2802544. URL http://ieeexplore.ieee.org/document/8281444/. 3.1

[187] Marco Tognon, Burak Yüksel, Gabriele Buondonno, and Antonio Franchi. Dynamic decentralized control for protocentric aerial manipulators. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6375–6380. IEEE, may 2017. ISBN 978-1-5090-4633-1. doi: 10.1109/ICRA.2017.7989753. URL http://ieeexplore.ieee.org/document/7989753/. 3.1

[188] Marco Tognon, Hermes A Tello Chavez, Enrico Gasparin, Quentin Sable, Davide Bicego, Anthony Mallet, Marc Lany, Gilles Santi, Bernard Revaz, Juan Cortes, and Antonio Franchi. 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, apr 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2895880. URL https://ieeexplore.ieee.org/document/8629273/. 3.1

[189] Daichi Toratani. Research and development of double tetrahedron hexa-rotorcraft (dot-hr). *Proceedings of the 28th International Congress of the Aeronautical Sciences*, pages 1–8, 2012. URL http://www.icas-proceedings.net/ICAS2012/PAPERS/727.PDF. 3.1

[190] Miguel Ángel Trujillo Soto, José Martínez-de Dios, Carlos Martín, Antidio Viguria Jimenez, and Aníbal Ollero. Novel aerial manipulator for accurate and robust industrial NDT contact inspection: A new tool for the oil and gas inspection industry. *Sensors*, 19(6):1305, mar 2019. ISSN 1424-8220. doi: 10.3390/s19061305. URL https://pubmed.ncbi.nlm.nih.gov/30875905. 3.1

[191] Nikolaus Vahrenkamp, Tamim Asfour, Giorgio Metta, Giulio Sandini, and Rüdiger Dillmann. Manipulability analysis. In *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), Osaka, Japan, November 29 - Dec. 1, 2012*, pages 568–573. IEEE, 2012. doi: 10.1109/HUMANOIDS.2012.6651576. URL http://dx.doi.org/10.1109/HUMANOIDS.2012.6651576. 5.4

[192] Voliro. Voliro airborne robotics, 2020. URL https://www.voliro.com/. 3.1, 3.1

[193] Richard M Voyles and Guangying Jiang. Hexrotor UAV platform enabling dextrous interaction with structures - preliminary work. In *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–7. IEEE, nov 2012. ISBN 978-1-4799-0165-4. doi: 10.1109/SSRR.2012.6523891. URL http://ieeexplore.ieee.org/document/6523891/. 3.1

[194] Hidefumi Wakamatsu, Eiji Arai, and Shinichi Hirai. Knotting/unknotting manipulation of deformable linear objects. *The International Journal of Robotics Research*, 25(4):371–395, 2006. doi: 10.1177/0278364906064819. 6.1

[195] Angelina Wang, Thanard Kurutach, Kara Liu, Pieter Abbeel, and Aviv Tamar. Learning robotic manipulation through visual planning and acting, 2019. 7.1

[196] Bin Wang, Han Shi, Enuo Cui, Hai Zhao, Dongxiang Yang, Jian Zhu, and

Shengchang Dou. A robust and efficient framework for tubular structure segmentation in chest ct images. *Technology and Health Care*, 29:655–665, 2021. ISSN 1878-7401. 4. 6.1, 6.3.1

[197] Chenglong Wang, Yuichiro Hayashi, Masahiro Oda, Hayato Itoh, Takayuki Kitasaka, Alejandro F. Frangi, and Kensaku Mori. Tubular structure segmentation using spatial fully connected network with radial distance loss for 3d medical images. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 348–356, 2019. ISBN 978-3-030-32226-7. 6.1, 6.3.1

[198] Yan Wang, Xu Wei, Fengze Liu, Jieneng Chen, Yuyin Zhou, Wei Shen, Elliot K. Fishman, and Alan L. Yuille. Deep distance transform for tubular structure segmentation in CT scans. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3832–3841, 2020. ISSN 1063-6919. doi: 10.1109/CVPR42600.2020.00389. 6.3.1

[199] Yixuan Wang, Dale McConachie, and Dmitry Berenson. Tracking partially-occluded deformable objects while enforcing geometric constraints. In *2021 International Conference on Robotics and Automation (ICRA)*, pages 1–7, 2021. 6.1

[200] Rens Werink. *On the Control Allocation of Fully-Actuated and Over-Actuated Multirotor UAVs*. Essay (master), University of Twente, 2019. URL http://essay.utwente.nl/77102. 2.7

[201] Han W Wopereis, Jaap J Hoekstra, Tjark H Post, Gerrit A Folkertsma, Stefano Stramigioli, and Matteo Fumagalli. Application of substantial and sustained force to vertical surfaces using a quadrotor. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2704–2709. IEEE, may 2017. ISBN 978-1-5090-4633-1. doi: 10.1109/ICRA.2017.7989314. URL http://ieeexplore.ieee.org/document/7989314/. 3.1

[202] Ruonan Xu, Jianjun Luo, and Mingming Wang. Kinematic and dynamic manipulability analysis for free-floating space robots with closed chain constraints. *Robotics and Autonomous Systems*, 130:103548, 2020. ISSN 0921-8890. doi: 10.1016/j.robot.2020.103548. URL https://www.sciencedirect.com/science/article/pii/S0921889019309236. 5.4

[203] Mengyuan Yan, Yilin Zhu, Ning Jin, and Jeannette Bohg. Self-supervised learning of state estimation for manipulating deformable linear objects. *IEEE Robotics and Automation Letters*, 5(2):2372–2379, 2020. doi: 10.1109/LRA.2020.2969931. 6.1, 6.4

[204] Chao Yao, Jan Krieglstein, and Klaus Janschek. Modeling and sliding mode control of a fully-actuated multirotor with tilted propellers. *IFAC-*

*PapersOnLine*, 51(22):115–120, jan 2018. ISSN 24058963. doi: 10.1016/j.ifacol.2018.11.527. URL https://www.sciencedirect.com/science/article/pii/S2405896318332336. 2.7

[205] Chao Yao, Micha Schuster, Zijian Jiang, Klaus Janschek, and Michael Beitelschmidt. Sensitivity analysis of model-based impedance control for physically interactive hexarotor. *IFAC-PapersOnLine*, 52(15):597–602, jan 2019. ISSN 24058963. doi: 10.1016/j.ifacol.2019.11.741. URL https://www.sciencedirect.com/science/article/pii/S2405896319317355. 3.1

[206] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, 1985. doi: 10.1177/027836498500400201. URL https://doi.org/10.1177/027836498500400201. 4.1

[207] Burak Yüksel. *Design, Modeling and Control of Aerial Robots for Physical Interaction and Manipulation*. phdthesis, Universität Stuttgart, 2017. URL https://homepages.laas.fr/afranchi/robotics/sites/default/files/phd-thesis-2017-Yueksel.pdf. 3.1

[208] Burak Yüksel, Cristian Secchi, Heinrich H Bülthoff, and Antonio Franchi. Reshaping the physical properties of a quadrotor through ida-pbc and its application to aerial physical interaction. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6258–6265, Hong Kong, China, may 2014. IEEE. ISBN 978-1-4799-3685-4. doi: 10.1109/ICRA.2014.6907782. URL http://ieeexplore.ieee.org/document/6907782/. 2.1, 3.1

[209] Burak Yüksel, Cristian Secchi, Heinrich H Bülthoff, and Antonio Franchi. Aerial physical interaction via ida-pbc. *The International Journal of Robotics Research*, 38(4):403–421, apr 2019. ISSN 0278-3649. doi: 10.1177/0278364919835605. URL https://doi.org/10.1177/0278364919835605. 3.1

[210] Jingjing Zhang, Liang Liu, Binhai Wang, Xiguang Chen, Qian Wang, and Tianru Zheng. High speed automatic power line detection and tracking for a UAV-based inspection. In *2012 International Conference on Industrial Control and Electronics Engineering*, pages 266–269, 2012. doi: 10.1109/ICICEE.2012.77. 6.1

[211] Moju Zhao, Tomoki Anzai, Fan Shi, Xiangyu Chen, Kei Okada, and Masayuki Inaba. Design, modeling, and control of an aerial robot dragon: A dual-rotor-embedded multilink robot with the ability of multi-degree-of-freedom aerial transformation. *IEEE Robotics and Automation Letters*, 3(2):1176–1183, apr 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2793344. URL http://ieeexplore.ieee.org/document/8258850/. 2.3

[212] Guang Zhou, Jinwei Yuan, I-Ling Yen, and Farokh Bastani. Robust real-time UAV based power line detection and tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 744–748, 2016. doi: 10.1109/ICIP.2016.7532456. 6.1

[213] Zhi-Gang Zhou, Yong-An Zhang, and Di Zhou. Geometric modeling and control for the full-actuated aerial manipulating system. In *2016 35th Chinese Control Conference (CCC)*, pages 6178–6182, Chengdu, China, jul 2016. IEEE. ISBN 978-9-8815-6391-0. doi: 10.1109/ChiCC.2016.7554326. URL http://ieeexplore.ieee.org/document/7554326/. 3.1

[214] Alexandros Zormpas, Konstantia Moirogiorgou, Kostas Kalaitzakis, George A. Plokamakis, Panayotis Partsinevelos, George Giakos, and Michalis Zervakis. Power transmission lines inspection using properly equipped unmanned aerial vehicle (UAV). In *2018 IEEE International Conference on Imaging Systems and Techniques (IST)*, pages 1–5, 2018. doi: 10.1109/IST.2018.8577142. 6.1

[215] Andrew Zulu and Samuel John. A review of control algorithms for autonomous quadrotors. *Open Journal of Applied Sciences*, 14(4):547–556, feb 2014. doi: 10.4236/ojapps.2014.414053. URL http://dx.doi.org/10.4236/ojapps.2014.414053. 2.7