Title

Expanding the Operational Environments of UAVs: Design, Control, and Motion Planning for a Tensegrity Aerial Vehicle and an Uncrewed Underwater Aerial Vehicle

Author

Zha, Jiaming

Publication Date

2023

Peer reviewed|Thesis/dissertation

Expanding the Operational Environments of UAVs: Design, Control, and Motion Planning for a Tensegrity Aerial Vehicle and an Uncrewed Underwater Aerial Vehicle

By

Jiaming Zha

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Assistant Professor Mark Mueller, Chair
Professor Claire Tomlin
Associate Professor Koushil Sreenath

Summer 2023

Expanding the Operational Environments of UAVs: Design, Control, and Motion Planning for a Tensegrity Aerial Vehicle and an Uncrewed Underwater Aerial Vehicle

Abstract

Expanding the Operational Environments of UAVs: Design, Control, and Motion Planning for a Tensegrity Aerial Vehicle and an Uncrewed Underwater Aerial Vehicle

By

Jiaming Zha

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Assistant Professor Mark Mueller, Chair

This dissertation explores methodologies for enhancing the operational capabilities of Uncrewed Aerial Vehicles (UAVs). By tightly integrating design, control strategy, and planning algorithms, we have enabled UAVs to operate in environments that pose significant challenges for traditional flying robots.

The dissertation comprises three parts. The first part focuses on collision-resilient UAVs designed for cluttered environments with obstacles that are difficult to detect and/or avoid. We introduce a vehicle featuring an icosahedron tensegrity structure for resilience to high-speed collisions. The design of the vehicle is guided by a model-based approach, which employs dynamics simulation to predict structural stresses in the system. Furthermore, an autonomous re-orientation controller is presented to facilitate post-collision flight resumption, enabling the vehicle to rotate from any given orientation to ones ready for takeoff.

The second part presents a collision-inclusive, sampling-based motion planning algorithm for narrow and cluttered environments. Incorporating collisions into planning yields two benefits. First, collisions can be exploited for quick changes of movement directions. Second, the allowance of collisions results in more efficient sampling in confined spaces, thereby reducing the planning time needed. The algorithm's effectiveness in narrow environments is demonstrated through Monte Carlo simulations, and the trajectories generated by the algorithm are validated by tracking experiments using the tensegrity aerial vehicle.

The final part broadens UAVs' use to multi-domain environments with a miniature Uncrewed Aerial Underwater Vehicle (UAUV), capable of operation both in air and underwater. A pressure-based depth estimator and a control strategy for water breaching have been developed to facilitate the water-air transition. These tools help the UAUV to create a transition window, ensuring all propellers are exposed to air, and to determine the ideal timing to switch controller modes from water to air, thereby guaranteeing a successful transition.

To my cherished family, my supportive friends, and to all who have generously assisted me on my journey.

Contents

Contents **ii**

1 Introduction **1**
1.1 Dissertation outline 2
1.2 Contribution 3
1.3 Source publications 3

2 Related work in the literature **5**
2.1 Collision-resilient UAVs and tensegrity-structured applications 5
2.2 Work related to the collision-inclusive sampling-based planner for quadcopters 7
2.3 UAUV designs in the literature 9

3 Collision-resilient tensegrity aerial vehicle for cluttered environments **11**
3.1 Introduction 12
3.2 Design of the tensegrity shell 13
3.3 Dynamics model and control of tensegrity aerial vehicle 22
3.4 Validation with experimental vehicle 29
3.5 Conclusion 35

4 Exploiting collisions for quadcopter motion planning in cluttered environments **36**
4.1 Introduction 37
4.2 Quadcopter motion primitive and collision detector 38
4.3 Algorithm description 40
4.4 Illustrative example: motion planning in a tunnel 45
4.5 Experimentally tracking planned trajectories 47
4.6 Conclusion 50

5 Water-air transition with a miniature unmanned aerial underwater vehicle **51**
5.1 Introduction 52
5.2 Modeling of the system dynamics 53
5.3 Vehicle performance in different media and during transitions 56

5.4 State estimation and control strategy 59
5.5 Hardware design 63
5.6 Experimental validation 64
5.7 Conclusion 69

6 Conclusion and future work **70**
6.1 Conclusion 70
6.2 Future work 71

Bibliography **73**

Acknowledgments

First and foremost, I wish to express my deepest gratitude to my parents and partner. Your love and dedication have served as a strong pillar of support, for which I am profoundly grateful. I genuinely could not have reached this point without your assistance.

I would like to thank my advisor, Professor Mark Mueller, for his steadfast guidance and high standards that continuously pushed me to think harder and approach problems with greater rigor. Mark, you have reshaped my thought process and forever changed the way I approach problem-solving. The reasoning skill I have learned from you is one of the the most valuable assets I have gained during my time at Berkeley.

To all my dear lab mates and friends, I extend my heartfelt thanks for your support. Whether it was technical advice or emotional encouragement, your contributions have been instrumental in my journey.

The work in this dissertation would not have been possible without the funding support from the UC Berkeley Fellowship for Graduate Study, the UC Berkeley Jane Lewis Fellowship, the UC Berkeley Fire Research Group, and the USDA AI Institute for Next Generation Food Systems (AIFS). This work also relies on the experimental testbed at the HiPeRLab, which is the result of contributions from many people. A full list of these contributors can be found at **hiperlab.berkeley.edu/members/**.

Finally, I am grateful for the trails at the bay waterfront and behind the Berkeley campus. The whispers of the wind, the rhythm of the waves, and the shadows of the trees have all played a significant role in sustaining my spirits amidst the tumult of academic challenges and the COVID-19 pandemic. They have served as my silent therapists and potent energizers, and for that, I am truly thankful.

Chapter 1

Introduction

With improvements in computational power and sensing technologies, Uncrewed Aerial Vehicles (UAVs) have seen significant advancements in recent years. They have found use in a wide array of applications across multiple sectors, ranging from aerial photography [1] and mapping [2] to agricultural monitoring and spraying [3]. Despite these substantial strides, the full potential of UAVs remains untapped. Currently, UAV applications are mostly confined to open-air spaces with minimal obstacles, heavily relying on clear visibility and GPS signals.

The limitations of UAVs become apparent when compared with birds, the natural flyers that have inspired the initial design of aerial vehicles. Birds exhibit remarkable abilities far exceeding UAVs when operating in complex environments. For instance, torpedo gannets [4] and shags [5] can dive deep into water to catch fish, then immediately re-emerge into flight, demonstrating the ability to operate seamlessly in environments encompassing multiple media. Meanwhile, goshawks can fly in narrow spaces and cluttered environments by pushing their wings and claws against their surroundings, thereby incorporating contact into their flight [6]. These capabilities, honed over millions of years of evolution, are typically achieved through a combination of specialized body structures, agile kinesthetic skills, and efficient flight planning stategies.

In this dissertation, we explore methods to expand the operational environments and capabilities of UAVs. Similar to how birds have developed their specialized skills, the enhanced abilities of the aerial vehicles presented in this dissertation are accomplished through tight integration of physical advancements in design with computational enhancements in control strategies and motion planning algorithms.

Specifically, we introduce two types of specialized UAVs designed to operate in environments that pose challenges for traditional flying robots. The first is a collision-resilient tensegrity aerial vehicle. Thanks to its ability to withstand and recover from high-speed collisions, it can safely operate in cluttered environments with hard-to-detect and/or avoid obstacles. Additionally, we propose a motion planning algorithm that leverages collisions for quick changes of movement directions and more efficient random tree exploration. The second specialized UAV is an amphibious robot named miniature Uncrewed Aerial Underwater Vehicle (mini UAUV), capable of operating both underwater and in the air, and transi-

tioning between the two media. The mini UAUV features a soft, waterproof shell design that enables ambient pressure measurement without direct contact between water and the onboard pressure sensor. Moreover, a breaching strategy is proposed to facilitate the mini UAUV's control mode switch from water to air. With its specialized design and breaching control strategy, the mini UAUV can operate smoothly in a multi-domain environment without the need for any additional mechanical structure, achieving amphibious mobility while maintaining agility comparable to that of purely aerial vehicles.

1.1 Dissertation outline

This dissertation is organized as follows.

In Chapter 3, we discuss methods to expand UAV capabilities in cluttered environments. We introduce the tensegrity aerial vehicle, a collision-resilient flying robot designed with an icosahedron tensegrity structure. We establish an approach for predicting structural stresses during collisions via dynamics simulation, guiding component selection during the design process. Our approach leads to the successful creation of an experimental vehicle with robust collision resilience, capable of surviving a 7m drop with an 11.7m/s landing speed. Furthermore, we propose a re-orientation controller, enabling the vehicle to rotate on the ground post-collision, so as to reach an orientation easy for takeoff. This combination of collision resilience and post-collision flight resumption makes the tensegrity aerial vehicles ideally suited for field operations in cluttered environments. Related experimental videos can be viewed at **youtu.be/XsLVRd2nMd0**. The tools used to develop and analyze the tensegrity aerial vehicles can be accessed at **github.com/muellerlab/TensegrityAerialVehicle.git**.

In Chapter 4, we introduce a novel motion planning algorithm, which can enhance the ability of collision-resilient aerial vehicles in cluttered environments. The planner samples collisions as intersection between generated motion primitives and obstacles, and connects collision states with other sampled states to create collision-inclusive trajectories. We demonstrate that allowing for collision in planning improves the performance of the sampling-based planner in narrow spaces like tunnels. Additionally, we validate the trajectory generated by the planner with tracking experiments using the collision-resilient tensegrity aerial vehicle. Related experimental videos are available at **youtu.be/MmDHra3wYK4**.

In Chapter 5, we shift our focus to broadening the operational medium of aerial vehicles. Specifically, we present the design of a miniature uncrewed aerial underwater vehicle, which can fly in air, swim in water, and transition between these two media. The vehicle features a simplified mechanical design akin to a traditional quadcopter, eliminating the need for additional mechanical components. To facilitate the transition from water to air, we introduce a water-breaching control strategy, which can autonomously switch between the underwater and the aerial control modes. Related experimental videos can be viewed at **youtu.be/y4-ZcgsTGAQ**.

Chapter 6 concludes the dissertation and discusses potential future research directions.

1.2 Contribution

The contributions of this dissertation include proposing new UAV designs that can operate in challenging environments, their corresponding controllers, and related planning algorithms. Specifically:

1. **Collision-Resilient Tensegrity Aerial Vehicle:** 1) We present a model-based approach for designing collision-resilient tensegrity aerial vehicles, supported by a dynamics simulation tool we have open sourced. 2) We propose a re-orientation controller to facilitate flight resumption post-collision, and make the corresponding development and analysis tools open sourced. 3) We validate the design approach and the controller with an experimental vehicle and demonstrate its ability to survive collisions, resume flight, and perform short-range autonomous operations in an unknown environment.

2. **Collision-Inclusive Motion Planning:** 1) We present a sampling-based motion planner that can find collision-inclusive trajectories for quadcopters. 2) We demonstrate that the collision-inclusive planner is more likely to generate better result than collision-exclusive planner in narrow spaces with Monte Carlo studies. 3) We experimentally validate the planned collision-inclusive trajectory by tracking it with our tensegrity aerial vehicle.

3. **Mini UAUV:** 1) We propose a miniature UAUV design that can operate in water, air, and breach the still water surface without the aid of additional mechanical actuators. The vehicle is lightweight and can fly with agility comparable to purely aerial vehicles. 2) We create a barometer-based Kalman Filter depth estimator and a water-breaching control strategy that can switch the control mode based on the surrounding environment. 3) We experimentally validate the water breaching strategy.

Together, these works provide new approaches to enable UAVs to operate with enhanced safety in cluttered spaces, and with greater agility in multi-domain environments, two spaces traditionally considered infeasible for UAVs.

1.3 Source publications

The materials in this dissertation are based on following publications:

- Jiaming Zha, Xiangyu Wu, Ryan Dimick, and Mark W. Mueller, "Design and control of a collision-resilient aerial vehicle with an icosahedron tensegrity structure," *IEEE/ASME Transactions on Mechatronics (TMECH)*, under review.

- Jiaming Zha and Mark W. Mueller, "Exploiting collisions for sampling-based multicopter motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7943-7949.

- Jiaming Zha, Xiangyu Wu, Joseph Kroeger, Natalia Perez, and Mark W. Mueller, "A collision-resilient aerial vehicle with icosahedron tensegrity structure," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 1407–1412.
- Jiaming Zha, Eric Thacher, Joseph Kroeger, Simo A. Mäkiharju, and Mark W. Mueller, "Towards breaching a still water surface with a miniature unmanned aerial underwater vehicle," in *Proceedings of the International Conference on Unmanned Aerial Systems (ICUAS)*, IEEE, 2019, pp. 1178–1185.

Chapter 2

Related work in the literature

This chapter provides a review of the literature pertinent to the three core parts of this dissertation: collision-resilient tensegrity aerial vehicles, collision-inclusive quadcopter motion planning, and uncrewed aerial underwater vehicles (UAUV). The review is structured into three sections, each dedicated to one of these themes, providing an exploration of previous advancements and existing challenges that have informed the research objectives and methodologies presented in the subsequent chapters.

2.1 Collision-resilient UAVs and tensegrity-structured applications

One of the primary design objectives for UAVs is to decrease weight, as additional mass can impede UAVs' agility and maneuverability. However, this often results in structures that are susceptible to impacts and collisions. Contacts with the surrounding environment can compromise the operational abilities of UAVs, leading to premature termination of missions.

Despite significant advancements, collision-avoidance algorithms have yet to fully mitigate this issue. The limitations primarily stem from two areas: sensor technology and onboard computational power. Cameras, serving as critical sensors for most UAV collision-avoidance pipelines, frequently encounter difficulties in detecting thin objects like wires due to constraints dictated by sensing physics and processing algorithms. Additionally, their performance can deteriorate under extreme light conditions such as intense sunlight or pitch darkness. The limit of onboard computational power can also cap the performance of collision avoidance systems. In complex environments, the vehicle may fail to compute a safe trajectory under the given time limit, thereby increasing the risk of collisions.

Given these challenges, collision-resilient aerial vehicles provide a viable solution for operating safely in cluttered environments. These resilient designs give UAVs "second chances" when onboard collision-avoidance systems fail. Among the various collision-resilient designs, the concept of tensegrity holds considerable promise. Tensegrity refers to a structural prin-

ciple that suspends rigid components within a tension network. It can avoid large stress caused by bending and thus achieve substantial resilience with little structural weight.

In the following paragraphs, we will explore studies proposing designs for collision-resilient aerial vehicles. Additionally, we will discuss application examples using tensegrity structures to enhance collision resilience.

Collision-resilient flying robot designs

Main approaches to create collision-resilient flying robots in the literature include 1) protecting the robot with external structures, 2) constructing the robot with soft materials or morphing structures capable of absorbing substantial energy before breaking, and 3) using a combination of both.

The first approach emphasizes shielding vulnerable parts from obstacles with external protectors. For example, spherical body shells that can completely encase aerial vehicles have been proposed in [7] and [8]. Meanwhile, spherical propeller guards that can passively rotate around quadcopter motors are suggested in [9]. These guards can also serve as wheels, allowing the vehicle to horizontally move on the ground. Likewise, a cylindrical body guard that can fully envelop the vehicle and enable it to roll laterally on the ground is discussed in [10]. A freely rotating origami shell that offers lightweight protection is demonstrated in [11], and a mortise-and-tenon protector that can be created from 2D sheet materials and assembled into a 3D protective structure is detailed in [12].

The second approach utilizes materials or components specifically designed to absorb significant energy before breaking. Examples include dual-stiffness flight frames that soften upon impact to prevent damage [13], flexible rotor blades that can bend without breaking during collisions [14], and passively-foldable frames for quadcopters [15].

Some designs incorporate both approaches, protecting the vehicle with an external structure capable of absorbing a large amount of energy. For instance, a propeller-guarded vehicle with spring-loaded arms is proposed in [16], and a quadcopter with a passively-morphing exoskeleton is introduced in [17].

Tensegrity structure

Tensegrity structures have gained popularity in recent years for collision-resilient applications. Comprised of rigid bodies suspended in a tension network, tensegrity structures can distribute external loads among structural members through tension and compression, effectively avoiding large stress caused by bending. Due to this structural advantages, tensegrity structures have been proposed for applications in diverse areas such as aircraft wings [18], landers [19, 20, 21], exploratory rovers [22, 23], swarm terrestrial explorers [24], and general collision-resilient robotic platforms [25]. The benefits of tensegrities also make them suitable for aerial vehicles. An investigation comparing different tensegrity shells for aerial vehicles, supported by drop tests, is detailed in [26]. In [27], our first publication on tensegrity aerial vehicles, and the prior work to the research to be presented in Chapter 3, we create a

quadcopter with a stiff tensegrity shell that can survive high-speed collisions and rotate on the ground. Another example is the 'Tensodrone' which incorporates a soft tensegrity shell with springs, as showcased in [28], along with a subsequent design featuring self-morphing abilities. The soft tensegrity design helps increase collision resilience at the potential cost of larger vehicle size and vibration. The comparison between stiff and soft tensegrity shell designs will be further discussed in Chapter 3.

2.2 Work related to the collision-inclusive sampling-based planner for quadcopters

This section introduces literature related to the sampling-based collision-inclusive motion planner for quadcopters. Specifically, we will discuss quadcopter motion primitives, sampling-based planning structures, and methods to integrate collisions into motion planning schemes.

Quadcopter Motion Primitives

Motion primitives are short trajectory segments which form a critical foundation for quadcopter path planning. Planning algorithms for quadcopters usually revolve around the creation, selection, and connection of primitives to form safe reference trajectories.

The generation of quadcopter motion primitives typically leverages the property of differential flatness [29]. Differential flatness enables the full recovery of a dynamic system's states and inputs from a group of "flat states" and their finite order derivatives. In the context of a quadcopter, both thrust and attitude can be directly computed from acceleration, the second-order derivative of position. Consequently, quadcopter motion primitives only need to explicitly define the position, and the rest of the states (velocity, acceleration, attitude, etc.) and inputs (thrust and angular rates) will be implicitly encoded. This greatly reduces the dimensionality of the planning problem.

A crucial category of quadcopter primitives is the minimum snap primitive proposed in [30]. This method defines the quadcopter primitives as polynomial splines and solves a quadratic problem (QP) to minimize snap (the second derivative of acceleration) in order to determine the parameters of the splines. Improved formulations of minimum snap primitives designed for enhanced online computation speed were later proposed in [31] and [32]. The minimum snap trajectory is highly versatile and extendable. As these primitives are generated by solving optimization problems online, costs and constraints can be customized to meet various planning requirements. However, extra attention to the optimization problem formulation is needed to ensure the formulation possesses desirable properties. Otherwise, the problems may not be solved efficiently and reliably.

The minimum jerk primitives, introduced in [33], present a useful alternative for quadcopter motion planning. These primitives are defined as fifth-order polynomials of time, with fully or partially fixed start and end state, as well as a specified total time duration.

The optimization problem that minimizes the integration of jerk (the derivative of acceleration) for these primitives is found to have an elegant closed-form algebraic solution. This discovery paves the way for an efficient method of generating quadcopter motion primitives, enabling the creation of over a million minimum jerk primitives in less than a second using a modern laptop. Furthermore, the feasibility of control inputs (thrust and body rates) can be swiftly verified, and collisions between minimum jerk primitives and convex obstacles can be efficiently detected [34]. This makes these primitives particularly useful for tasks in cluttered environments. Their rapid generation speed, coupled with the abilities to quickly check input-feasibility and detect collisions, makes it possible to generate a large array of primitive candidates and select the best primitive during real-time operation.

Sampling-based planning

One effective approach to assist autonomous systems in finding feasible trajectories between desired states is to gather samples in the state space and connect them with feasible (in terms of both input-feasible and collision-free) trajectories. Methods using this approach, such as rapidly exploring random trees (RRT) [35] and probabilistic road maps (PRM) [36], are referred to as sampling-based planning methods. In particular, RRT*, a variant of RRT, has gained considerable popularity due to its unique asymptotic optimality characteristic [37]. Researchers have extended the RRT* algorithm to work with dynamic systems with differential constraints [38] and have developed various sampling methods [39] and heuristics to guide the sampling process [40, 41] in order to increase the rate of convergence for RRT*.

Collision-inclusive planning

A key step of motion planning is to check for possible collisions with the surrounding environment. Trajectories that intersect with obstacles are typically discarded by planners. However, in recent years, the development of many autonomous systems capable of withstanding collisions (vehicles detailed in Section 2.1 are perfect examples) has made a change in this perspective. Given that these vehicles can implement trajectories that are not collision-free, they can plan their motion within an expanded feasible state space. This can lead to better planning results with decreased trajectory cost or shorter duration time.

This concept of utilizing collisions for improved trajectories is discussed in [42], which also proposes a method for finding collision-inclusive optimal trajectories using mixed integer programming. The effectiveness of this method is later experimentally validated in [43]. Meanwhile, a collision-inclusive planning structure with proof of optimality for hybrid dynamical systems is proposed in [44]. Moreover, collisions have been found to be advantageous in stochastic optimal steering problems, as contact helps reduce the uncertainty in state estimation [45].

In Chapter 4, we will explore how collisions can be utilized for enhanced sampling-based motion planning for quadcopters.

2.3 UAUV designs in the literature

UAUVs are amphibious vehicles capable of operating both underwater and in the air. They can perform challenging tasks within multi-domain environments, such as inspecting offshore oil platforms and surveying coastal ecosystems. UAUVs appear in various forms in the literature, from waterproof fixed-wing aeroplanes [46] to squid-like soft robots [47]. A comprehensive survey of recent UAUV developments can be found in [48].

In the following discussion, we will primarily focus on UAUVs with vertical take-off and landing (VTOL) capabilities, a category to which our work to be introduced in Chapter 5 belongs. Compared to fixed-wing UAUVs, VTOL UAUVs exhibit greater agility and maneuverability. However, they also encounter larger challenges during water-air transitions due to the drastic change in propeller thrust from operating in water to operating in air, which have a 1000-fold difference in density. Moreover, during the transition, propellers' interaction with the fluid interface also creates a significant variability in the phase fraction of the operating media, primarily attributed to propeller-induced air entrainment. This entrainment is driven by a combination of shear forces at the water-air interface, vortex-induced entrainment, and air entrapment from falling liquid [49]. The interaction of these mechanisms generates a highly unsteady flow field in which the VTOL UAUV must operate, thereby making water-air transition more difficult. To overcome challenges during the transition process, VTOL UAUV designs in the literature typically use supportive mechanical structures, which are detailed below.

VTOL UAUV Designs

One popular approach for water-air transition for VTOL UAUVs is to use double-layered propellers. The Naviator UAUV series [50, 51], for instance, feature designs with eight propellers grouped in two layers and a hybrid control system. They can reliably generate thrust during the water-air transition as there is always a set of four propellers clearly underwater or in the air. Similarly, the Hybrid UAUV [52] employs two layers of propellers: four aerial propellers and four aquatic propellers specifically designed for underwater mobility. Accompanying the Hybrid UAUV, a robust switched control strategy is proposed and validated with high fidelity simulation [53]. Another double-layered design is presented in [54], wherein the vehicle employs an adaptive sliding mode dynamical surface control to transition between different media. An alternative design approach for media transition involves the use of buoyant devices. For example, the Loon-Copter [55] is a UAUV equipped with a controllable water ballast that can adjust the overall density of the vehicle. To emerge from the water, the Loon-Copter empties its ballast, floats to the water surface, and then takes off when all propellers are clearly out of the water. The SeaHawk Alfa drone [56] is designed with a detachable buoy. This buoy floats on the water surface and is connected to the drone body with cables. Prior to take-off from water, the drone reattaches itself to the buoy, using the additional buoyancy force to raise the propellers out of the water. Furthermore, a

piston-driven buoy system is applied in the NeZha III [57], a hybrid UAUV design with both rotor-based VTOL capability and wing-based horizontal glide ability.

While these additional mechanical structures facilitate media transition, they come at the cost of increased weight and reduced agility. In Chapter 5, we will introduce our approach, which creates a UAUV without additional mechanical structures with the help of a specialized breaching control strategy, thereby maintaining agility comparable to purely aerial vehicles.

Chapter 3

Collision-resilient tensegrity aerial vehicle for cluttered environments

In this chapter, we introduce an aerial vehicle design that can safely operate in cluttered environments filled with obstacles that are hard to detect and/or avoid. With an icosahedron tensegrity structure (a 20-face polygon composed of rigid rods suspended in a tension network of strings), the design is exceptionally collision-resilient. With a re-orientation controller, the vehicle can rotate on the ground to point its propellers upward, enabling it to take off again post-collision. To help the vehicle navigate in GPS-denied and vision-occluded environments, we have in addition adopted the inertial navigation strategy in [58], which decreases estimation error by breaking long trajectories into shorter hops and using pseudo-velocity measurement updates when the Inertial Measurement Unit (IMU) indicates the vehicle is stationary.

The development of the tensegrity aerial vehicle went through two key stages. The first stage involved a conceptual exploration. We attached an existing miniature quadcopter to a tensegrity shell (see the left vehicle in Fig. 3.1), and developed a naive re-orientation controller by adapting our existing flight attitude controller. The outcomes of this initial stage of research was detailed in the following paper:

- Jiaming Zha, Xiangyu Wu, Joseph Kroeger, Natalia Perez, and Mark W. Mueller, "A collision-resilient aerial vehicle with icosahedron tensegrity structure," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 1407–1412. [27]

During the second stage, an improved design fully leveraging the properties of the tensegrity structure was created (see the right vehicle in Fig. 3.1). By directly mounting the motors and electronics onto the tensegrity shell, we eliminated the need for rigid connections between the tensegrity rods, thereby enhancing the vehicle's resilience to collisions. In addition, we created a new and more accurate method for stress analysis based on a dynamics simulator, which can capture transient effects and structural deformation. Furthermore, we proposed an

Figure 3.1: The icosahedron tensegrity aerial vehicles. Left: stage one experimental vehicle, built with a miniature quadcopter attached to a tensegrity shell. Right: stage two experimental vehicle, all electronics of which are directly mounted on the tensegrity rods. Both vehicles are built with carbon fiber rod of 20cm length.

enhanced re-orientation strategy, which can systematically evaluate the feasibility of rotation between icosahedron faces, and map torque control signals to individual propeller commands, all while considering the feasible operational range of each propeller. The resulting vehicle has remarkable collision resilience, capable of withstanding impacts with speed of 11.7 m/s. Its ability to survive collisions, resume operation post-collision, and operating using only inertial sensors were validated through a field experiment conducted in a forest previously unknown to the vehicle. The discussion pertaining this second stage of our work, which is the primary focus of this chapter, has been submitted for potential publication:

- Jiaming Zha, Xiangyu Wu, Ryan Dimick, and Mark W. Mueller, "Design and control of a collision-resilient aerial vehicle with an icosahedron tensegrity structure," *IEEE/ASME Transactions on Mechatronics (TMECH)*, under review.

3.1 Introduction

Autonomous aerial vehicles, being weight-sensitive, are often fragile. Damage to their propellers or electronics can result in the loss of their ability to fly. Protective measures for these vehicles typically fall into two categories: detecting and avoiding collisions, and/or

preventing physical damage caused by collisions. Methods of the first category focus on sensing surrounding spaces and finding safe paths based on the collected information. A survey summarizing recent development in the area is in [59]. Methods of the second category, which the work in this chapter belongs to, help aerial vehicles operate more safely in cluttered environments, where accidental collisions may occur due to imperfect sensing or control. Many aerial vehicles with collision-resilient designs exist in the literature, and a review of them can be found in in Chapter 2.

In this chapter, we present the design of collision-resilient flying robots, termed tensegrity aerial vehicles, featuring icosahedron tensegrity structures. These vehicles incorporate the collision resilience of tensegrities and the mobility of quadcopters. They can withstand high-speed impacts and resume operation after collisions. With these capabilities, they can safely operate in cluttered environments without complex collision-avoidance strategies. To guide the design process of these vehicles, we propose a model-based approach that employs dynamics simulation to predict structural stresses during collisions, and to help us select components that can endure these stresses. Additionally, we create an autonomous re-orientation strategy to help the vehicles take off again after collisions. Leveraging the sphere-like geometry of the icosahedron, the tensegrity aerial vehicles can rotate from an arbitrary orientation on the ground to ones easy for takeoff. With collision resilience and re-orientation ability, tensegrity aerial vehicles can operate in cluttered environments without complex collision-avoidance strategies. Moreover, we further extend the vehicles' ability by adopting the inertial navigation method in [58], which enables the vehicles to perform short-range autonomous operations without external sensors. The resulting vehicles can thus serve as field robots and work on challenging tasks such as traversing through a cluttered corridor filled with smoke to search for survivors.

The rest of the chapter is organized as follows. In Section 3.2, we introduce the simulation-based approach used for designing the tensegrity, along with an analysis showcasing the structural advantages of the icosahedron tensegrity shells. Subsequently, in Section 3.3, we explore the dynamics of tensegrity aerial vehicles and introduce the re-orientation controller. Experimental validations are presented in Section 3.4, and finally, Section 3.5 concludes the chapter. Related experimental videos can be viewed at **youtu.be/XsLVRd2nMd0**. The tools used to develop and analyze the tensegrity aerial vehicles can be accessed at **github.com/muellerlab/TensegrityAerialVehicle.git**.

3.2 Design of the tensegrity shell

In this section, we motivate the idea of protecting a quadcopter with a stiff icosahedron tensegrity shell, introduce the approach used to design the tensegrity with stress analysis based on a dynamics simulation, and showcase the structural advantage of the tensegrity design. The tools for the related simulation and analysis can be accessed via the link in Section 3.1.

We choose to design the tensegrity aerial vehicle in the form of a quadcopter because

the abilities to hover and to vertically take off and land make operations easier in cluttered environments. Meanwhile, we choose to protect the quadcopter with a 6-rod orthogonal icosahedron [60] tensegrity, whose near-spherical shape offers omnidirectional protection with minimal structural weight.

The tensegrity shell's primary role is to shield the quadcopter from damage during collisions. Hence, it must withstand impacts without breaking, and its deformation should be small to prevent external obstacles from making contact with the internal components. Consequently, the successful design of the tensegrity depends on the selection of components (rods and strings) that possess suitable stiffness and strength.

We favor stiff components with little flexibility for two reasons. First, a stiff shell exhibits little deformation during collisions, requiring less buffer space to protect internal components like propellers from exposure. Thus, the tensegrity can be smaller in size, and this helps the vehicle to fit through narrow gaps. Second, a stiff tensegrity has little vibration within the shell structure, resulting in less flight disturbance. Meanwhile, we prefer lightweight components as they help retain the agility and flight time of the aerial vehicle. In the following subsection, we detail the process of determining whether certain components meet these design requirements.

Stress analysis with dynamics simulation

To predict if a tensegrity aerial vehicle can survive a collision, we simulate the dynamics of the tensegrity structure during the collision and calculate the stress in the structure with the simulation result. In contrast to the static stress analysis method previously proposed during the stage one of the tensegrity aerial vehicle research [27], this dynamics simulation method has two advantages. First, it accounts for the tensegrity deformation and captures the transient effects during the propagation of stress. Second, it considers the stress concentration caused by the mass of the quadcopter mounted on the tensegrity rods. These advantages lead to a more accurate stress estimate and allow us to easily verify if the tensegrity design meets the deformation criteria.

The tensegrity vehicle is modeled as point masses suspended in a stress network, as depicted in Fig. 3.2a. We define a *tensegrity node* as the point where a rod connects to strings. An icosahedron tensegrity has 12 nodes, each is a point mass representing the mass of the fasteners at the node's position, as well as half of the rod and the strings connected to that node. Each tensegrity node connects to a rod and four strings, represented as a massless linear spring-damper pair (see Fig. 3.2b).

The quadcopter is modeled using four evenly-distributed *quadcopter nodes*, which are mass nodes attached to a pair of parallel rods in the tensegrity shell. Each quadcopter node represents the mass of propeller and motor at its location, and a quarter of the batteries and electronics. These quadcopter nodes divide the *full-length rod* they're mounted on into three *short rods*. Each connection between short rods is represented as a torsional spring-damper pair, as shown in Fig. 3.2c, with the torsional spring constant derived from the rod bending model, as we will show later. It is important to note that our model does not treat the rod

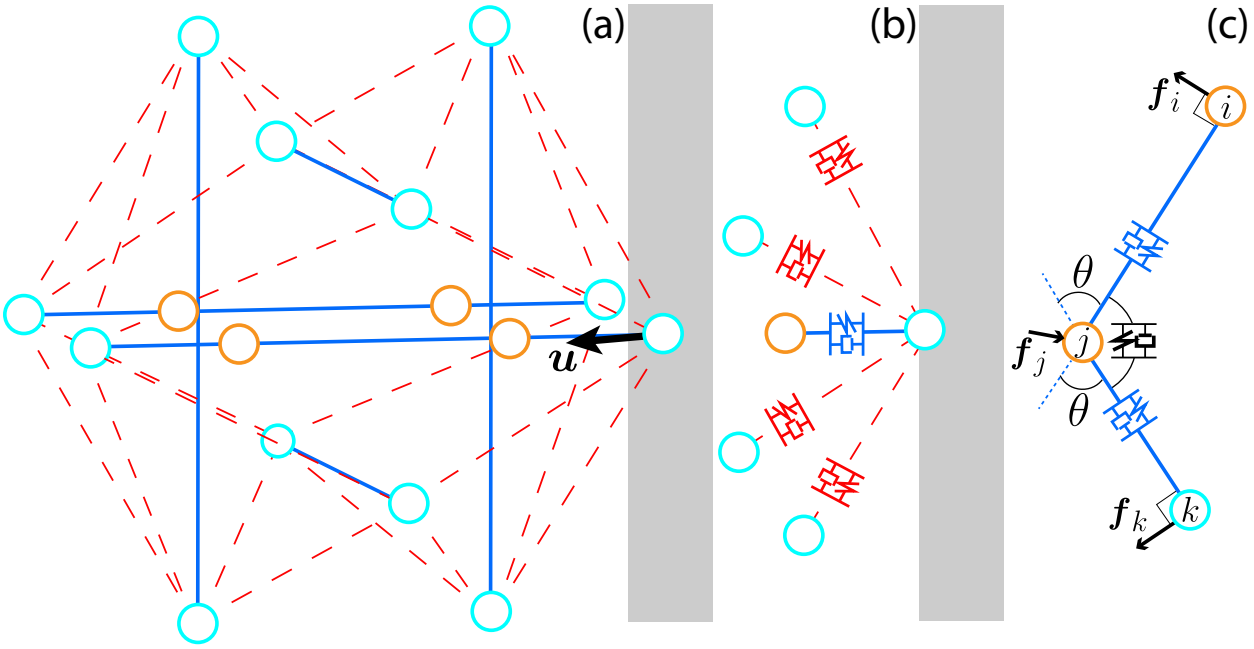

Figure 3.2: (a) The tensegrity vehicle is simplified as point masses in a stress network. Cyan spheres represent tensegrity nodes whereas orange spheres represent quadcopter nodes. (b) Strings and rods are modeled as massless spring-damper pairs. (c) Connections between two short rods are modeled as torsional spring-damper pairs.

hosting quadcopter nodes as a single, inflexible entity. Instead, it allows for relative rotation between short rods, facilitating the capture of transient stress concentration effects resulting from the uneven mass distribution caused by the mounting of the quadcopter.

We denote the nodes in the system as n_i, where $i = 1, ..., 12$ for the tensegrity nodes and $i = 13, ..., 16$ for the quadcopter nodes. The position of the i^{th} node is represented by $\boldsymbol{x}_i$. For simplicity and consistency, variables related to rods are denoted with a superscript r and those related to strings with a superscript s. The connectivity of the nodes can then be represented with indicator variables $N^r_{i,j}$ and $N^s_{i,j}$:

$$N^r_{i,j} = \begin{cases} 1, \text{ if a rod connects } n_i \text{ and } n_j \\ 0, \text{ otherwise} \end{cases} \tag{3.1}$$

$$N^s_{i,j} = \begin{cases} 1, \text{ if a string connects } n_i \text{ and } n_j \\ 0, \text{ otherwise} \end{cases} \tag{3.2}$$

We further define $T_{i,j}$ and $C_{i,j}$ as the value of the tensile force in string and compression force in rod connecting node n_i and n_j. They can be calculated from Hooke's law, with a

special modification that compressed strings generate no force:

$$T_{i,j} = \begin{cases} N^s_{i,j} K^s (L_{i,j} - L^s), \text{ if } L_{i,j} \geq L^s \\ 0, \text{ otherwise} \end{cases} \tag{3.3}$$

$$C_{i,j} = N^r_{i,j} K^r_{i,j} (L^r_{i,j} - L_{i,j}) \tag{3.4}$$

where K^s and $K^r_{i,j}$ respectively are the spring constants of the string and the rod. We use subscripts i and j to specify rod-related variables, as these indices correspond to the two end nodes of the rod. $L_{i,j} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||$ is the distance between node n_i and n_j. L^s and $L^r_{i,j}$ are the corresponding pre-deformation length of the string and the rod. Note that due to the existence of possible self-stress (also known as pre-tension) in the icosahedron tensegrity [61], tensegrity components may be deformed even without an external load.

In addition to the tensile and compression forces, there exist linear damping forces which inhibit relative linear motion between nodes. We assume that each damping force is aligned with the corresponding string or rod, and its value is proportional to the relative velocity of the nodes:

$$D_{i,j} = (N^s_{i,j} + N^r_{i,j})\, c_{i,j} (\dot{\boldsymbol{x}}_j - \dot{\boldsymbol{x}}_i)^T \boldsymbol{e}_{i,j} \tag{3.5}$$

where $\boldsymbol{e}_{i,j} \in \mathbb{R}^3$ is the unit vector pointing from n_i to n_j and $c_{i,j}$ is the corresponding linear damping coefficient.

Additionally, for the connections between short rods, which are modeled as torsional spring-damper pairs and shown in Fig. 3.2c, we assume the spring moment is proportional to the angle between the neighboring rod and the damping moment is proportional to the angular velocity:

$$M_j = \begin{cases} \xi_j \theta + c_j \dot{\theta}, \text{ if } n_j \text{ connects two short rods} \\ 0, \text{ otherwise} \end{cases} \tag{3.6}$$

where ξ_j is the torsional spring constant at n_j, derived from rod bending model, while c_j is the torsional damping constant at n_j. For pure bending of a rod, the radius of curvature equals the product of Young's modulus E^r and second moment of area I^r, divided by the bending moment M_j [62]. Given that the bending angle is the ratio between the rod length and the radius of curvature, the torsional spring constant can be computed as:

$$\xi_j = \frac{E^r I^r}{L_{i,k}} \tag{3.7}$$

As the mass of rods is assumed to be lumped at the nodes, the moment is equivalent to forces acting on nodes at the ends of the rods in orthogonal directions and a balancing force acting on the joint:

$$\boldsymbol{f}_i = \frac{M_j}{L_{i,j}} \boldsymbol{e}_{\perp ji}, \quad \boldsymbol{f}_k = \frac{M_j}{L_{j,k}} \boldsymbol{e}_{\perp jk}, \quad \boldsymbol{f}_j = -(\boldsymbol{f}_k + \boldsymbol{f}_i) \tag{3.8}$$

where $\boldsymbol{e}_{\perp ji}$ and $\boldsymbol{e}_{\perp jk}$ are respectively unit vectors perpendicular to $\boldsymbol{e}_{j,i}$ and $\boldsymbol{e}_{j,k}$, pointing in the directions that would decrease the joint angle θ.

Let $\boldsymbol{f}_{bi}$ represent the force due to bending on node n_i and let $\boldsymbol{u}_i$ represent the external force acting on n_i. We use a method similar to [63] and derive the equations of motion of the system with Newton's second law for each node i:

$$\boldsymbol{f}_{bi} + \boldsymbol{u}_i + \sum_j (T_{i,j} - C_{i,j} + D_{i,j})\boldsymbol{e}_{i,j} = m_i \ddot{\boldsymbol{x}}_i \tag{3.9}$$

where m_i is the mass of n_i.

To simulate the system's dynamics, we need to define the external forces that act on the tensegrity during the collision process. We estimate the force by simplifying the obstacle as a stiff linear spring, and assume the magnitude of the force acting on the tensegrity is proportional to the distance that the tensegrity node has penetrated the obstacle:

$$||\boldsymbol{u}_i|| = k_o p_i \tag{3.10}$$

where k_o is the stiffness of the obstacle and p_i is the penetration distance of node n_i. Studies on the stiffness of common obstacles like concrete walls can be found in the literature [64]. We assume the surface of the obstacle is frictionless, so reaction forces are normal to the surface of the obstacle.

We simulate the dynamics system described by Eq. (3.9) by providing the initial position and velocity of the nodes and numerically solving the corresponding initial value problem with the Radau method in the SciPy library [65], which is chosen for its good general performance with stiff problems. The solution gives us positions of nodes over the simulated time, and we can then extract the tensile and compressive forces, as well as bending moment from these positions. The axial stress in the string or rod connecting nodes n_i and n_j can then be expressed as a function of simulation time as follows:

$$\sigma^s_{i,j}(t) = \frac{T_{i,j}(t)}{A^s}, \quad \sigma^r_{i,j}(t) = \frac{C_{i,j}(t)}{A^r} \tag{3.11}$$

where $\sigma^s_{i,j}$ and $\sigma^r_{i,j}$ are axial stress in corresponding strings and rods respectively. A^s and A^r are cross sectional areas of strings and rods. Furthermore, when considering the connection between two short rods, we can use the rod bending formula from [62] to compute the stress induced by bending at the rod's surface:

$$\sigma^b_j(t) = \frac{M_j(t) r}{I^r} \tag{3.12}$$

where r is the radius of rod. Thus, we can calculate the maximum stress at the node connecting two short rods as the sum of the bending stress at the rod surface and the maximum axial stress in the rods connected to it:

$$\sigma^r_j(t) = \sigma^b_j(t) + \max_i(\sigma^r_{i,j}(t)) \tag{3.13}$$

We can then use the computed stress information to check if the candidate components meet the design objectives. First, the stresses in the strings have to be smaller than their yielding strength σ^{sy}, with a factor of safety for string η^s:

$$\forall i, j, t \quad \eta^s \sigma^s_{i,j}(t) < \sigma^{sy} \tag{3.14}$$

Second, we need to ensure that the axial stress in each rod is less than its yield strength σ^{ry} and critical buckling strength $\sigma^{rb}_{i,j}$, with a safety factor for the rod η^r:

$$\forall i, j, t \quad \eta^r \sigma^r_{i,j}(t) < \min(\sigma^{ry}, \sigma^{rb}_{i,j}) \tag{3.15}$$

Here the rod's critical buckling strength can be approximated with Euler's buckling theory:

$$\sigma^{rb}_{i,j} = \frac{\pi^2 E^r I^r}{A^r (L^r_{i,j})^2} \tag{3.16}$$

Third, the stresses at the nodes connecting two short rods should also be smaller than the rod yielding strength with the safety factor for rod:

$$\forall j, t \quad \eta^r \sigma^r_j(t) < \sigma^{ry} \tag{3.17}$$

In addition to these stress conditions, we also need to ensure that the propellers and electronic components are not exposed during collisions. This can be done by computing the distances between the tensegrity surface and the quadcopter nodes, and ensuring that they are larger than a given threshold. By using this dynamics simulation along with stress checks and exposure checks, we can efficiently rule out candidate components that don't meet our design objectives without the need to physically construct and test the tensegrity structures.

Structural advantage of the icosahedron tensegrity

In an icosahedron tensegrity shell, external loads are distributed among structural members as tension and compression, thereby avoiding large stress caused by bending. As a result, an icosahedron tensegrity shell can better survive collisions than common protective structures like propeller guards. We illustrate this structural advantage through a Monte Carlo study, which simulates wall-collision experiments and compares the maximum stresses in two aerial vehicle designs (a tensegrity and a propeller-guarded) during the collisions.

Both designs for our simulated experiments host a quadcopter with a total mass of m_q and propellers of diameter d. The first design has the smallest tensegrity shell that can fully enclose the propellers, whereas the second design features the smallest propeller-guarded frame able to host the quadcopter. For simplicity, we depict the vehicles' body-fixed frames with three axes ($\boldsymbol{e}^{\boldsymbol{B}}_x$, $\boldsymbol{e}^{\boldsymbol{B}}_y$, $\boldsymbol{e}^{\boldsymbol{B}}_z$) orthogonal to each other, as illustrated in Fig. 3.3. We assume that the tensegrity shell and the propeller guard frame both have the same mass m_s and are composed of solid cylindrical rods of identical material, thus sharing the same density ρ^r and Young's modulus E^r. To fully define the tensegrity structure, we in addition specify

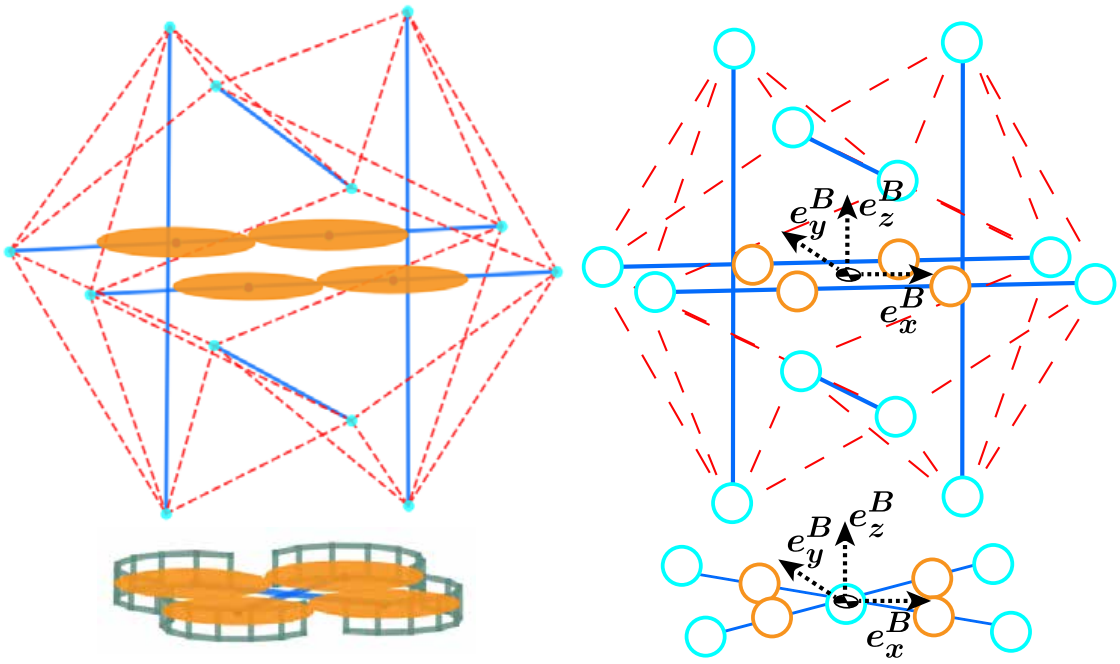

Figure 3.3: Left: illustration of the two collision-resilient aerial vehicles used for comparison. The top has a tensegrity shell whereas the bottom uses a propeller guard. Both vehicles have the smallest possible protection structure to host quadcopters with propellers of the same size. Right: we model both vehicles as point masses suspended in a stress network. We describe the vehicle's body-fixed frame with a set of three axes orthogonal to each other: $\boldsymbol{e}_x^B$, $\boldsymbol{e}_y^B$ and $\boldsymbol{e}_z^B$. Notice that for the tensegrity aerial vehicle, the quadcopter nodes are on the rods parallel to the $\boldsymbol{e}_x^B$ axis.

γ_m, the ratio between the total mass of rods and total mass of strings in the tensegrity shell, and F_s, the pre-tension force in strings. It is worth noting that our analysis shows the maximum stress during a collision is insensitive to these parameters. Meanwhile, similar to the tensegrity model, the quadcopter with propeller guard is simplified as point masses in a network of rods modeled as massless linear spring-damper pairs connected by joints modeled as torsional springs and dampers. Notice that there is a minor difference in the model: for the joints connecting perpendicular rods, the rest angles corresponding to zero moment are $\frac{\pi}{2}$. Moreover, for both the tensegrity vehicle and the propeller-guard vehicle, we assume the dampers will make the corresponding systems, including the spring-damper pair and their directly-connected nodes, critically damped. In our simulated experiments, we consider a wall with stiffness k_o. Before the collision, the tensegrity aerial vehicle and the propeller guard vehicle move perpendicularly toward the wall with a speed v. Both vehicles do not rotate before collisions. In the Monte Carlo study, we simulate 2000 experiments, each with a different random collision orientation generated by the following steps. First, we randomly sample points with a uniform distribution on the surface of a unit sphere attached to the

vehicle's body-fixed frame. Then, we compute the collision orientation as the rotation which maps the vector pointing from the origin to the sampled point in the body-fixed frame to a vector pointing perpendicularly to the wall in an inertial frame attached to the wall. Due to the symmetry of both vehicles, we only need to study orientations corresponding to nodes sampled in a single octant (one of the eight divisions of the Euclidean space separated by the three orthogonal axes) of the sphere surface.

The Monte Carlo study are conducted with key parameters in Table 3.1. The parameters correspond to a tensegrity shell made with carbon fiber rods and braided nylon string. Additionally, we choose $F_s = 20$N, which corresponds to a stiff shell without large pre-tension stress in the system. Notice that given the same structural mass budget, the rods used in the tensegrity are longer and therefore thinner.

Table 3.1: Key parameters used in the comparison simulation example

Parameter	Value
total structure mass	$m_s = 50$g
total quadcopter mass	$m_q = 250$g
string pre-tension	$F^s = 20$N
rod-string mass ratio	$\gamma_m = 20$
rod density	$\rho^r = 2000\text{kg/m}^3$
string density	$\rho^s = 1150\text{kg/m}^3$
rod Young's modulus	$E^r = 3.2\times10^{10}$Pa
string Young's modulus	$E^s = 4.1\times10^{9}$Pa
diameter of 2.5-inch propellers	$d = 63$mm
wall stiffness	$k_o = 4.7\times10^{7}$N/m
initial speed before collision	$v = 5$m/s

The result of the Monte Carlo study, visualized in Fig. 3.4, shows that the tensegrity holds a structural advantage over the propeller guard for collision resilience. Among the 2000 simulations, the tensegrity's mean maximum stress is 34.4MPa, compared to 100.5MPa in the propeller guard. For 80% of the samples, the maximum stress in the tensegrity vehicle is smaller than half of that in the propeller-guarded vehicle. On the other hand, the propeller-guarded vehicle experiences a smaller maximum stress than the tensegrity aerial vehicle (i.e. propeller guard is superior) in only 2.7% of the cases. Moreover, note that the high-stress points in the tensegrity plot are not symmetrically distributed. This comes from the non-uniform placement of the quadcopter parts, which are solely attached to the rods parallel to the $\boldsymbol{e}_x^{\boldsymbol{B}}$ axis. Hence, the most severe stress is experienced when these rods collide perpendicularly with the wall. In such circumstances, the deformation within the tensegrity structure is restricted, leading to a less effective load distribution. This analysis result suggests that during high-speed operations, tensegrity aerial vehicles should avoid flying with these rods pointing forward to avoid structural failures.

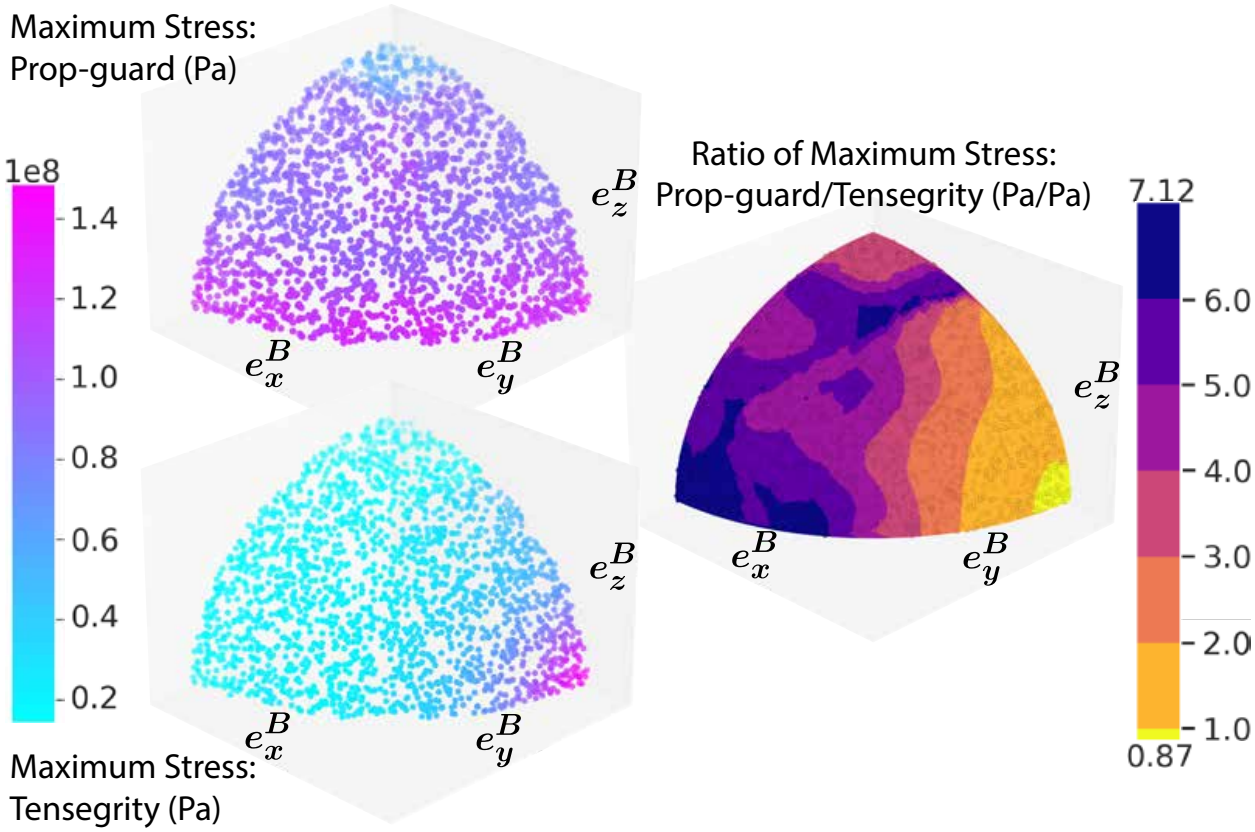

Figure 3.4: Visualization of the Monte Carlo study result. The positions of the points correspond to the collision orientation. Top left: scatter plot of the maximum stress in propeller guard during collisions. Bottom left: scatter plot of the maximum stress in tensegrity. Right: the ratio of the maximum stress in propeller guard to that in tensegrity. Larger values indicate more tensegrity advantage. The color on the surface is interpolated from the scattered simulated experiment data points.

In addition, we have investigated the structural advantage of the icosahedron tensegrity for vehicles of various scales. Specifically, we scale all length-related parameters linearly, all mass and force-related parameters cubically, while maintaining the ratios and material characteristics parameters from Table 3.1. We then conduct the same Monte Carlo analysis for the scaled tensegrity aerial vehicle and propeller-guarded vehicle and record the ratio of maximum stresses. The result of the analysis is shown in Fig. 3.5. As the vehicle size increases, the icosahedron tensegrity's collision resilience relative to the propeller guard also improves. As the propeller guard increases in size, bending becomes the primary source of stress due to the increased moment arm length. Consequently, as the vehicle scales up, the maximum stress in the tensegrity shell increases at a slower rate than that in the

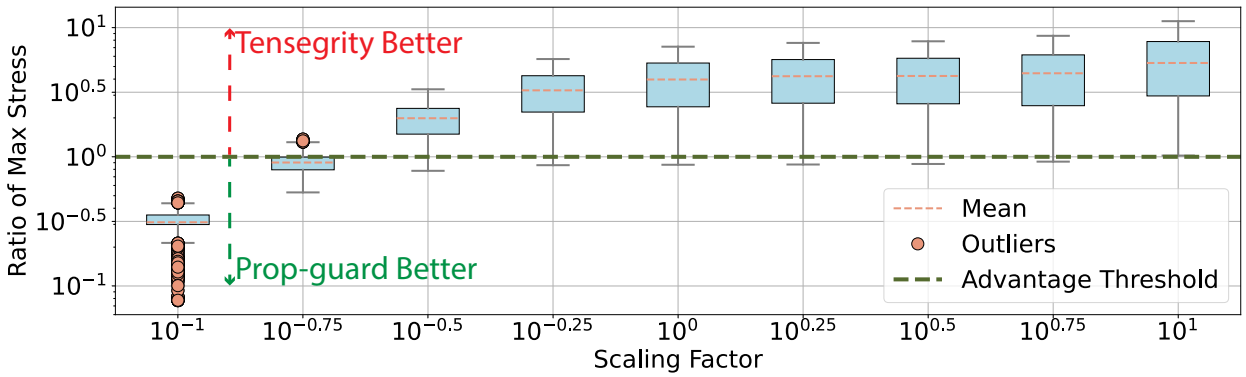

Figure 3.5: This figure illustrates the structural advantage of tensegrity over propeller guard for aerial vehicles of varying scales. The horizontal axis represents the scaling factor, while the vertical axis indicates the relative advantage, measured as the ratio of the maximum stress in the propeller guard to that in the tensegrity during collision simulations. The rising trend suggests the tensegrity's advantage becoming more prominent with larger vehicles.

propeller guard. Conversely, as the vehicle size decreases, the propeller guard becomes more effective compared to the tensegrity. However, at smaller scales, the maximum stresses during collisions also decrease, and factors like air resistance become dominant, which in turn reduces the necessity for high-speed collision resilience.

3.3 Dynamics model and control of tensegrity aerial vehicle

In this section, we introduce the models and controllers of the tensegrity aerial vehicles. The vehicles primarily execute two types of motion: in-flight, they operate like standard quadcopters with a flight controller; on the ground, they employ a re-orientation controller to rotate themselves to an orientation with propellers pointing upward, preparing for takeoff.

During stage one of the tensegrity aerial vehicle development [27], we proposed a strategy that repurposed the flight attitude controller for re-orientation. It relied on physical experiments to determine the feasibility of rotations and imposed a constraint of sum of thrusts being zero, limiting the rotational torque the vehicle could produce. This section introduces a new re-orientation strategy, which offers two improvements. First, it systematically determines rotation feasibility and plans re-orientation paths. Second, it incorporates a new thrust converter that optimizes the vehicle's re-orientation torque command while considering the thrust constraints. Hence, it increases the reliability of the re-orientation process.

This section aims to provide a generalized strategy for modeling and controlling tensegrity

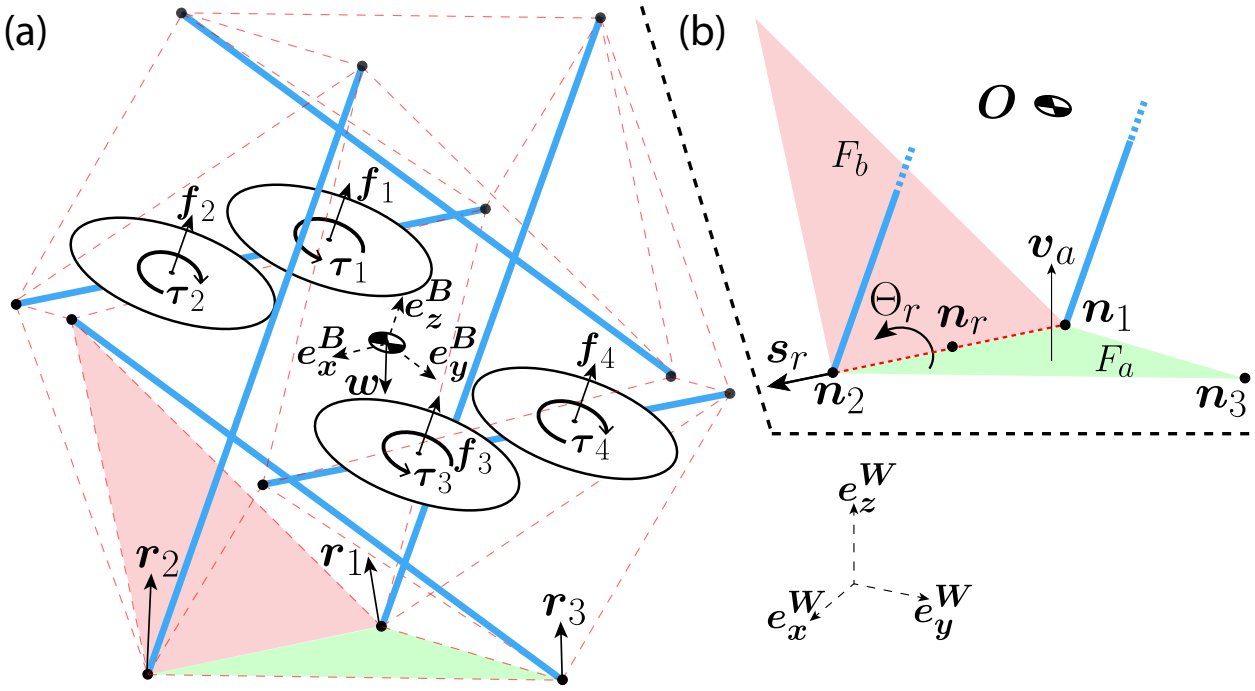

Figure 3.6: The figure shows the dynamics model of a tensegrity aerial vehicle during a rotation between faces. (a) The vehicle experiences weight $\boldsymbol{w}$, thrusts and yaw torques generated by the propellers, f_i and τ_i (where $i = 1, 2, 3, 4$), and contact forces $\boldsymbol{r}_j$ (where $j = 1, 2, 3, \ldots$). $\boldsymbol{W}$ denotes the world frame fixed to the ground and $\boldsymbol{B}$ denotes the vehicle's body-fixed frame. (b) The tensegrity rotates from face F_a to its neighbor face F_b. We study the rotation about rotation point $\boldsymbol{n}_r$ with a rotation axis $\boldsymbol{s}_r$ and a total rotation angle Θ_r. The tensegrity contacts the environment at multiple points denoted by $\boldsymbol{n}_j$.

aerial vehicles. For discussions on the specific experimental vehicle we created, please refer to Section 3.4. The code for related re-orientation analysis is available in our open source repository (see link in Section 3.1).

Vehicle dynamics and controller during flight

Given the stiff shells of the tensegrity aerial vehicles, we make the assumption that the vehicles behave as rigid bodies when they are not colliding with obstacles. As a result, the vehicles are modeled identically to standard quadcopters, and conventional quadcopter controllers are utilized for flight operations.

Vehicle dynamics for re-orientation

The model used to describe the dynamics of a tensegrity aerial vehicle during re-orientation is shown in Fig. 3.6a. External forces and torques include weight of the vehicle, thrust and torque from propellers, and reaction forces and torques due to contact with the environment. The vehicle's attitude is defined as a rotation matrix $\boldsymbol{R}$, mapping vectors from the body-fixed frame $\boldsymbol{B}$, which is affixed to the center of mass $\boldsymbol{o}$, to the world frame $\boldsymbol{W}$, which is inertial and affixed to the ground, i.e., $\boldsymbol{v^W} = \boldsymbol{R v^B}$. To avoid possible confusion, when a vector is used in analysis across different frames, we use superscript to indicate in which frame the vector is expressed.

The translational dynamics comes from Newton's law:

$$m\ddot{\boldsymbol{d}} = \boldsymbol{w^W} + \boldsymbol{R e_z^B} \sum_{i=1}^{4} f_i + \sum_j \boldsymbol{r_j^W} \tag{3.18}$$

where m is the vehicle mass, $\boldsymbol{w}$ is its weight, $\boldsymbol{d}$ is its position relative to a fixed point in the world frame, $\ddot{\boldsymbol{d}}$ is the cooresponding acceleration, f_i is the thrust generated by the i^{th} propeller, $\boldsymbol{e_z^B}$ is a unit vector pointing along the z-axis of the body-fixed frame, and $\boldsymbol{r}_j$ represents the reaction force from the environment acting on node j.

Rotational dynamics is modeled with Euler's equation:

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} + \boldsymbol{S}(\boldsymbol{\omega})\,\boldsymbol{J}\boldsymbol{\omega} = \sum_{i=1}^{4} f_i \boldsymbol{m}_{o,i} + \sum_j (\,\boldsymbol{S}\Big(\boldsymbol{n_j^B}\Big)\,\boldsymbol{r_j^B}) \tag{3.19}$$

where $\boldsymbol{S}(\cdot)$ maps a $\mathbb{R}^3$ vector to a corresponding $\mathbb{R}^{3\times3}$ skew-symmetric matrix. Left multiplying $\boldsymbol{S}(\cdot)$ is equivalent to the cross product. $\boldsymbol{J}$ is the moment of inertia matrix of the vehicle with respect to its center of mass, and $\boldsymbol{n}_j$ is the position of the node j that is in contact with the environment. Meanwhile, the angular velocity vector of the vehicle, $\boldsymbol{\omega} \in \mathbb{R}^3$, represents the rotation velocity between the body-fixed frame and the world frame, and it relates to the attitude matrix as follows:

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\,\boldsymbol{S}(\boldsymbol{\omega}) \tag{3.20}$$

Moreover, $\boldsymbol{m}_{o,i} \in \mathbb{R}^3$ represents the torque with respect to the center of mass generated by the i^{th} propeller with a unit thrust, and can be computed as:

$$\boldsymbol{m}_{o,i} = \boldsymbol{S}\Big(\boldsymbol{p_i^B}\Big)\,\boldsymbol{e_z^B} + h_i \kappa \boldsymbol{e_z^B}, \tag{3.21}$$

where $\boldsymbol{p}_i$ is the position of propeller i, h_i denotes the handedness of the propeller i (1 for right-handed, -1 for left-handed), and κ is the propeller's torque coefficient. On the right of Eq. (3.21), the first term represents the torque coming from the cross product of the moment arm and its respective thrust force, whereas the second term captures the drag torque from propeller rotation.

Re-orientation strategy

To facilitate the resumption of flight after collisions, a re-orientation controller is created to rotate the vehicles from arbitrary orientations to ones easy for takeoff. An icosahedron tensegrity has twenty faces, and we define two faces as neighboring if they share two nodes. Assuming the tensegrity is on flat ground, we denote the face in contact as F_i if the i^{th} face is touching the ground. Fig. 3.7 shows an unfolded icosahedron, illustrating the neighboring relationship of tensegrity faces. When face 4 or 9 (highlighted in the figure) is the contact face, the propellers point upward, indicating the tensegrity is prepared for takeoff. Thus, the objective of the re-orientation is to rotate the tensegrity aerial vehicle so that face 4 or 9 becomes the contact face.

The re-orientation strategy decomposes the task into a sequence of rotations between neighboring tensegrity faces, offering two key benefits. First, each rotation is simple to model, as the rotation axis corresponds to the line shared by the neighboring faces and the total rotation angle is determined by the icosahedron shape. Second, the strategy simplifies the problem into a finite state machine, thus enhancing robustness. If a rotation fails and the vehicle lands on an unexpected face, the controller can re-plan the path and continue with the task.

We define the re-orientation paths as a series of desired rotations to rotate the vehicle from the starting faces to a goal face. To find these paths, we follow a two-step procedure. First, we create a connection graph where the nodes represent the contact faces, and directed edges indicate feasible rotations between neighboring faces. Then, we search on this graph to find the shortest path from any starting face to its closest goal face.

The feasibility of a rotation between neighboring faces is evaluated by assessing whether the tensegrity aerial vehicle can generate a set of thrusts that counteract the gravitational

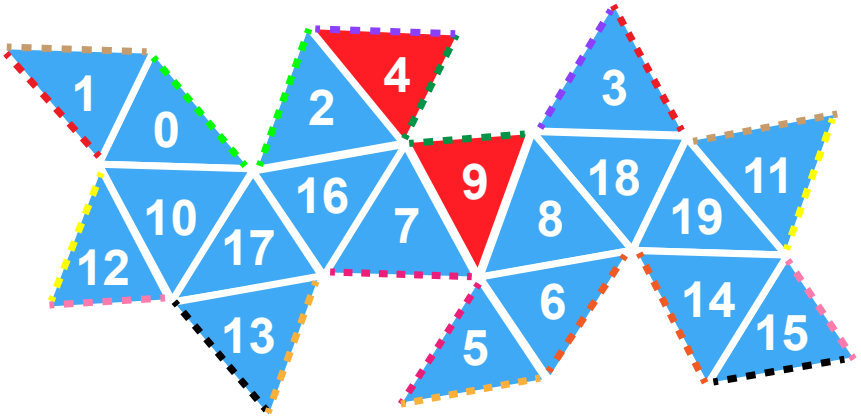

Figure 3.7: Faces of an unfolded icosahedron tensegrity. The tensegrity can take off when face 4 or 9 is contacting the ground. Dashed lines with the same color indicate overlapped edges when the tensegrity is folded back.

torque, without causing the vehicle to slide or leave the ground. We assume that the electronic speed controllers (ESCs) of the vehicles are configured to drive the propellers bi-directionally, enabling the vehicles to generate additional torque for re-orientation. For the following analysis, we use the notation illustrated in Fig. 3.6b, where F_a denotes the starting face and F_b represents the neighboring face to rotate to. We denote $\boldsymbol{n}_r$ as the rotation point, $\boldsymbol{s}_r$ as the rotation axis in the body-fixed frame, and Θ_r as the rotation angle. For a rotation to be feasible, there must exist a set of thrusts $[f_1, f_2, f_3, f_4]$ and reaction forces $[\boldsymbol{r}_1, \boldsymbol{r}_2]$ satisfying following conditions:

$$
\begin{cases}
\boldsymbol{e}_z^{\boldsymbol{B}} \sum_{i=1}^{4} f_i + \sum_{j=1}^{2} \boldsymbol{r}_j^{\boldsymbol{B}} + \boldsymbol{w}^{\boldsymbol{B}} = \boldsymbol{0} & (3.22) \\
\sum_{i=1}^{4} f_i \boldsymbol{m}_{r,i} + \sum_{j=1}^{2} \boldsymbol{S}(\boldsymbol{n}_j^{\boldsymbol{B}} - \boldsymbol{n}_r^{\boldsymbol{B}}) \boldsymbol{r}_j^{\boldsymbol{B}} = \boldsymbol{S}(\boldsymbol{n}_r^{\boldsymbol{B}}) \boldsymbol{w}^{\boldsymbol{B}} & (3.23) \\
0 \leq \boldsymbol{r}_j^{\boldsymbol{B}} \cdot \boldsymbol{v}_a^{\boldsymbol{B}}, \forall j \in \{1, 2\} & (3.24) \\
||\boldsymbol{r}_j^{\boldsymbol{B}} - (\boldsymbol{r}_j^{\boldsymbol{B}} \cdot \boldsymbol{v}_a^{\boldsymbol{B}}) \boldsymbol{v}_a^{\boldsymbol{B}}||_2 \leq \mu (\boldsymbol{r}_j^{\boldsymbol{B}} \cdot \boldsymbol{v}_a^{\boldsymbol{B}}), \forall j \in \{1, 2\} & (3.25) \\
f_{min} \leq f_i \leq f_{max}, \forall i \in \{1, 2, 3, 4\} & (3.26)
\end{cases}
$$

The above conditions specify the scenario when the tensegrity vehicle fully compensates its gravitational torque and is about to initiate rotation. At this moment, the third contact point is about to leave the ground. Thus, its reaction force goes to zero and is therefore not included in the equations. Eq. (3.22), derived from Newton's law, describes the force balance, while Eq. (3.23) illustrates the balance of the moments about the rotation point, $\boldsymbol{n}_r$. We use $\boldsymbol{m}_{r,i} \in \mathbb{R}^3$ to represent the torque with respect to $\boldsymbol{n}_r$ generated by a unit thrust of propeller i. Similar to Eq. (3.21), it is computed as:

$$
\boldsymbol{m}_{r,i} = \boldsymbol{S}\left(\boldsymbol{p}_i^{\boldsymbol{B}} - \boldsymbol{n}_r^{\boldsymbol{B}}\right) \boldsymbol{e}_z^{\boldsymbol{B}} + h_i \kappa \boldsymbol{e}_z^{\boldsymbol{B}}, \tag{3.27}
$$

The constraint (3.24) assures that the two nodes retain contact with the ground, as the reaction forces have non-negative components along $\boldsymbol{v}_a$, the unit ground-normal vector. Meanwhile, the no sliding condition is captured by (3.25), where μ denotes the friction coefficient between the vehicle and the ground. The feasible propeller thrust range is given in (3.26), where f_{max} represents the maximum thrust each propeller can generate, and f_{min} symbolizes the negative thrust value produced when the propeller spins reversely at peak speed.

Based on the feasibility analysis, a connection graph can be constructed. A fully connected graph indicates that the vehicle can re-orient to any contact face. In contrast, the presence of disconnected nodes on the graph indicates that the vehicle is incapable of leaving the corresponding contact faces through rotation. This suggests that the vehicle fails to generate enough torque to counterbalance the gravitational torque under the thrust range, the contact constraint and the no-sliding constraint. A design update incorporating stronger motors and/or longer moment arms, and a recheck of re-orientation feasibility are recommended to solve the problem. Once the connection graph has been constructed, the shortest

paths from any starting face to the goal faces can be determined. Section 3.4 provides an example of generating the re-orientation paths for our experimental vehicle.

Reference rotation trajectory for re-orientation

For each re-orientation step, the controller first identifies the rotation required for the face change, generates a reference rotation trajectory, and then tracks the generated trajectory. We employ a two-piece trajectory, which accelerates from a stationary state to the maximum angular velocity for the first half of the duration and then decelerates to stop for the second half. The angular acceleration remains constant in magnitude throughout, with a direction change at the midpoint of the duration. We have opted for this trajectory since it allows for straightforward tuning of the reference angular acceleration (which determines the aggressiveness of the trajectory) by adjusting the total trajectory time, denoted as T. Note that the total rotation angle Θ_r is constant, so the magnitude of the reference angular acceleration solely depends on T:

$$||\ddot{\boldsymbol{\Theta}}_{ref}|| = \frac{4\Theta_r}{T^2} \tag{3.28}$$

Ideally, in the absence of tracking error, the total torque command equals the sum of the gravitational torque offset and the torque to generate the desired angular acceleration, which is inversely proportional to T^2. Thus, for large T, the offset predominantly dictates the total torque command. As T decreases, tracking torque becomes dominant, and even a small adjustment in T can significantly alter the torque command. Hence, when tuning T, we recommend beginning with a large initial value and then reducing it incrementally until the desired rotational behavior is achieved, all while ensuring the feasibility of the thrusts.

At a given time t, we can express the reference state with a reference rotation vector $\boldsymbol{\Theta}_{ref}$, a reference angular velocity vector $\dot{\boldsymbol{\Theta}}_{ref}$, and a reference angular acceleration vector $\ddot{\boldsymbol{\Theta}}_{ref}$. All three vectors point along the rotation axis $\boldsymbol{s}_r$. Moreover, we can find the reference vehicle attitude at time t from the reference rotation vector as:

$$\boldsymbol{R}_{ref}(t) = \boldsymbol{R}_s f_{\boldsymbol{Rv}}(\boldsymbol{\Theta}_{ref}(t)), \tag{3.29}$$

where $\boldsymbol{R}_s$ is the attitude of the vehicle before the start of rotation, and $f_{\boldsymbol{Rv}}(\cdot)$ converts a rotation vector to its corresponding rotation matrix [66].

Tracking controller for re-orientation

To track the reference trajectory, we design a controller which generates a desired angular acceleration reducing the error as a second-order system:

$$\ddot{\boldsymbol{\Theta}}_d = \ddot{\boldsymbol{\Theta}}_{ref} + 2\zeta_r\omega_r(\dot{\boldsymbol{\Theta}}_{ref} - \hat{\boldsymbol{\omega}}) + \omega_r^2(\boldsymbol{\delta}_r) \tag{3.30}$$

where ζ_r is the desired damping ratio, ω_r is the desired natural frequency of the rotation and $\hat{\boldsymbol{\omega}}$ is the angular velocity reading from the rate gyroscope. The attitude error, $\boldsymbol{\delta}_r$, represented as a rotation vector in the body-fixed frame, can be computed as:

$$\boldsymbol{\delta_r} = f_{\boldsymbol{vR}}(\boldsymbol{R}^{-1}\boldsymbol{R}_{ref}) \tag{3.31}$$

where $f_{\boldsymbol{vR}}(\cdot)$ is the inverse of $f_{\boldsymbol{Rv}}(\cdot)$ and it converts a rotation matrix to a rotation vector.

The total desired torque command to track the trajectory can then be computed as the sum of the torque needed to offset gravity and the torque required to track the trajectory:

$$\boldsymbol{\tau}_d = \boldsymbol{J}_r\ddot{\boldsymbol{\Theta}}_d + \ \boldsymbol{S}(\hat{\boldsymbol{\omega}})\,\boldsymbol{J}_r\hat{\boldsymbol{\omega}} - \boldsymbol{\tau}_g \tag{3.32}$$

where $\boldsymbol{J}_r$ is the mass moment of inertia of the tensegrity aerial vehicle with respect to the rotation point. $\boldsymbol{\tau}_g$ is the gravitational torque offset and can be computed as the cross product between the vector pointing from rotation point to the center of mass and the gravitational vector.

Next, we convert the torque command to per-propeller thrust commands that the vehicle can directly implement. The mapping from the thrusts to the generated torque is:

$$\boldsymbol{\tau}_p = \sum_{i=1}^{4} f_i \boldsymbol{m}_{r,i} \tag{3.33}$$

Notice torque $\boldsymbol{\tau}_p \in \mathbb{R}^3$. Hence, Eq. (3.33) forms a linear system with three equations and four unknown thrusts, leading to an under-determined mapping from torque to thrust. Moreover, due to the physical limits of the motors and the propellers, we also need to take thrust saturation into account. To find the thrust commands, we solve the following problems:

When no thrusts are saturated and the exact desired torque can be generated, we command thrusts that minimize the sum of squares of the thrusts:

$$\begin{aligned} &\min_{f_i} \sum_{i=1}^{4} f_i^2 \\ &\text{s.t. } \boldsymbol{\tau}_d = \sum_{i=1}^{4} f_i \boldsymbol{m}_{r,i} \\ &\qquad f_{min} \le f_i \le f_{max} \end{aligned} \tag{3.34}$$

If problem (3.34) has no feasible solution, force saturation is unavoidable. This usually happens when the controller tries to correct a larger-than-expected tracking error. In these cases, we solve for a set of feasible thrusts that minimize the norm of the error between the desired torque and the torque that can be generated, while taking into account the constraint

of thrust generation authority:

$$\begin{aligned} &\min_{f_i} || \sum_{i=1}^{4} f_i \boldsymbol{m}_{r,i} - \boldsymbol{\tau}_d ||_2 \\ &\text{s.t. } f_{min} \leq f_i \leq f_{max} \end{aligned} \tag{3.35}$$

Both optimization problems (3.34) and (3.35) are of relatively low dimension. Consequently, they can be solved in real-time, even on embedded systems with limited computation power, using tools like CVXGEN [67]. A figure illustrating the thrust mapping for an example rotation problem and demonstrating the advantage of the optimization-based thrust converter introduced above can be found in Section 3.4.

3.4 Validation with experimental vehicle

In this section, we present the experimental tensegrity aerial vehicle, and the tests and analysis demonstrating its abilities. Video clips of all experiments are available online (see link in Section 3.1).

Experimental vehicle

An experimental vehicle (the right vehicle in Fig. 3.1) is designed and built to validate the proposed vehicle functionalities. We have designed the tensegrity shell with rods of 20mm length, so the shell can fit micro-scale quadcopter parts inside while still able to pass through narrow gaps between obstacles. Our design aims at providing protection against collisions at a target operation speed of 6m/s. We employ the design methodology described in Section 3.2 to identify suitable candidate materials for constructing the tensegrity shell. Among all candidates that satisfy the design requirements, carbon fiber rods with 6mm outer diameters and braided nylon strings have been selected based on factors such as weight, cost, and availability. The vehicle weighs 300g. Each motor can generate a maximum thrust of 2.8N for a short time (for re-orientation) or 2.2N continuously (for flight). The thrust-to-weight ratio for flight is about 3:1. The mass breakdown of the vehicle is as follows.

Shell	Batteries	Electronics	Motors
95g	75g	50g	80g

The resulting vehicle does not have a flat frame commonly used in quadcopter designs. Instead, its motors and computation units are directly mounted to the tensegrity shell using custom-designed 3D-printed mounts. Furthermore, to ensure even weight distribution, the design is powered by two batteries connected in series. Each battery is attached to one of the horizontal rods of the shell (see Fig. 3.8a). The design uses tension hooks to adjust self-stress in the tensegrity (see Fig. 3.8b) and 3D-printed end caps with fiber-glass infill to secure connections between rods and strings (see Fig. 3.8c).

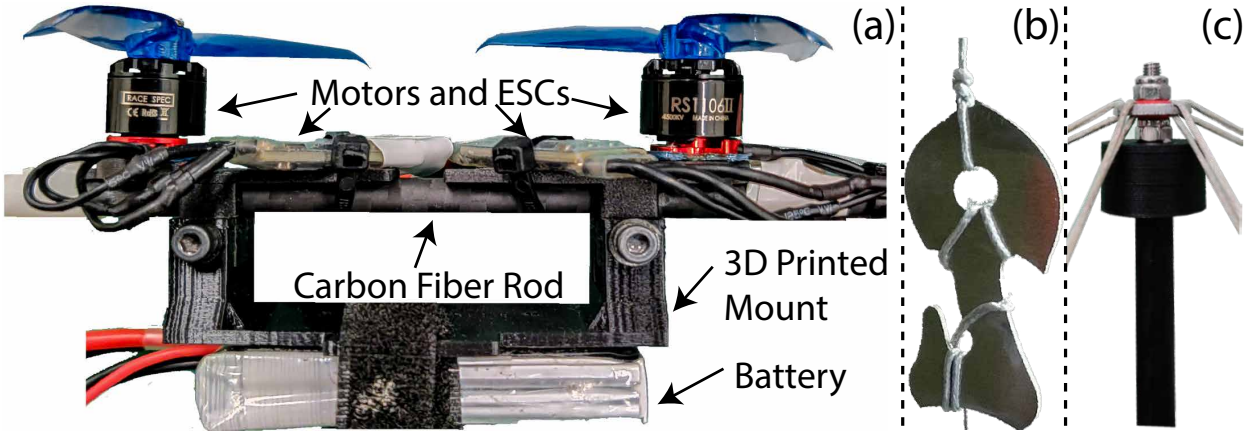

Figure 3.8: (a) Side-view of the tensegrity vehicle. Each horizontal rod has a battery and a pair of motors and ESCs attached to it. (b) Tension hook. (c) End cap connecting a rod to strings.

Collision resilience

A drop experiment and an in-flight collision experiment are conducted to verify the collision resilience of the experimental vehicle. In the drop experiment, the vehicle is dropped to a concrete pavement to find the collision speed it can survive. The vehicle successfully survives a drop from a 7m tall balcony with a landing speed of 11.7m/s. When we drop the vehicle from the next available balcony of 10.5m with a landing speed of 14.4m/s, the tensegrity fails with a string getting snapped. For comparison, a 250g quadcopter built with a commercial propeller-guarded frame [68] hosting the same propellers and electronics as the experimental vehicle, fractures after a 3.25m drop with a 8.0m/s landing speed. A composite image showing both drop experiments is shown in Fig. 3.9.

In addition to the drop test, we also control the experimental vehicle to accelerate towards a concrete wall and collide with it to confirm the vehicle's ability to survive collision during an actual flight. An image sequence from a high-speed video of the collision process is displayed in Fig. 3.10. The vehicle survives a collision of 7.8m/s, the fastest flying speed it can reach under the space limit of our flight space. All components within the tensegrity structure remain intact throughout the process, and the vehicle retain its ability to fly post-collision.

Re-orientation

We implement and test our re-orientation strategy proposed in Section 3.3. When generating the re-orientation paths, we assume friction coefficient $\mu = 0.2$, accounting for friction between the vehicle and slippery surfaces like wooden floors. The vehicle's ESCs are configured for bi-directional motor operation, thus providing increased torque generation authority

Figure 3.9: Composite image of the drop experiments validating the collision resilience of the experimental vehicle. Left: the experimental vehicle survives a 7m drop with a landing speed of 11.7m/s. Right: for comparison, the quadcopter hosting same electronics and propellers, built with commercial propeller guard, fractures after a 3.25m drop with a landing speed of 8m/s. The fracture is highlighted with the yellow box.

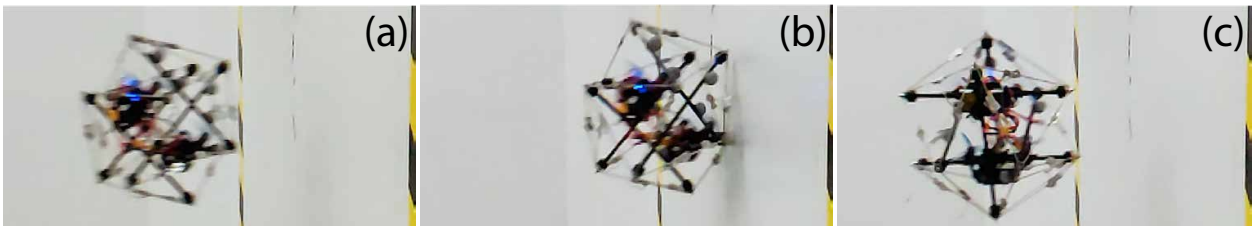

Figure 3.10: Sequence of images showing the process of a collision against a concrete wall: (a) Vehicle accelerates towards the wall. (b) Vehicle comes to a full stop. (c) Vehicle bounces back from the wall. The collision speed is 7.8m/s.

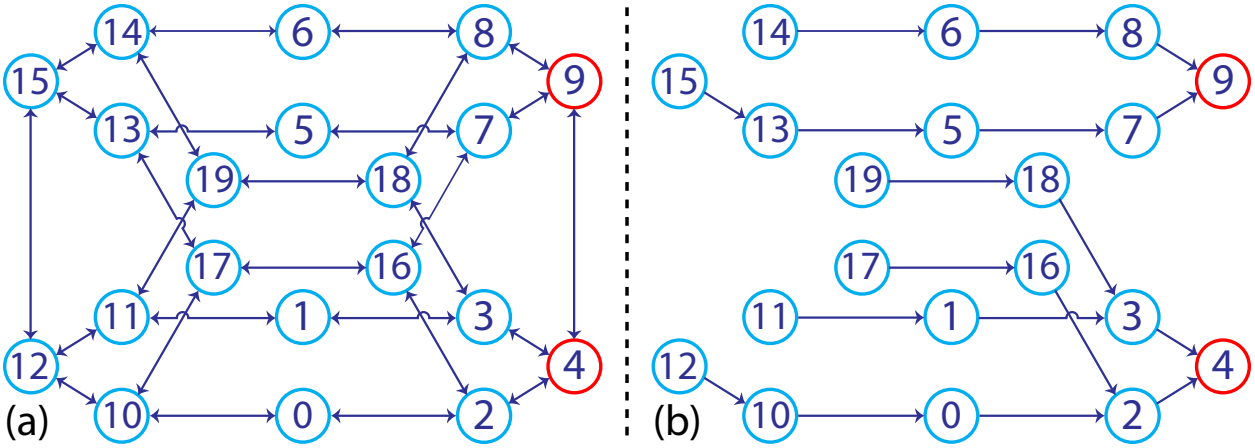

Figure 3.11: Generation of reorientation path. Nodes represent contact faces, with red denoting the goal faces to rotate towards. (a) A graph of all feasible face rotations is generated. Arrows indicate feasible rotations. (b) The shortest paths for each face to rotate to its closest goal face are generated, with arrows showing rotation directions.

during re-orientation. Fig. 3.11a illustrates feasible rotations between adjacent faces, while Fig. 3.11b presents the generated re-orientation paths. Notice that the vehicle can re-orient from any start face to the desired goal faces.

We also investigate the advantage of relaxing the constraint of sum of thrusts being zero in [27] via computing the additional payload that can be added to the center of mass before the re-orientation paths fail. We solve the optimization problems maximizing the vehicle mass under constraints (3.22) to (3.26) for all neighboring rotations. When an additional 0-thrust-sum constraint is imposed, the re-orientation paths will fail with an additional 12g mass. However, without this constraint, the vehicle can re-orient from all faces with an additional mass up to 30g.

In addition, we analyze the advantage of the optimization-based torque-thrust converter

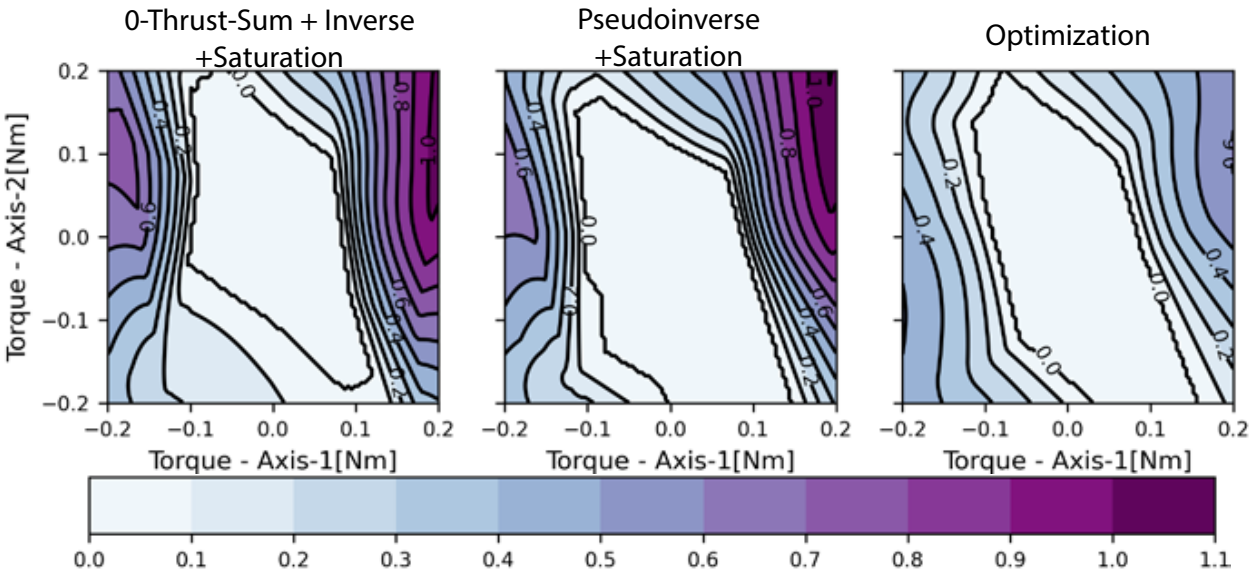

Figure 3.12: Thrust conversion error rates for rotation from face 3 to 4 with three different methods. Axis-1 is the rotation axis, and axis-2 points from the rotation point to the origin of the body-fixed frame. Darker color represents larger error rate and is undesirable.

in Section 3.3. When thrust saturation occurs, the generated torque will deviate from the command. The rate of this error, defined as the ratio between the error's norm and the command's norm, is used to gauge the effectiveness of thrust conversion. To demonstrate the advantage of our method, we use the rotation from face 3 to face 4 as an example, as it requires a torque that rolls, pitches, and yaws the vehicle simultaneously. We compare the error rates of three methods, as shown in Fig. 3.12: 1) Adding an additional 0-thrust-sum constraint to the under-determined linear system Eq. (3.33) to make it fully-determined, solving the combined linear system for desired thrusts, and saturating the thrusts based on the feasible range. 2) Computing desired thrusts by solving Eq. (3.33) with the pseudoinverse method, and then saturating the thrusts. 3) Computing the thrust commands by directly solving the optimization problems in (3.34) and (3.35). The figure shows that the optimization method has the largest error-free region. Furthermore, in scenarios with thrust saturation, the optimization-based method shows a much smaller error rate. This suggests that the optimization-based torque-thrust converter improves the vehicle's ability to implement re-orientation rotations.

With the planned re-orientation paths and the optimization-based thrust converter, the experimental vehicle can reliably re-orient and take off. Videos demonstrating successful re-orientations, including a scenario overcoming an initial failure caused by an external disturbance can be accessed through the link in Section 3.1.

Autonomous operation in forest environment

In this subsection, we present an experiment demonstrating the experimental vehicle's ability to autonomously operate in a cluttered environment. The vehicle is directed towards a goal in a forest previously unknown to the vehicle, which contains tree obstacles and uneven terrain. We employ the Extended Kalman Filter (EKF) from [69] to estimate the vehicle's state, including position, velocity, and attitude. Given the absence of external aids like GPS or motion capture, we employ the estimation strategy from [58], which improves estimation accuracy by breaking a long flight into short hops and updating the EKF with pseudo zero-velocity measurements when vehicle sensors indicate a stationary status. Upon detecting a collision (the norm of accelerometer readings exceeds a threshold), the vehicle seeks to stabilize itself and land softly. After landing, it re-orients and attempts to hop around the obstacle it has collided with.

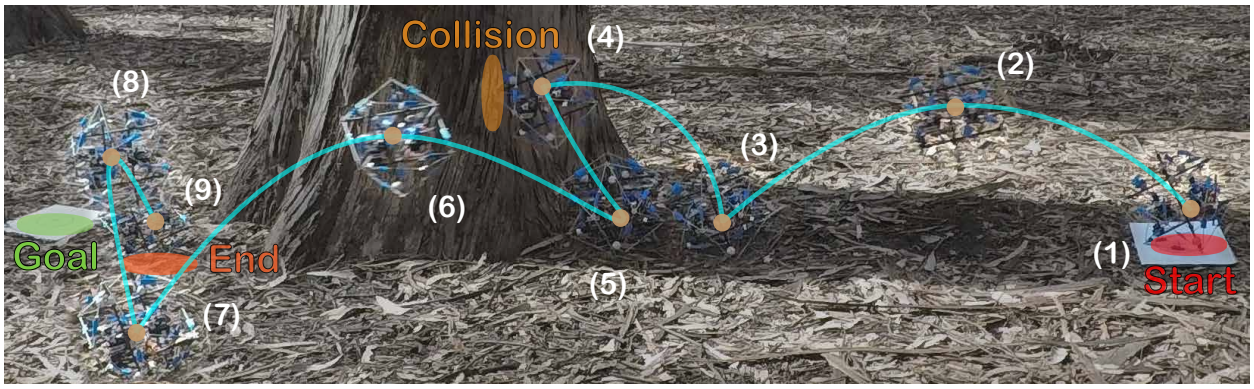

Figure 3.13: Composite image of the tensegrity aerial vehicle autonomously operating in a previously unknown forest environment. The cyan curve marks the movement of the vehicle. The vehicle is ordered to move from the start point on the right side of the figure to the goal point on the left. A tree obstacle exists between the two points. The vehicle successfully survives a collision with the tree and arrives at an endpoint close to the goal. The distance between the goal point and the end point is 0.25m. The background is desaturated to highlight the vehicle movement.

The outdoor environment experiment reveals certain limitations of the vehicle. Navigation accuracy is restricted by the inertial sensors' accuracy and range. High-impact collisions can cause sensor saturation, introducing significant error into the state estimator. Also, the re-orientation controller's performance can be hindered by torque limitations, particularly when the vehicle lands on steep slopes or is trapped by large ground indentations. To mitigate these issues, we lower the hop speed to avoid high-velocity collisions and instruct the vehicle to attempt backward hops when trapped.

Fig. 3.13 presents a composite image from the test. The vehicle is tasked to travel 3m in a specified direction with an unforeseen tree obstacle en route. During its second hop, the

vehicle collides with the tree, manages to survive the impact, logs the obstacle's position, executes a sideways hop to evade the obstacle, and proceeds towards the goal.

3.5 Conclusion

In this chapter, we introduce the tensegrity aerial vehicle, a collision-resilient flying robot design with an icosahedron tensegrity structure. We establish an approach for predicting structural stresses during collisions via a dynamics simulation, which facilitates component selection during the design process. This approach contributes to the successful creation of an experimental vehicle with strong collision resilience, capable of surviving a 7m drop with a 11.7m/s landing speed. Additionally, we develop a re-orientation controller, enabling the vehicle to take off post-collision. This combination of collision resilience and post-collision flight resumption makes the tensegrity aerial vehicle ideally suited for field operations in cluttered environments filled with obstacles.

Chapter 4

Exploiting collisions for quadcopter motion planning in cluttered environments

Aerial vehicles with collision-resilient designs, such as the tensegrity aerial vehicle discussed in Chapter 3, have brought new opportunities for motion planning in cluttered environments. Aerial vehicles can leverage collisions to instantaneously change their movement directions, thereby circumventing aggressive maneuvers required for rapid turning. In this chapter, we present a sampling-based collision-inclusive motion planning algorithm for quadcopters. This algorithm extends from the classic Optimal Rapidly-Exploring Random Tree (RRT*) [37] structure with following key modifications. First, it connects sampled states with quadcopter primitives, thereby facilitating efficient candidate trajectory generation and enabling rapid checks for command feasibility. Second, it samples candidate collision states by evaluating potential intersections between motion primitives and obstacles, and therefore incorporating collisions as integral components of the generated candidate trajectories.

In addition to harnessing collisions for rapid direction changes, the integration of collisions into the sampling-based motion planning offers another notable advantage: it can help the planner find feasible trajectories in confined spaces, such as tunnels, with less computation time. In such scenarios, allowing for collision expedites the random tree growth, as samples are no longer discarded due to collisions, leading to more exploring efficiency.

The planning algorithm introduced in this chapter builds upon two insightful research in the field of quadcopter motion planning, which we recommend to interested readers. The first is the minimum-jerk quadcopter motion primitive generator, proposed in:

- Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," in *IEEE Transactions on Robotics*, IEEE, 31, no. 6 (2015): 1294-1310. [33]

This tool allows us to quickly generate quadcopter primitives between sampled states and check their feasibility.

The second is the rapid collision detection algorithm for polynomial quadcopter motion primitives:

- Nathan Bucki, and Mark W. Mueller, "Rapid collision detection for multicopter trajectories," in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, 2019, pp. 7234-7239. [34]

This tool allows us to efficiently check if a minimum jerk motion primitive will collide with a convex obstacle. We update the algorithm to predict the specific time and quadcopter state at the collision.

It is noteworthy that the algorithm proposed in this chapter is specifically tailored for quadcopter motion planning. By specializing the algorithm for quadcopters and exploiting the differential flatness of quadcopter dynamics, we achieve exceptional planning speed, a trade-off we believe to be worthwhile given the widespread use of quadcopters and the critical role of timeliness in UAV navigation.

The material in this chapter is based on the following previously published work:

- Jiaming Zha and Mark W. Mueller, "Exploiting collisions for sampling-based multicopter motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7943-7949.

4.1 Introduction

Our research on collision-resilient tensegrity aerial vehicles, discussed in Chapter 3, opens up new possibilities for physical contact with surrounding objects during operation. This development prompts us to harness this unique capability to create trajectories within an expanded, collision-inclusive state space. The method we propose in this chapter takes a sampling-based approach to incorporate collisions into quadcopter motion planning.

Our method, based on the RRT* algorithm [37], exploits the differential flatness inherent in quadcopter dynamics. This enables swift generation of minimum jerk motion primitives and efficient collision detection. A literature review on quadcopter motion primitives, sampling-based planning structures, and planning with collisions can be found in Chapter 2. By identifying collisions and predicting post-collision vehicle states, our method can generate collision states and potentially connect them with other sampled states to form collision-inclusive trajectories.

Planning trajectories with collisions presents an interesting trade-off. On one hand, more computation time is required to detect and generate states involving collisions. On the other hand, the planner will no longer discard samples because of collisions, enabling more efficient exploration of the random tree. We find that the benefit of planning with collision is likely to outweigh its computation cost in narrow environments such as tunnels, where collision is likely to take place and collision-exclusive planners are forced to discard most of their samples. Furthermore, both simulations and experiments have shown that collisions can

assist aerial vehicles in rapidly changing their direction of movement, thereby eliminating the need for aggressive maneuvers.

The remainder of this chapter is organized as follows: In Section 4.2, we introduce the quadcopter minimum jerk primitive and rapid collision detection algorithm for these primitives, two vital components of our motion planner. Section 4.3 details the collision-inclusive motion planning algorithm itself. A Monte Carlo study demonstrating the algorithm's advantages is then presented in Section 4.4. Section 4.5 presents the experimental validation of tracking the planned trajectory with our tensegrity aerial vehicle. We then conclude the chapter with Section 4.6.

4.2 Quadcopter motion primitive and collision detector

This section introduces two important building blocks of the collision-inclusive motion planner, the motion primitive generator for quadcopters [33] and the rapid collision detector [34] for such primitives. Here we only introduce the key results and explain how they are adapted to work with our collision-inclusive motion planner.

Quadcopter motion primitive generator

The quadcopter motion primitive generator is a computationally efficient tool that can generate and check the input-feasibility of about one million motion primitives in a second on a modern computer. This enables us to rapidly connect sampled states to form feasible trajectories.

The generator computes a thrice differentiable trajectory which guides the quadcopter from an initial state at time t_0 to a final state at time t_f, while minimizing the cost function

$$J = \int_{t_0}^{t_f} ||\boldsymbol{j}(t)||^2 dt \tag{4.1}$$

where $\boldsymbol{j}(t)$ is the jerk of the quadcopter at time t and is defined as the third order derivative of the position. The cost J can be interpreted as the upper bound of the product of the norm of inputs (in terms of total thrust and angular velocity) to the quadcopter system (see detailed derivation in [33]). Hence, a trajectory with lower cost tends to be less aggressive and is more likely to be input-feasible.

The quadcopter motion primitive generated is a fifth order polynomial:

$$\boldsymbol{x}(t) = \boldsymbol{a}_0 t^5 + \boldsymbol{a}_1 t^4 + \boldsymbol{a}_2 t^3 + \ddot{\boldsymbol{x}}_0 t^2 + \dot{\boldsymbol{x}}_0 t + \boldsymbol{x_0} \tag{4.2}$$

where $\boldsymbol{x}(t)$ is the position at time t and $\boldsymbol{x}_0$, $\dot{\boldsymbol{x}}_0$, $\ddot{\boldsymbol{x}}_0$ are position, velocity and acceleration at the start of motion primitive. Vector parameters $\boldsymbol{a}_0$, $\boldsymbol{a}_1$ $\boldsymbol{a}_2 \in \mathbb{R}^3$ describe the trajectory, and are solved as linear functions of the initial state and the final state.

Let $\mathcal{P} = \text{MotionPrimitive}((\boldsymbol{s}_0, t_0), (\boldsymbol{s}_f, t_f))$ be the function generating the motion primitive connecting state $\boldsymbol{s}_0$ and $\boldsymbol{s}_f$ between time t_0 and t_f. A state is defined as the combination of position, velocity and acceleration of the vehicle. We define $\boldsymbol{C}(\mathcal{P})$ as the function evaluating the cost of the motion primitive with Eq. (4.1). For each generated motion primitive, we can check if the inputs for the quadcopter to implement the primitive satisfy bounds on the minimum and maximum total thrust and the magnitude of the angular velocity. We define $\text{InputFeasible}(\mathcal{P})$ as the function checking the input feasibility of the motion primitive. A method for quickly implementing this check is detailed in [33].

Rapid Collision Detection

The rapid collision detection algorithm [34] checks if a generated motion primitive will collide with convex obstacles in the environment. Non-convex obstacles may be approximated by defining them as a union of convex obstacles.

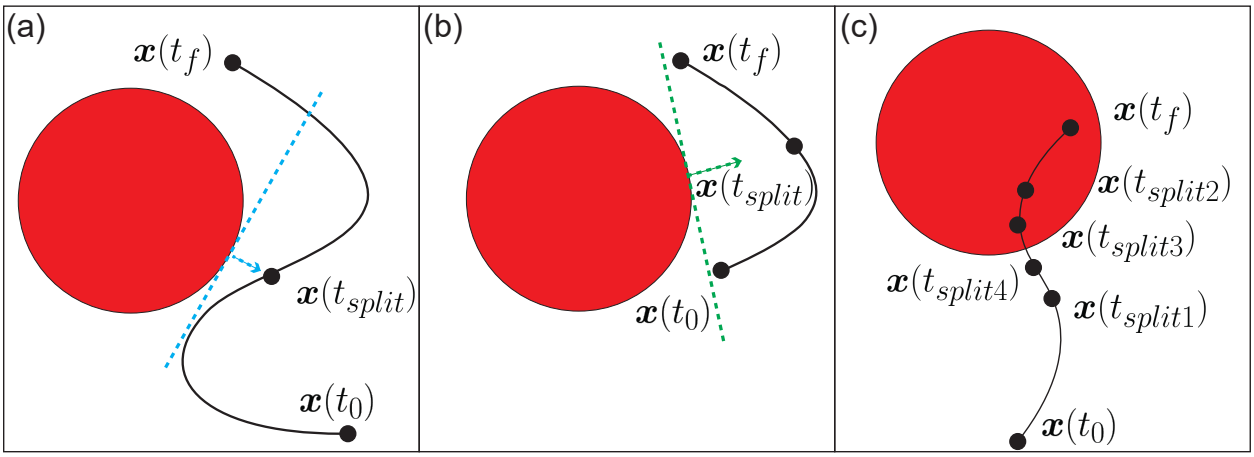

Figure 4.1: A graphic illustration of key ideas of the collision detection process. (a) The detector checks if the primitive stays on the same side of the separation plane. If not, it bisects the primitive and checks either half that crosses the plane. (b) If the primitive piece stays on the same side of the separation plane, we know the primitive is free of collision. (c) We can estimate the collision time by keep bisecting the primitive until the section crossing the separation plane is shorter than a certain duration threshold.

For a given motion primitive starting at time t_0 and ending at time t_f, the algorithm first checks the spatial relationship between the given obstacle and the begin position $\boldsymbol{x}(t_0)$, end position $\boldsymbol{x}(t_f)$ and the mid-time position $\boldsymbol{x}(t_{split})$ respectively. Here $t_{split} = (t_0 + t_f)/2$. If all three points are outside the obstacle, the algorithm then checks if the primitive crosses a separation plane between the obstacle and $\boldsymbol{x}(t_{split})$. This separation plane is defined as the

tangential plane of obstacle surface that includes a point $\boldsymbol{p}$, which is located in the obstacle and has minimum distance to $\boldsymbol{x}(t_{split})$.

If the whole primitive does not cross the separation plane, it is guaranteed to be free of collision. If the primitive crosses the separation plane, we can bisect it into two sections, $\boldsymbol{x}(t_0) \rightarrow \boldsymbol{x}(t_{split})$ and $\boldsymbol{x}(t_{split}) \rightarrow \boldsymbol{x}(t_f)$ and repeat the above checking process on the section(s) crossing the separation plane. An illustration of this idea is shown in Fig 4.1a and Fig 4.1b.

In [34], the recursion terminates once any part of the primitive is found lying inside the obstacle. We modify the recursion termination rule to compute the time of the collision. When checking for collision, we can keep bisecting the primitive until the time of the checked primitive section crossing the separation plane is smaller than a certain threshold. We can thus estimate the time of collision as the beginning time of this primitive section, as shown in Fig. 4.1c. We define CollisionFree($\mathcal{P}$) as the function checking if the primitive collides with any obstacle in the space. We also denote CollisionTime($\mathcal{P}$) as the function finding the time of the first collision between a given primitive and the obstacles in the environment.

4.3 Algorithm description

This section introduces the algorithm of the sampling-based method for collision-inclusive motion planning. The algorithm searches for a potentially collision-inclusive trajectory connecting the start state $\boldsymbol{s}_0$ and the goal state $\boldsymbol{s}_f$.

The algorithm is based on the RRT*, but with following differences. First, a collision state generation step is introduced to allow states involving collisions to be considered. Second, instead of connecting sampled states with lines, we connect them with motion primitives described in Section 4.2. Third, we pair each sampled state with a sampled time. The corresponding time difference between two states is the duration of the primitive connecting them, which determines the input-feasibility of the primitive. Within a given computation time limit, the algorithm keeps sampling and connecting state-time pairs to generate trajectories in the class of concatenated minimum jerk primitives. The fastest candidate among all generated trajectories connecting $\boldsymbol{s}_0$ and $\boldsymbol{s}_f$ is then returned as the best solution found by the planner. For the rest of this section, we will first introduce how key steps of the original RRT* algorithm are modified to find collision-inclusive trajectories for quadcopters. Then, we will present the whole planner and discuss the benefits and disadvantages of planning with collisions.

Collision model

We define the state of the vehicle by its position, velocity and acceleration, i.e. $\boldsymbol{s} = [\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}]$. Given a pre-collision state $\boldsymbol{s}$, the function CollisionModel($\boldsymbol{s}$) predicts the post collision state $\boldsymbol{s}^+ = [\boldsymbol{x}^+, \dot{\boldsymbol{x}}^+, \ddot{\boldsymbol{x}}^+]$. This model depends on the vehicle design and the material of contact surface and may vary among different quadcopters.

Connecting nodes as state-time pairs

We define a node $\boldsymbol{n}$ as a pair of vehicle state and end time, $\boldsymbol{n} = (\boldsymbol{s}, t)$. This indicates that it takes the vehicle time t to reach $\boldsymbol{s}$ from the start node $(\boldsymbol{s}_0, 0)$, where $\boldsymbol{s}_0$ is the start state of the planning problem. When a node represents a state at collision, it will be coupled with a post-collision node containing the same time and a post-collision state predicted by the collision model. Denote $\boldsymbol{PC}(\boldsymbol{n})$ as the function accessing the post-collision node of $\boldsymbol{n}$. Define CONNECT$(\boldsymbol{n}_1, \boldsymbol{n}_2)$ as the process of generating the motion primitive connecting two nodes, as shown in Algorithm 1. Moreover, we define a path between two nodes as a set of primitives connecting the nodes and the cost of the path as the sum of the costs of the primitives the path contains. We further define the cost of node, COST$(\boldsymbol{n})$ as the lowest cost of the feasible path found connecting the start node to $\boldsymbol{n}$.

Algorithm 1 Connect nodes

```
1: function CONNECT(n_1, n_2)
2:     assert time of n_1 is before time of n_2
3:     if n_1 is a collision node then
4:         return MOTIONPRIMITIVE(PC(n_1), n_2)
5:     else
6:         return MOTIONPRIMITIVE(n_1, n_2)
```

Generating sample states

For each step, the planner generates a node sample $\boldsymbol{n}_s$ via a random process. With possibility η_f, the goal sampling rate, the state of $\boldsymbol{n}_s$ is set as the goal state and with possibility (1- η_f), the state of $\boldsymbol{n}_s$ is uniformly sampled from the state space. The time t of $\boldsymbol{n}_s$ is sampled uniformly from $[0, t^*_{end}]$ where t^*_{end} is the time of the shortest feasible trajectory from the start state to the goal state the algorithm has found. This value is initiated with an overestimate of the shortest feasible trajectory time and then updated throughout the planning. We use function SAMPLE() to denote the above process.

Collision node generation

After the sampled node is generated, we search for a node in $\mathcal{T}$ whose time is before the sampled node and can be connected to it with a primitive of the lowest cost. Define CLOSESTNODE$(\mathcal{T}, \boldsymbol{n}_s)$ as the function of finding such node.

Then, we check if the primitive connecting the closest node to the sampled node collides with any obstacle in the environment. If the primitive is free of collision, we use the sampled node directly for future optimal connection attempts. However, if the primitive collides with any obstacle in the environment, we generate a collision node with the time and the state of the vehicle right before the collision. Meanwhile, we use the collision model to predict the

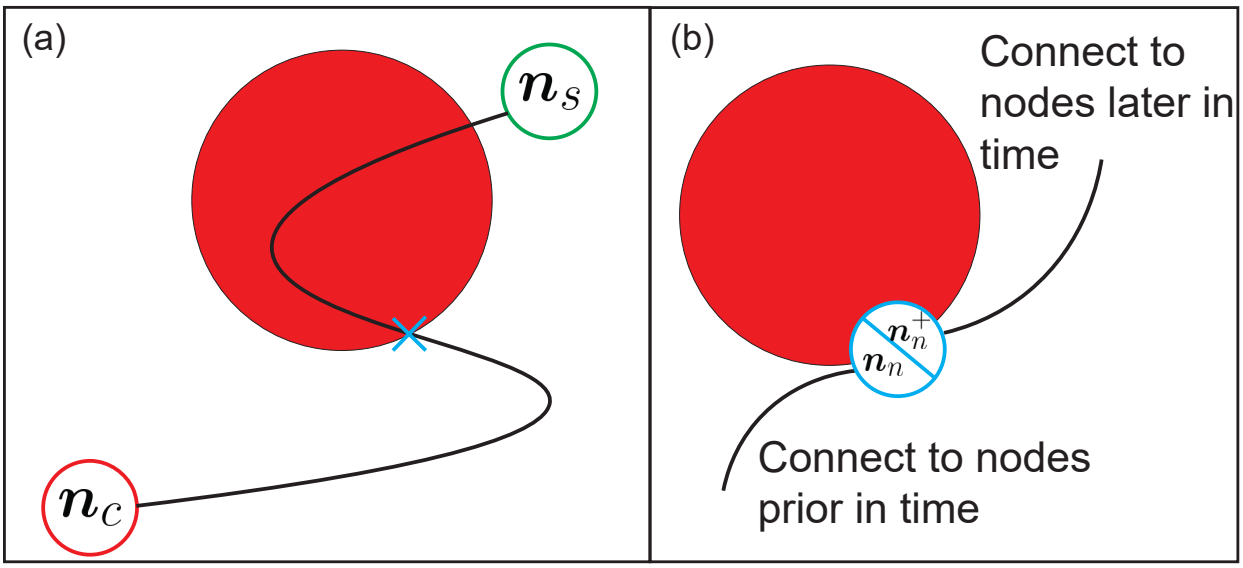

Figure 4.2: A graphic illustration of the collision node generation process. After node $\boldsymbol{n}_s$ is sampled, we connect it with its closest node $\boldsymbol{n}_c$. If a collision takes place, we generate a pre-collision node with the time and the state right before the collision. We also predict its state after the collision and set it as the post-collision node for possible future connections.

Algorithm 2 Collision node generation

function GetCollisionNode($\boldsymbol{n}_s$)
  $\boldsymbol{n}_c \leftarrow$ ClosestNode($\mathcal{T}, \boldsymbol{n}_s$)
  $\mathcal{P} \leftarrow$ Connect($\boldsymbol{n}_c, \boldsymbol{n}_s$)
  if CollisionFree($\mathcal{P}$) **then** $\boldsymbol{n}_n \leftarrow \boldsymbol{n}_s$
  else
    $t_n \leftarrow$ CollisionTime($\mathcal{P}$)
    $\boldsymbol{s}_n \leftarrow \mathcal{P}(t_n)$
    $\boldsymbol{n}_n \leftarrow (\boldsymbol{s}_n, t_n)$
    $\boldsymbol{PC}(\boldsymbol{n}_n) \leftarrow$ (CollisionModel($\boldsymbol{s}_n$), t_n)
  return $\boldsymbol{n}_n$

post-collision state. This process is illustrated in Fig. 4.2 and we define the process as the GetCollisionNode($\boldsymbol{n}_s$) function, with its implementation detailed in Algorithm 2.

Connect along minimum cost path and rewire

After sampling and collision node generation, we want to connect the generated node to a best feasible parent node in $\mathcal{T}$ so that its cost is minimized. If such a feasible parent can

be found, we will add the generated node to $\mathcal{T}$. Afterwards, we rewire the tree to ensure the optimality of all connections.

Algorithm 3 Connect along minimum cost path and rewire

function ConnectMinCostPath($\boldsymbol{n}_n$)
  Initialize an empty heap $\mathcal{H}$.
  $k \leftarrow 2e \cdot log(|\mathcal{T}|)$
  Parent($\boldsymbol{n}_n$) $\leftarrow$ *null*
  Cost($\boldsymbol{n}_n$) $\leftarrow \infty$
  for $\boldsymbol{n}$ in $\mathcal{T}$ with time prior to $\boldsymbol{n}_n$ **do**
    $\mathcal{P} \leftarrow$ Connect($\boldsymbol{n}, \boldsymbol{n}_n$)
    if $\boldsymbol{C}(\mathcal{P}) > max(\mathcal{H})$ **then continue**
    if not InputFeasible($\mathcal{P}$) **then continue**
    if not CollisionFree($\mathcal{P}$) **then continue**
    if Cost($\boldsymbol{n}$) + $\boldsymbol{C}(\mathcal{P})$ < Cost($\boldsymbol{n}_n$) **then**
      Cost($\boldsymbol{n}_n$) $\leftarrow$ Cost($\boldsymbol{n}$)+ $\boldsymbol{C}(\mathcal{P})$
      Parent($\boldsymbol{n}_n$) $\leftarrow \boldsymbol{n}$
    Push $\boldsymbol{C}(\mathcal{P})$ into $\mathcal{H}$
    if $|\mathcal{H}| > k$ **then** pop the largest value in $\mathcal{H}$
  if Parent($\boldsymbol{n}_n$) is not *null* **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\boldsymbol{n}_n\}$
  return
function Rewire($\boldsymbol{n}_n$)
  Initialize an empty heap $\mathcal{H}$.
  $k \leftarrow 2e \cdot log(|\mathcal{T}|)$
  for $\boldsymbol{n}$ in $\mathcal{T}$ with time after $\boldsymbol{n}_n$ **do**
    $\mathcal{P} \leftarrow$ Connect($\boldsymbol{n}_n, \boldsymbol{n}$)
    if $\boldsymbol{C}(\mathcal{P}) > max(\mathcal{H})$ **then continue**
    if not InputFeasible($\mathcal{P}$) **then continue**
    if not CollisionFree($\mathcal{P}$) **then continue**
    if Cost($\boldsymbol{n}_n$) + $\boldsymbol{C}(\mathcal{P})$ < Cost($\boldsymbol{n}$) **then**
      Cost($\boldsymbol{n}$) $\leftarrow$ Cost($\boldsymbol{n}_n$)+ $\boldsymbol{C}(\mathcal{P})$
      Parent($\boldsymbol{n}$) $\leftarrow \boldsymbol{n}_n$
      Update the cost of descendants of $\boldsymbol{n}$
    Push $\boldsymbol{C}(\mathcal{P})$ into $\mathcal{H}$
    if $|\mathcal{H}| > k$ **then** pop largest value in $\mathcal{H}$
  return

Notice that comparing to the original RRT*, we are not generating a "near neighbor" set to decrease the number of connections, because the metric of "distance" between two nodes

is the cost of the primitive connecting them and such cost cannot be calculated before the connection attempt. However, for planning tasks that involve a large number of nodes, we can use a heap, $\mathcal{H}$ to track k-smallest costs of feasible primitives found so far. Following the suggestion of [37], we set $k = 2e \cdot log(|\mathcal{T}|)$, where $|\mathcal{T}|$ is the cardinality of $\mathcal{T}$. If a primitive candidate has a cost exceeding the largest value in the heap, it will be discarded without the feasibility check. This pre-screening helps decrease computation time and has an effect similar to the "near neighbor" screening in the original RRT*. The whole process of connecting and rewiring the exploring tree is shown in Algorithm 3.

Full collision-inclusive sampling-based planner

Now, we combine the previous parts and present the full planner as Algorithm 4. After running the planner, we find the best end node as the node in $\mathcal{T}$ with an end state that has a smallest end time. After identifying the best end node, we can recover the best trajectory via backtracking from the best end node to the start node.

Algorithm 4 Collision-inclusive sampling-based planner

input: Start state $\boldsymbol{s}_0$, goal state $\boldsymbol{s}_f$, set of obstacles $\mathcal{O}$, state space $\boldsymbol{S}$, goal-sampling rate η_f
$\boldsymbol{n}_0 \leftarrow (\boldsymbol{s}_0, 0)$
PARENT$(\boldsymbol{n}_0) \leftarrow null$
COST$(\boldsymbol{n}_0) \leftarrow 0$
$\mathcal{T} \leftarrow \{\boldsymbol{n}_0\}$
while computation time $<$ planning time limit **do**
    $\boldsymbol{n}_s \leftarrow$ SAMPLE()
    if $\boldsymbol{n}_s$ is not a goal node **then**
        $\boldsymbol{n}_n \leftarrow$ GETCOLLISIONNODE$(\boldsymbol{n}_s)$
    else
        $\boldsymbol{n}_n \leftarrow \boldsymbol{n}_s$
    CONNECTMINCOSTPATH$(\boldsymbol{n}_n)$
    if $\boldsymbol{n}_n$ is added to $\mathcal{T}$ **then** REWIRE$(\boldsymbol{n}_n)$

Planning with collision brings two benefits. First, allowing collisions extends the feasible state space that the sampling-based planner can search in. Second, sampled nodes will not be discarded due to infeasibility caused by collisions. Hence, the collision-inclusive planner may add nodes to $\mathcal{T}$ more efficiently.

However, planning with collision also comes with disadvantages. First, the process of generating collision nodes takes additional computation time. Second, due to the collision node generation process, sampled nodes may concentrate near obstacle surfaces facing the exploring tree. As a result, expansion of the exploring random tree towards the other side of the obstacle may be slowed down. An example is illustrated in Fig. 4.3. Nodes $\boldsymbol{n}_a$ and $\boldsymbol{n}_b$ are in $\mathcal{T}$ and a node $\boldsymbol{n}_s$ is sampled on the right side of the obstacle. In Fig. 4.3a, with

the collision node generation process, $\boldsymbol{n}_s$ will be connected with its closest node, $\boldsymbol{n}_a$ and a collision node is generated on the left side of the obstacle. If the process is disabled, as in Fig. 4.3b, the sampled node will be connected with $\boldsymbol{n}_b$ and added directly to $\mathcal{T}$. We see that with the collision node generation, $\mathcal{T}$ can no longer reach the right of obstacle at this step and its expansion is slowed down in this example.

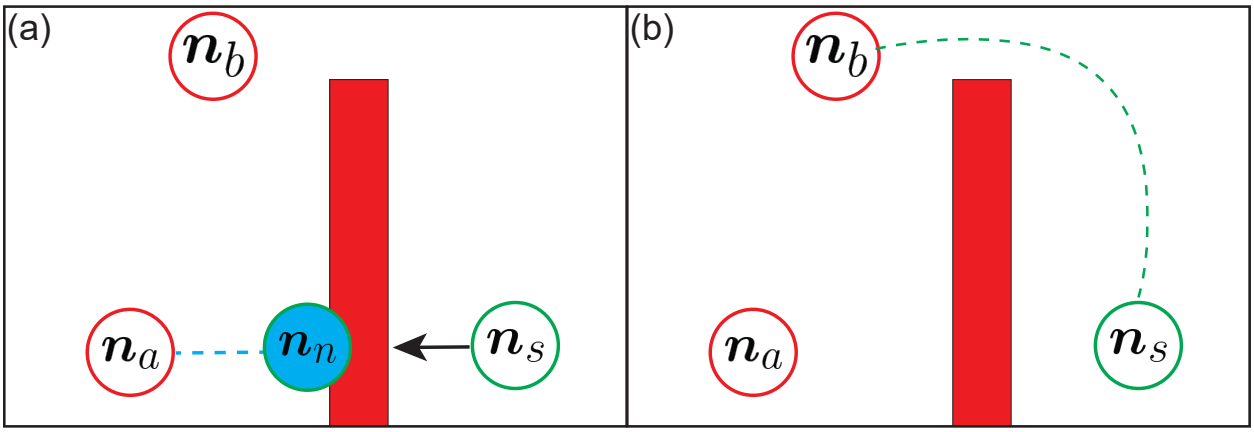

Figure 4.3: Illustration of how the collision node generation process can slow down the expansion of the tree. $\boldsymbol{n}_a$ and $\boldsymbol{n}_b$ are nodes in $\mathcal{T}$. The planner generates a sampled state $\boldsymbol{n}_s$ that can be connect to $\boldsymbol{n}_a$ with a lower cost. (a) During the collision node generation process, a collision node $\boldsymbol{n}_n$ will be generated on the left of the obstacle and added to $\mathcal{T}$. (b) If the collision node generation is disabled, $\boldsymbol{n}_s$ will be added to $\mathcal{T}$ directly. As a result, $\mathcal{T}$ can expand to the right of the obstacle in this step.

4.4 Illustrative example: motion planning in a tunnel

In this section, we present an illustrative example to showcase the trade-off between the cost and the benefits of planning with collisions and illustrate why the collision-inclusive planner may perform better than collision-exclusive planner in narrow spaces.

In this 2D example, the vehicle starts at position (1, 2)[m] and is in a 1m wide tunnel. It needs to move in the positive x-direction for 3.5 meters to leave the tunnel and then move in an open space to get to a goal position at (4, 5)[m]. We run both the collision-inclusive planner and the collision-exclusive planner on this problem for 10^5 times and compare the results generated.

A snapshot of the planning result after computing for 0.1s is shown in Fig. 4.4. We observe that the collision-inclusive planner generates a trajectory colliding with the tunnel wall. Moreover, its nodes in $\mathcal{T}$ are mainly collision nodes. This phenomenon is expected. The tunnel is very narrow, so most of the primitives generated involve collisions. Moreover, we

notice that the collision-exclusive planner generates more nodes in the open space right of the tunnel than the collision-inclusive planner, indicating a faster expansion of the exploring tree in that region. This echoes the discussion in Section 4.3 that the collision node generation process can sometimes slow down the spatial expansion of the exploring tree.

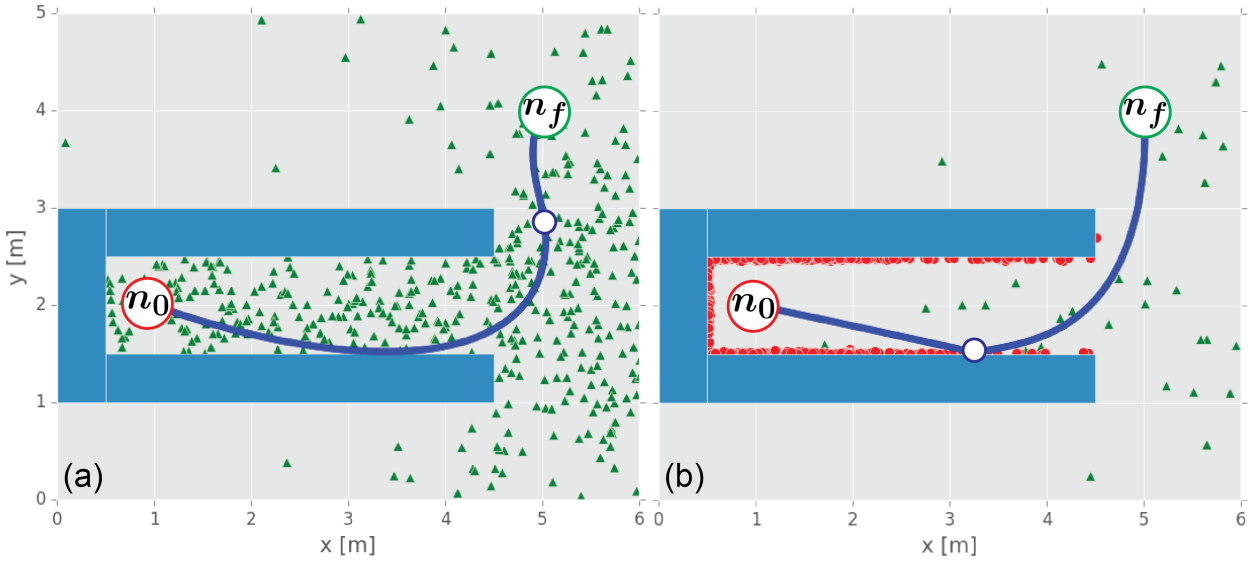

Figure 4.4: Snapshots after (a) collision-exclusive planner and (b) collision-inclusive planner running for 0.1s. Red dots are the feasible collision nodes in $\mathcal{T}$ and green triangles are the feasible non-collision nodes in $\mathcal{T}$. The blue curve is the feasible trajectory connecting $\boldsymbol{n}_0$ to $\boldsymbol{n}_f$ with the shortest time found. The blue circle on trajectory is an intermediate node.

Despite the slower expansion in the open space, the collision-inclusive planner still has a better overall planning performance. Here we compare the performance of the planners with the median of the minimum feasible trajectory time the planners have found among all trials. Median, instead of mean, is used to filter out the influence of outliers. Fig. 4.5a plots the median of the best trajectory time found versus computation time. It shows that the collision-inclusive planner finds a better trajectory than the collision-exclusive planner under the same time limit. The better performance can be credited to the two benefits of collision-inclusive planning. First, allowing collisions extends the feasible state space. Second, collision-inclusive planner can add nodes to $\mathcal{T}$ more efficiently, because no nodes are discarded due to infeasibility caused by collision. Fig. 4.5b shows the median of the numbers of nodes in $\mathcal{T}$ versus computation time. The figure shows that collision-inclusive planner can add nodes to $\mathcal{T}$ with a higher speed. The relatively low efficiency of the collision-exclusive planner suggests that most of its sampled nodes are deemed as infeasible due to collisions and cannot be added to $\mathcal{T}$. This example shows that the collision-inclusive planner is likely to outperform the collision-exclusive planner when the vehicle is in narrow spaces, which are

usually the environments that collision-resilient vehicles are designed to operate in.

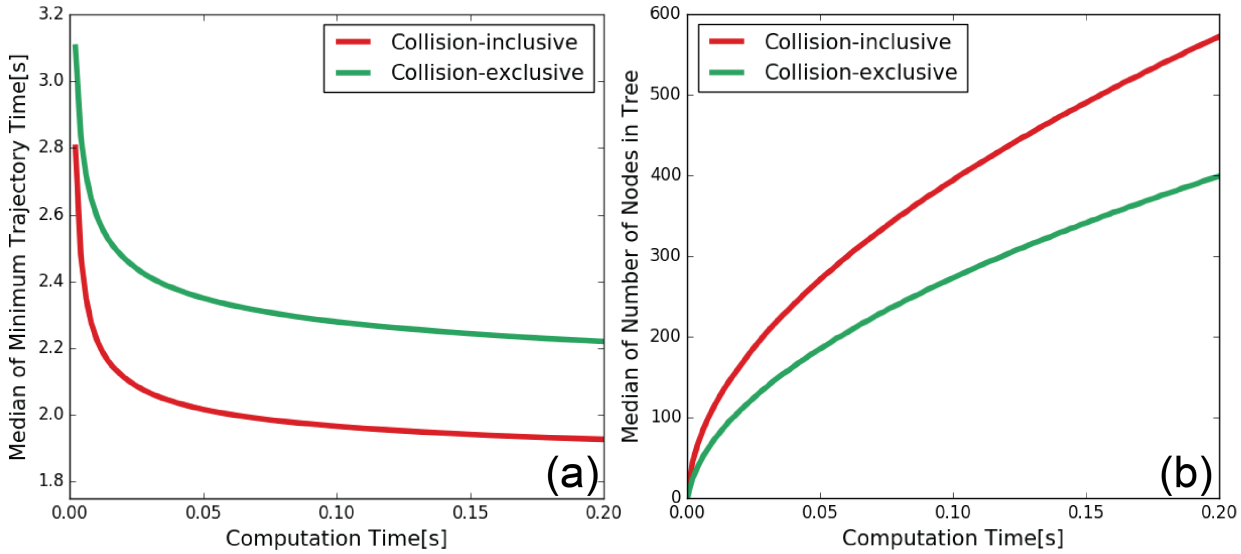

Figure 4.5: Comparison of the performance of planners after running the problem for 10^5 times. (a) Median of the minimum trajectory time vs computation time. (b) Median of the number of the nodes in $\mathcal{T}$ vs computation time.

4.5 Experimentally tracking planned trajectories

This section demonstrates the experiment of tracking planned collision-inclusive and collision-exclusive trajectories generated by our planner with a collision-resilient tensegrity aerial vehicle [27]. Link to the experiment videos: **youtu.be/MmDHra3wYK4**.

Collision model for the test platform

We predict that the post-collision position stays the same as the pre-collision position, $\boldsymbol{x}^+ = \boldsymbol{x}$. For post collision velocity, we use an empirical model similar to the one in [70] for the non-sliding case. The model predicts the velocity component normal to the obstacle with a linear function:

$$\dot{x}_n^+ = -e\dot{x}_n \tag{4.3}$$

Where e is the coefficient of restitution and $\dot{x}_n$ and $\dot{x}_n^+$ are velocity component normal to the obstacle before and after the collision. Moreover, the model predicts that the ratio between tangential impulse and normal impulse is proportional to the incidence angle, which is the

angle between the pre-collision velocity vector and the normal vector of the obstacle surface. As a result, we have

$$\dot{x}_t^+ = \dot{x}_t + \kappa(-e-1)\arctan\left(\frac{\dot{x}_t}{\dot{x}_n}\right)\dot{x}_n \tag{4.4}$$

where κ is a constant. $\dot{x}_t$ and $\dot{x}_t^+$ are velocity component tangential to the obstacle before and after the collision. With experiments, we identify $e = 0.43$ and $\kappa = 0.20$ for our experimental vehicle.

The post-collision acceleration is dependent on the attitude of the vehicle after collision, which can be hard to predict due to the large torque disturbance the vehicle may experience during the collision process. As a result, we assume $\ddot{\boldsymbol{x}} = \mathbf{0}$, which corresponds to a hovering status, and treat the difference between the true attitude after collision and the hovering attitude as an initial attitude error for the trajectory piece after the collision. As quadcopters have responsive attitude controllers, this error is expected to be corrected in a negligible time.

Improving the tracking of collision trajectories

Due to the torque disturbance during the collision, the vehicle can rotate with a large angular velocity after the collision. To mitigate the tracking error caused by this, we temporarily (for 0.3s) increase the gains of the attitude and angular rates controller of the vehicle after the collision.

Tracking experiment

The planned trajectories and tracking results for both collision-inclusive and collision-exclusive cases are shown in Fig. 4.6, and a composite image of the tracking experiment is shown in Fig. 4.7. The obstacle separates the space into two parts, connected by a 1m gap. The planned collision-inclusive trajectory takes 2.34 seconds, whereas the collision-exclusive trajectory takes 2.39 seconds.

For both scenarios, the quadcopter can successfully follow the reference to reach the end goal. However, we observe tracking error starting at the tip of the U-shape for both tracking attempts. For the collision-exclusive case, the error is caused by aggressive maneuver. For the collision-inclusive case, the collision introduces a torque disturbance that is not fully captured by the collision model. This makes the state of the vehicle deviates from the reference state and causes tracking error after the collision. This experiment verifies that trajectories generated by the collision-inclusive planner can be tracked successfully, and also suggests that better tracking of the collision trajectory can be achieved through decreasing the impact of torque disturbance on the system during collisions, either via physical designs or control strategies.

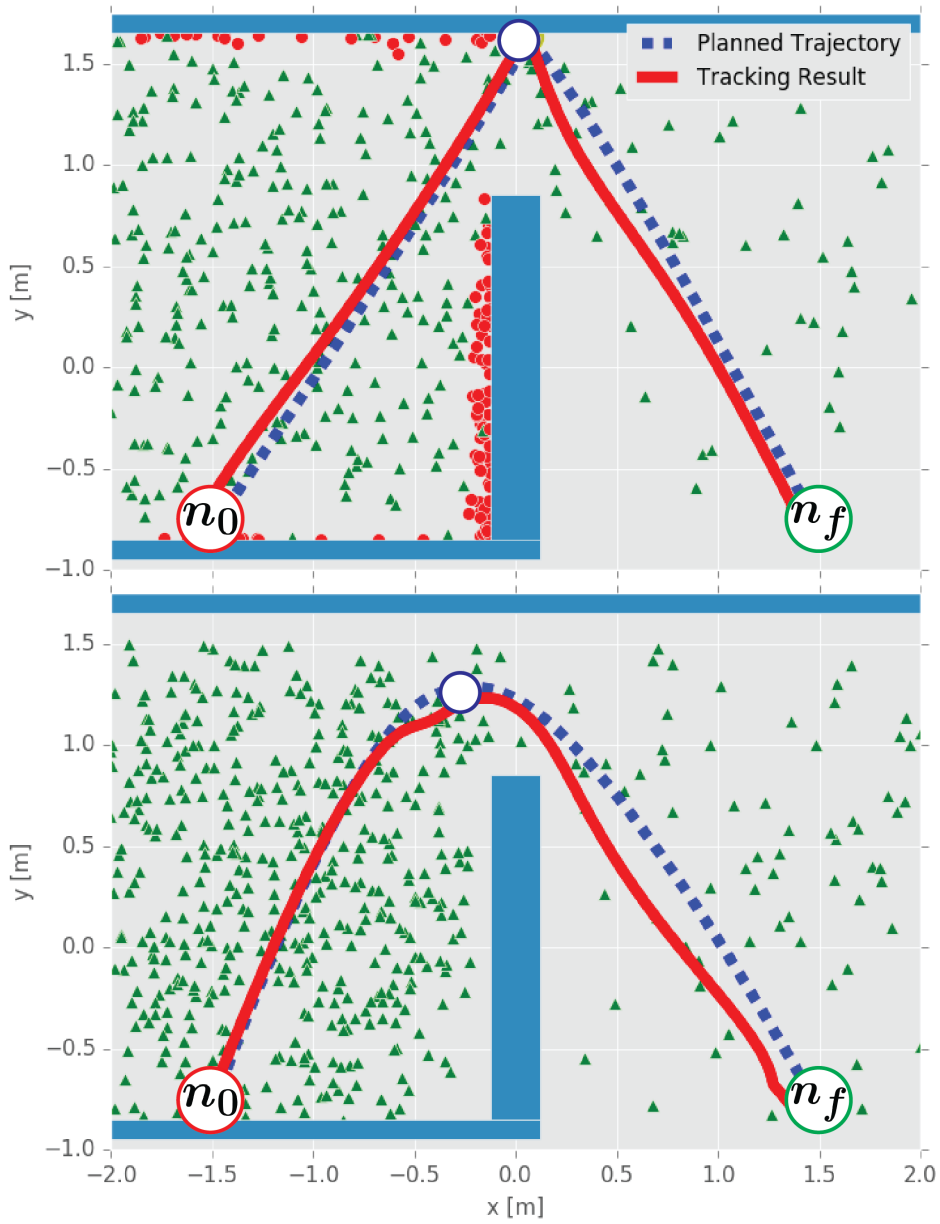

Figure 4.6: Experimental result of tracking planned trajectories from $\boldsymbol{n}_0$ to $\boldsymbol{n}_f$. Top: generated and executed collision-inclusive trajectory. Bottom: generated and executed collision-exclusive planner. The blue rectangles are the obstacles. Green triangles are non-collision nodes and the red dots are collision nodes in $\mathcal{T}$. The blue circle is an intermediate node.

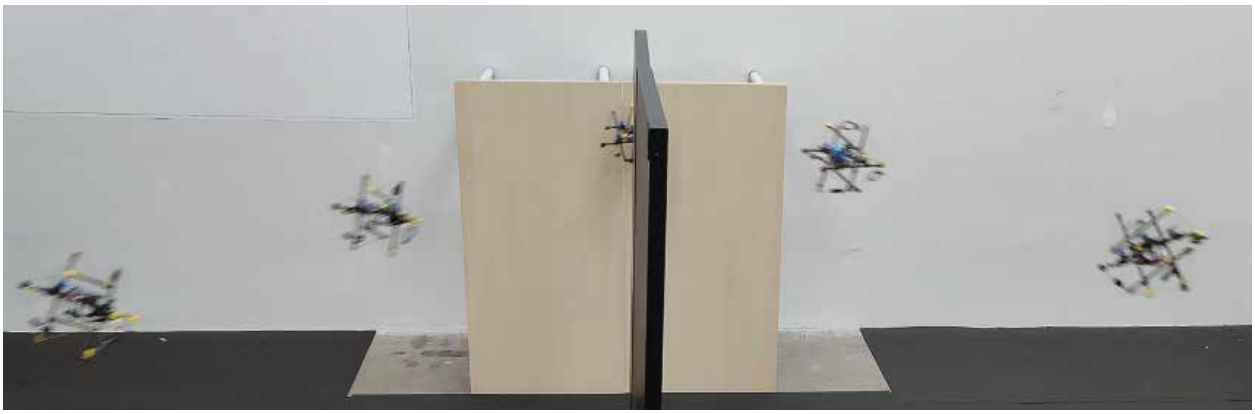

Figure 4.7: Composite image of the tracking experiment. The quadcopter tracks a trajectory from left to right while avoiding the black obstacle in the middle of the space. For the collision-inclusive trajectory, the quadcopter takes advantage of a collision with the yellow obstacle in the back. The distance between the shown left-most state and right-most state is about 2m.

4.6 Conclusion

In this chapter, we present a sampling-based motion planner that can exploit collisions to generate better trajectories for quadcopters. The planner samples collisions as impacts between generated motion primitives and obstacles, and connects collision states with other sampled states to form collision-inclusive trajectories.

Planning with collisions offers two benefits. First, allowing for collisions extends the feasible state space the planner can work in. Second, sampled states are no longer discarded due to infeasibility caused by collisions. Thus, the rate of adding samples to the exploring tree may increase. Collision-inclusive planning also has its disadvantages. The process of generating collision nodes requires additional computation time and it may cause sampled nodes to concentrate near obstacles, potentially slowing down the spatial expansion of the exploring tree. We illustrate with an example that the benefits of planning with collisions are likely to outweigh the disadvantages when searching for trajectories in narrow environments, where most generated trajectory pieces involve collisions.

We experimentally track trajectories generated by our planner. Experiment results indicate that a major source of tracking error comes from the disturbance that is not captured by the collision model. This can be mitigated by designing short and aggressive recovery trajectory pieces to decrease the variance of the state after collisions. Such trajectory pieces can also make the planner less dependent on the accuracy of collision models and hence make it easier to apply the planner on different quadcopter platforms.

Chapter 5

Water-air transition with a miniature unmanned aerial underwater vehicle

In previous chapters, we have introduced our work on extending the capabilities of UAVs in cluttered environments. In this chapter, we shift our focus and expand the operational environments of UAVs to multi-domain spaces with both air and water.

Amphibious mobility presents a unique opportunity in addressing an array of challenging tasks. These tasks include, but are not limited to, inspecting oil platforms, examining bridges, and surveying coastal ecosystems. However, accomplishing amphibious mobility is far from trivial for propeller-driven robots. The difference in densities between air and water, about a thousand-fold, poses a significant operation challenge. With the same rotating speed, a propeller will generate drastically different levels of thrust in air and water, which complicates the control of the robots. Furthermore, the transition from water to air is demanding, especially considering the added disturbance from the turbulence near the water surface.

To tackle these challenges, we introduce a miniature Unmanned Aerial Underwater Vehicle (mini UAUV) in this chapter, capable of amphibious mobility and transitioning between water and air. The mini UAUV features a simple mechanical design that resembles a traditional quadcopter. Its amphibious mobility is supported by three key components: 1) an in-depth characterization of the quadcopter propellers operating in both air and water regimes, 2) a Kalman Filter fusing accelerometer readings and barometer readings for depth estimation, and 3) a control strategy designed for the UAUV to breach still water surfaces. Using this strategy, the mini UAUV will first accelerate towards water surface to help its propellers leave water. It will then switch the control mode from water to air, spinning up its propellers and generating extra thrust to fully take off from water surface.

The material in this chapter is based on the following previously published work:

- Jiaming Zha, Eric Thacher, Joseph Kroeger, Simo A. Mäkiharju, and Mark W. Mueller, "Towards breaching a still water surface with a miniature unmanned aerial underwater

vehicle," in *Proceedings of the International Conference on Unmanned Aerial Systems (ICUAS)*, IEEE, 2019, pp. 1178–1185.

5.1 Introduction

Unmanned Aerial Vehicles (UAVs) and Autonomous Underwater Vehicles (AUVs) have been deployed for missions such as search and rescue [71], and construction inspection in environments hazardous to humans, like tall bridge towers [72] and hydroelectric dams [73]. In recent years, efforts have been made to combine UAVs and AUVs, resulting in the creation of Unmanned Aerial Underwater Vehicles (UAUVs) possessing both aerial and underwater mobility.

Figure 5.1: Image of the mini UAUV. The vehicle weighs 202g, and its largest linear measure is 14 cm. The three reflective markers are used by a motion capture system for in-flight state estimation.

To achieve such amphibious capability, a UAUV faces many challenges. Firstly, a UAUV needs to be waterproof yet lightweight enough for efficient flight. Secondly, a UAUV has to

generate sufficient thrust in both water and air, two media with densities differing by three orders of magnitude. Lastly, and the main focus of this chapter, a UAUV must overcome the difficulty of transitioning between water and air.

During the water-air transition, the UAUV loses the vertical buoyancy force provided by water. Concurrently, the thrust generated by propellers drastically decreases due to the change of the surrounding medium density. Furthermore, vortices will be induced by the propellers near the water surface, causing significant disturbances.

The challenges associated with water-air transition have led to the development of innovative UAUV designs incorporating various supportive mechanical structures such as multi-layered propellers and buoys. A literature review of these designs can be found in Chapter 2. However, such mechanical aids add extra weight and drag, which decrease the agility, maneuverability, and operation time of the vehicle.

In this chapter, we tackle the UAUV water-air transition challenge without relying on additional mechanical supports. We present a mechanically simple miniature UAUV system (abbreviated as mini UAUV, and shown in Fig. 5.1), which resembles conventional quadcopters, along with a transition control strategy to assist the mini UAUV in breaching the still water surface.

The trade-off with a simpler mechanical design is a deeper understanding of the operation of individual components in the air, water, and transition regimes. Particularly, as we use single-layered propellers instead of multi-layered ones, it becomes crucial to precisely define the performance of the propellers in both air and water. In addition, we have devised a depth estimator to help determine the vertical position of the mini UAUV relative to the free surface. This information helps the mini UAUV execute a well-timed switch from underwater to aerial operation. With the foundation provided by the propeller performance analysis and the depth estimator, we manage to achieve a mechanically-simple and low cost vehicle system that has amphibious mobility and can carry out media-transition.

The remainder of the chapter is structured as follows. We introduce the vehicle's dynamic model in Section 5.2. In Section 5.3, we present our result of characterizing the propeller performance in water, air, and the transition regimes. Section 5.4 details the depth estimator and control strategy for water breaching. The final design of the mini UAUV is presented in Section 5.5, followed by experimental validation in Section 5.6. We conclude the chapter in Section 5.7.

5.2 Modeling of the system dynamics

In this section we present the derivation of the translational and rotational dynamics of the mini UAUV. We will use non-bold letters like m for scalar, bold lower-case letters, such as $\boldsymbol{g}$ for vectors and bold upper-case letters, like $\boldsymbol{J}$ for matrices.

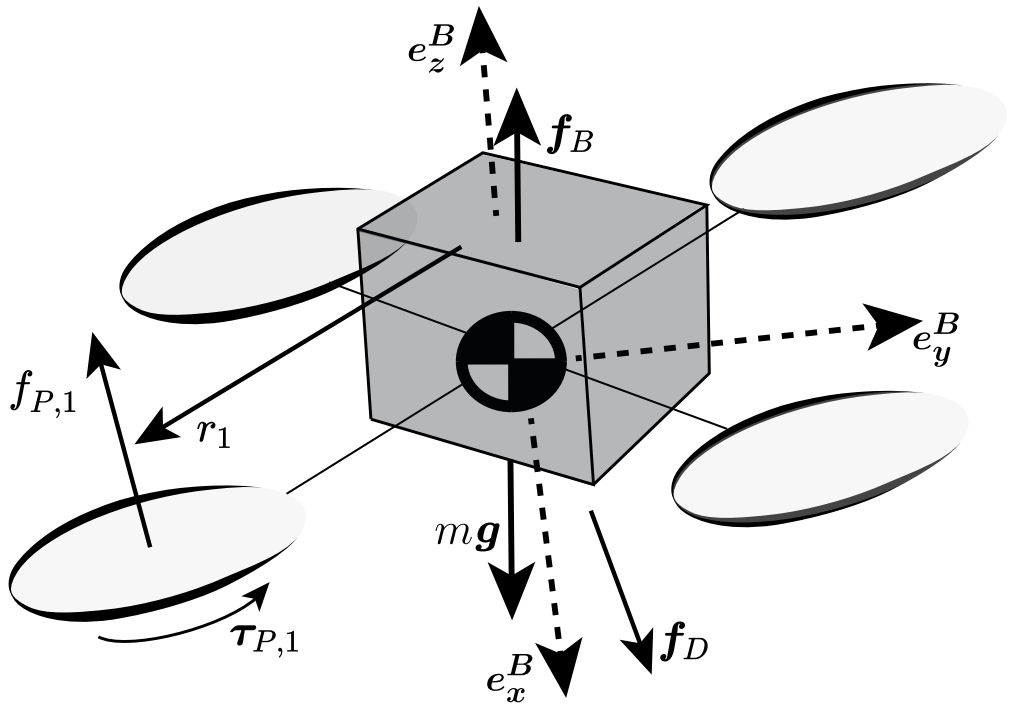

Figure 5.2: Salient forces and torques acting on the mini UAUV. Each propeller produces thrust $f_{P,i}$ and propeller torque $\tau_{P,i}$. The vector pointing from the center of mass of the vehicle to propeller i is denoted $\boldsymbol{r}_i$. When underwater, the system also experiences buoyancy $\boldsymbol{f}_B$ and drag $\boldsymbol{f}_D$. The vehicle's body-fixed frame is attached to the center of mass and defined with three axes orthogonal to each other, ($\boldsymbol{e}_x^{\boldsymbol{B}}$, $\boldsymbol{e}_y^{\boldsymbol{B}}$, $\boldsymbol{e}_z^{\boldsymbol{B}}$).

Translational dynamics

A diagram showing the mini UAUV under salient forces is shown in Fig. 5.2. Similar to our notation system in Chapter 3, the vehicle's attitude is defined as a rotation matrix $\boldsymbol{R}$, mapping vectors from the body-fixed frame $\boldsymbol{B}$ affixed to the center of mass, to the world frame $\boldsymbol{W}$, which is inertial and affixed to the ground, i.e., $\boldsymbol{v}^{\boldsymbol{W}} = \boldsymbol{R}\boldsymbol{v}^{\boldsymbol{B}}$. The vehicle's body-fixed frame is described with ($\boldsymbol{e}_x^{\boldsymbol{B}}$, $\boldsymbol{e}_y^{\boldsymbol{B}}$, $\boldsymbol{e}_z^{\boldsymbol{B}}$), three unit vectors orthogonal to each other. To avoid possible confusion, when a vector is used in analysis across different frames, we use superscript to indicate in which frame the vector is expressed.

The mass of the mini UAUV is m. Each propeller produces thrust force $f_{P,i}$ in the direction of $\boldsymbol{e}_z^{\boldsymbol{B}}$, and $\tau_{P,i}$ is the drag torque on propeller i. We use $\boldsymbol{r}_i$ to denote the vector pointing from the vehicle's center of mass to the propeller i. It is shifted up in the diagram for visibility. The buoyancy of the mini UAUV is given by $\boldsymbol{f}_B$, and $\boldsymbol{f}_D$ denotes the drag force the mini UAUV experiences. The unit gravity vector is denoted by $\boldsymbol{g}$.

With Newton's second law, we can derive the translational dynamics of the system:

$$m\ddot{\boldsymbol{d}} = m\boldsymbol{g} + \boldsymbol{R}\boldsymbol{e}_z^{\boldsymbol{B}} \sum_{i=1}^{4} f_{P,i} + \boldsymbol{f}_B + \boldsymbol{f}_D \tag{5.1}$$

where $\boldsymbol{d}$ is the vehicle's position relative to a fixed point in the world frame, and $\ddot{\boldsymbol{d}}$ is the acceleration vector of the mini UAUV.

According to Archimedes' principle, the direction of buoyancy is opposite to gravity and its value is equal to the weight of the media displaced by the vehicle:

$$\boldsymbol{f}_B = -\rho V \boldsymbol{g} \tag{5.2}$$

where V denotes the volume of the media displaced and ρ denotes the density of the media.

Meanwhile, drag force is exerted by the fluid in a direction that is opposite to the relative motion of the body with respect to fluid and can be approximated as [74]:

$$\boldsymbol{f}_D = -\frac{1}{2}\rho C_D(Re) A \left\| \dot{\boldsymbol{d}}_F \right\| \dot{\boldsymbol{d}}_F \tag{5.3}$$

where C_D represents drag coefficients and is a function of shape and Reynolds number, Re. A is the characteristic area of the mini UAUV. $\dot{\boldsymbol{d}}_F$ denotes the velocity vector of the vehicle with respect to surrounding fluid. Notice that when the surrounding fluid is stationary with respect to the world frame, $\dot{\boldsymbol{d}}_F = \dot{\boldsymbol{d}}$.

Here we neglect the effect of added mass and surface tension, which are likely to be small in comparison to buoyancy, thrust, and drag for a vehicle with a small body size.

Rotational dynamics

Similar to translational dynamics, we can model the rotational dynamics of the mini UAUV with Euler's equation:

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} + \boldsymbol{S}(\boldsymbol{\omega})\,\boldsymbol{J}\boldsymbol{\omega} = \sum_{i=1}^{4} (\boldsymbol{S}(\boldsymbol{r}_i)\,\boldsymbol{e}_z^{\boldsymbol{B}} f_{P,i} + \boldsymbol{e}_z^{\boldsymbol{B}} \tau_{P,i}) \tag{5.4}$$

where $\boldsymbol{S}(\cdot)$ maps a $\mathbb{R}^3$ vector to a corresponding $\mathbb{R}^{3\times3}$ skew-symmetric matrix. Left multiplying $\boldsymbol{S}(\cdot)$ is equivalent to the cross product. $\boldsymbol{J}$ is the moment of inertia matrix of the vehicle with respect to its center of mass. Meanwhile, the angular velocity vector of the vehicle, $\boldsymbol{\omega} \in \mathbb{R}^3$, represents the rotation velocity between the body-fixed frame and the world frame. It is expressed in the body-fixed frame and is related to the attitude matrix as follows:

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\,\boldsymbol{S}(\boldsymbol{\omega}) \tag{5.5}$$

Here we are neglecting the moment caused by drag as it is small comparing to the moment generated by propeller thrusts.

5.3 Vehicle performance in different media and during transitions

In this section we present the experimentally determined mapping between the input pulse-width modulation (PWM) command signal, motor's rotational speed, and thrust in both air and water for the mini UAUV's electronic speed controller (ESC) and brush-less motor pair. Notice that the mapping curves are only defined in the steady-state region far from the free surface. It is not possible to generate such curves accurately near the free surface (i.e. in the water-air transition region) because of the unsteady phase fraction of the fluid interacting with the propeller. Instead, we choose to characterize the bounds of the transition region, outside of which the free surface has minimal effect on propeller performance.

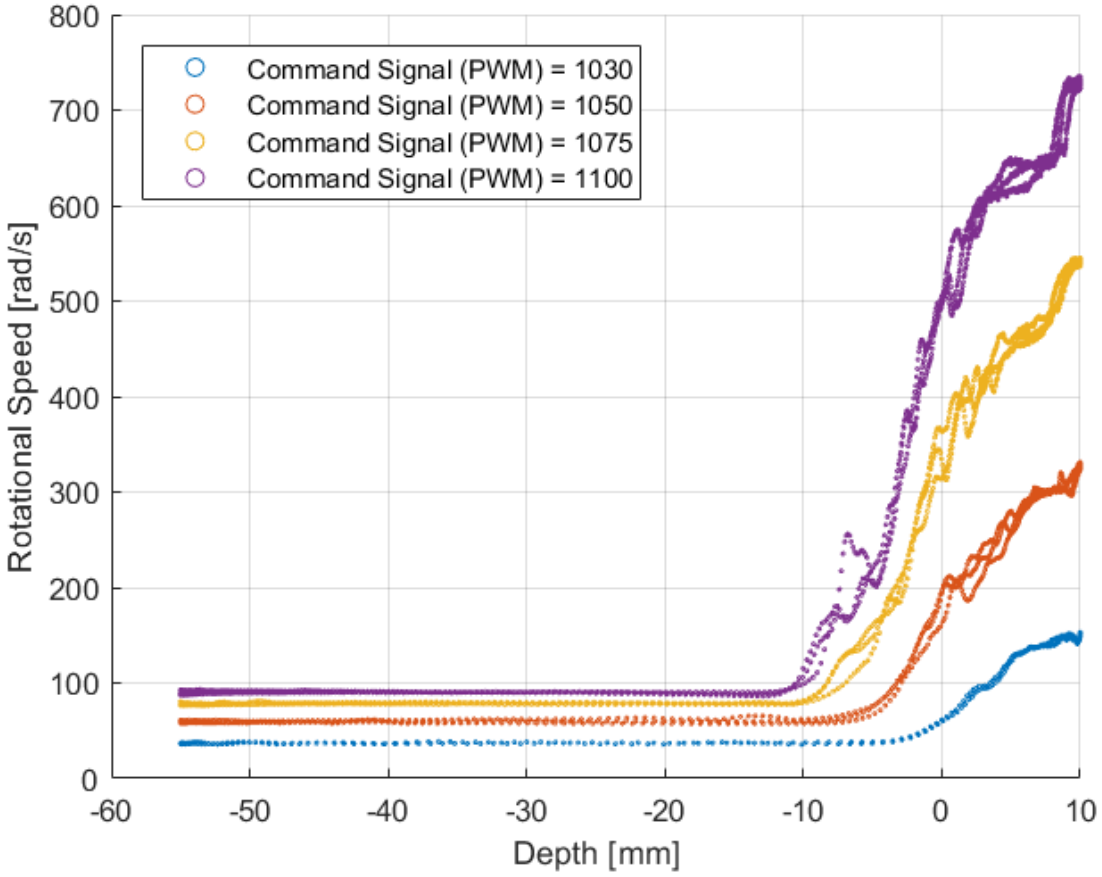

Figure 5.3: Propeller speed as a function of depth below free surface for different PWM command signals.

To simplify fluid-propeller interactions, the experiments are conducted using a single propeller. The propeller is mounted to a test stand, which measures the thrust generated. Rotational speed is measured independently by sampling the change in voltage from a single phase wire. For a three phase, 12-pole motor, the rotational speed, ω, is given in units of rad/s by $\omega = 12\pi f$ where f is the frequency of voltage fluctuations in the phase wire in Hz. The test stand is mounted to a linear stage, for which the position is determined with a encoder with $0.5\mu m$ accuracy. The linear stage is positioned vertically, so that the propeller is parallel to the free surface. This setup allows for precise control of the vertical position of the propeller relative to the free surface.

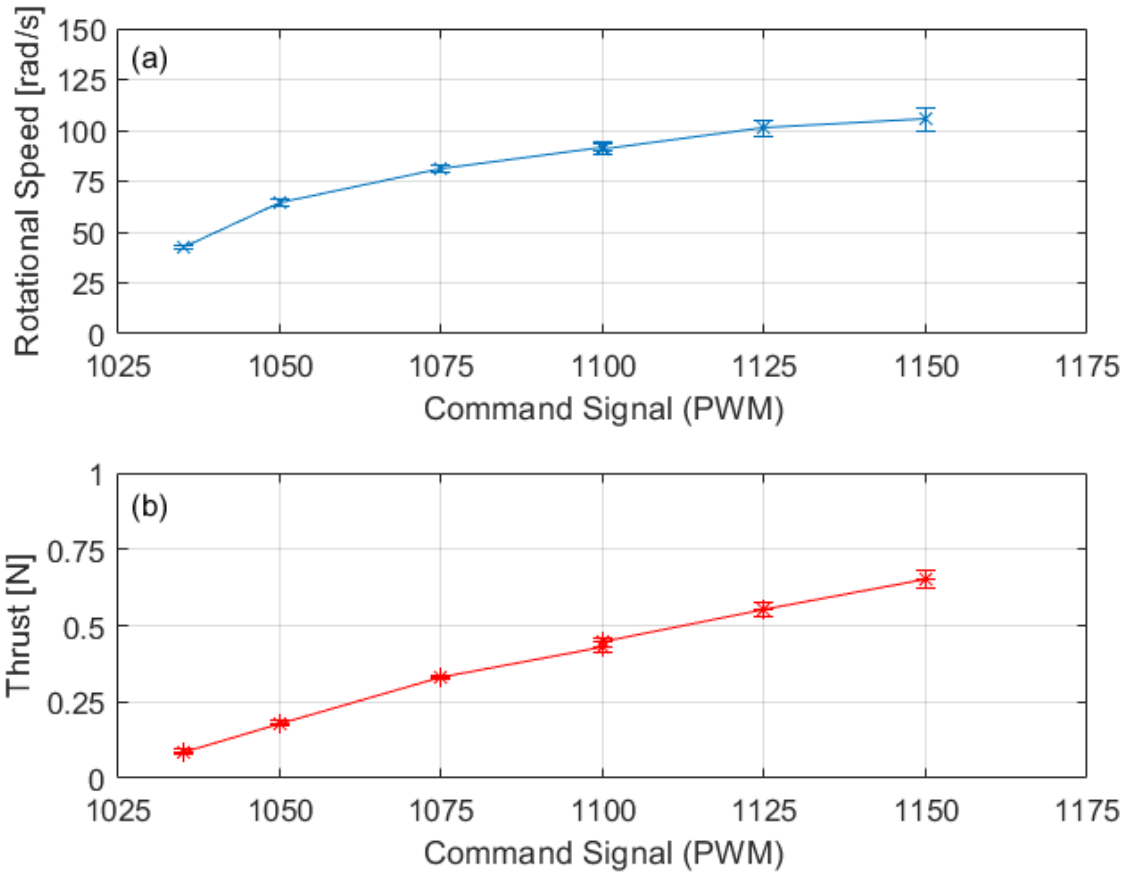

Figure 5.4: (a) Rotational speed and (b) thrust as a function of PWM input for operation of propeller underwater.

To determine the depth at which free surface effects are important, the propeller is moved smoothly from -55mm to 10mm while given constant PWM commands. The rotational speed plot for the tests can be found in Fig. 5.3. From the figure, we notice that increasing the

command signal value also increases the distance below water at which surface aeration can occur, and all rotational speed converges to steady state when the propeller is below -20mm. When above -20mm, air can be entrained below the surface of the water, and the propeller speed increases due to the reduction of the density of the surrounding fluid. Notice that the phase fraction of the fluid interacting with the propeller is highly variable, which in turn increases the variability of the speed across multiple tests.

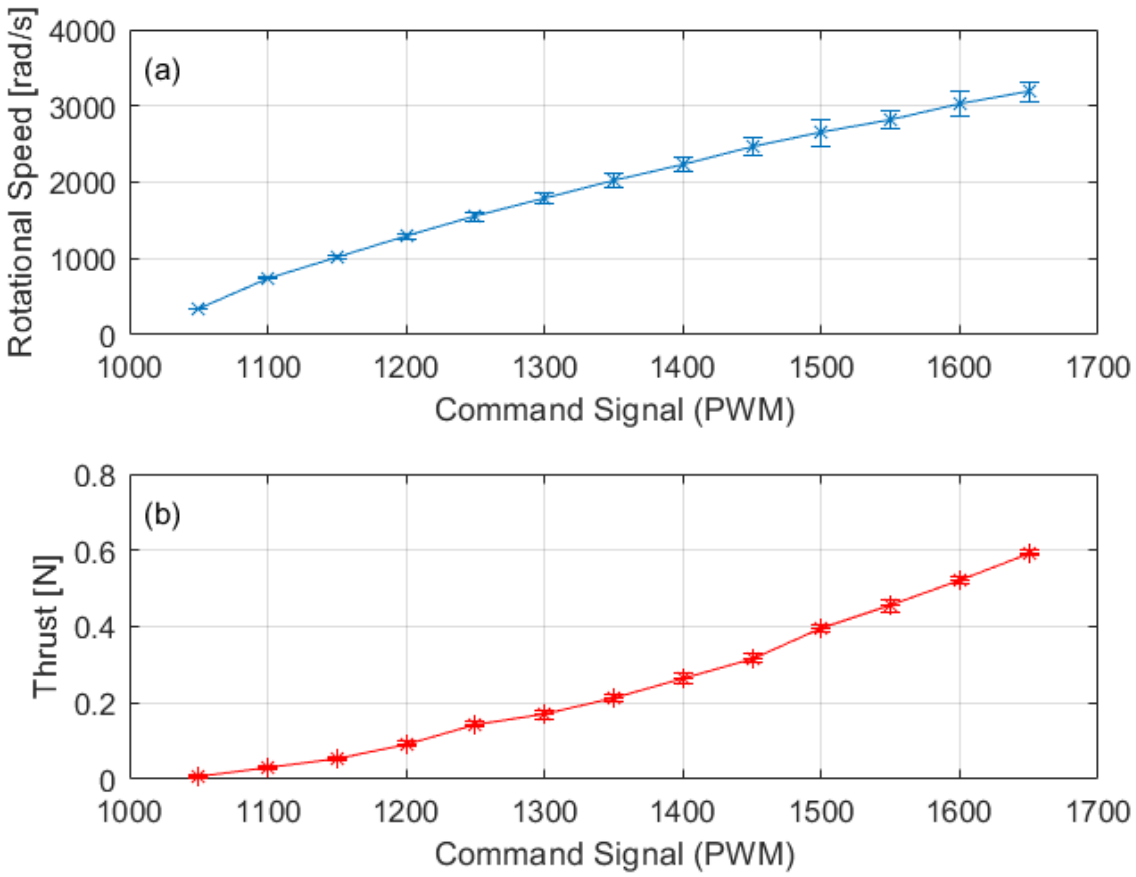

Figure 5.5: (a) Rotational speed and (b) thrust as a function of PWM input for operation of propeller in air.

The steady-state thrust and rotational speed of the propeller underwater as a function of command signal is given in Fig. 5.4. The given results are for a depth of -45mm. The buoyancy of the test setup below the water is subtracted from the data, so that the value plotted is solely the thrust generated by the propeller. It is worth noting that the lowest command signal used for the test is 1030 PWM, because this is the lowest signal that can

command the brushless motor to start spinning reliably underwater. The rotational speed and thrust of the propeller as a function of command signal in air is given in Fig. 5.5.

Comparing Fig. 5.4 and Fig. 5.5, we observe that the relationship between command signal and thrust, defined as command-thrust mapping, depends on the medium the propeller operates in. For example, a 1100 PWM command signal input will generate 0.42N in water, but only 0.03N in air. Clearly, if the command signal were to be kept constant during the transition from water to air, the UAUV would not produce enough thrust to support its own weight. Moreover, with the change of the command-thrust mapping, we also observe a significant shift of the operating range of command signals. Consequently, it is of great importance to accurately identify the medium the propellers operate in and precisely trigger the switch of the control regime at the interface. The control strategy for doing so is described in the following section.

5.4 State estimation and control strategy

In this section, we discuss the mini UAUV's sensing system, its depth estimator and its control strategy. It is worth noting that the mini UAUV's underwater state estimator is quite different from the one used in air, which relies on a motion capture system that can track objects in 3D space. When the mini UAUV operates underwater, it only has access to its IMU and pressure sensor. Thus, it can only estimate its roll, pitch, depth and vertical position with confidence.

Sensors

The mini UAUV's onboard sensing system is based on an IMU with 6 DOF. In addition, the mini UAUV also features a high precision pressure sensor (barometer) that provides ambient pressure readings. When the mini UAUV is flying, a motion-capture system measures position and orientation of the vehicle. However, the motion-capture measurement does not work well when the mini UAUV operates underwater because water attenuates the infrared signal reflected by motion-capture markers.

Underwater state estimation

When the vehicle operates underwater, we aim to estimate its depth and vertical speed. We present a simple Kalman Filter for estimating these states, using measurements from the onboard barometer and accelerometer.

The linear kinematics in the vertical direction of the world frame is captured as follows:

$$\begin{bmatrix} \dot{d}_z \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d_z \\ v_z \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a_z \tag{5.6}$$

where d_z is the vehicle's position along the world-fixed vertical, while v_z and a_z are the vehicle's velocity and acceleration along the vertical. The acceleration along the vertical is estimated using the onboard accelerometer, and a barometer provides depth measurements.

As a result, we can estimate the depth and the vertical velocity of the vehicle with a Kalman Filter, assuming constant acceleration over a sampling time T. The barometer reading is linear in the depth, allowing for a straight-forward implementation of a linear, time-varying Kalman Filter.

In-flight estimation

In flight, the vehicle is visible to a motion capture system, allowing for off-board estimation of the vehicle's full 6-DOF position and orientation states.

UAUV controller

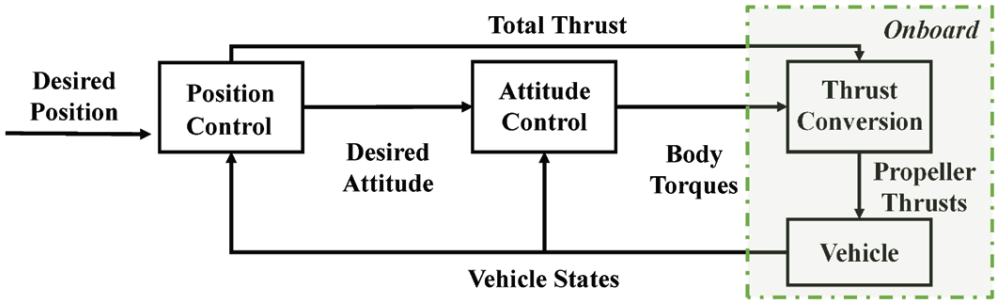

Figure 5.6: Controller architecture of the mini UAUV

The controller of the UAUV is presented in Fig. 5.6. It features a cascaded control structure. A position controller outputs desired total thrust and thrust direction, whereas an inner attitude controller computes desired torques. Finally, a mixer converts the total thrust and body torque commands to per-propeller thrust commands. This cascaded structure can be decoupled into two separate parts: an offboard controller for position and attitude control, and an onboard part implementing thrust conversion.

The position controller regulates the position error as a diminishing second order system with damping ratio ζ_p and natural frequency ω_p:

$$\ddot{\boldsymbol{d}}_d = 2\zeta_p\omega_p(\dot{\boldsymbol{d}}_d - \dot{\boldsymbol{d}}) + \omega_p^2(\boldsymbol{d}_d - \boldsymbol{d}) \tag{5.7}$$

wherein $\boldsymbol{d}$ is the position of the vehicle, $\ddot{\boldsymbol{d}}_d$ is desired acceleration, $\dot{\boldsymbol{d}}_d$ is desired velocity and $\boldsymbol{d}_d$ is desired position. All vectors in the equation above are represented in the world frame. Thus, we can find desired total thrust and its corresponding direction as:

$$f_d = m||\ddot{\boldsymbol{d}}_d - \boldsymbol{g}||_2, \qquad \boldsymbol{z}_{B,d}^{\boldsymbol{W}} = \frac{\ddot{\boldsymbol{d}}_d - \boldsymbol{g}}{||\ddot{\boldsymbol{d}}_d - \boldsymbol{g}||_2} \tag{5.8}$$

With the desired thrust direction and an additional desired yaw angle, the desired attitude can be defined as the attitude which maps $\boldsymbol{e}_z^{\boldsymbol{B}}$ to a vector aligned with $\boldsymbol{z}_{B,d}^{\boldsymbol{W}}$ in the world frame, while achieving the desired yaw angle. A desired angular velocity $\boldsymbol{\omega}_d^{\boldsymbol{B}}$ can be computed as proportional to the attitude error and then the desired angular acceleration follows:

$$\dot{\boldsymbol{\omega}}_d^{\boldsymbol{B}} = \frac{1}{\tau_t}(\boldsymbol{\omega}_d^{\boldsymbol{B}} - \tilde{\boldsymbol{\omega}}^{\boldsymbol{B}}) \tag{5.9}$$

where $\tilde{\boldsymbol{\omega}}^{\boldsymbol{B}}$ is the angular velocity measured by the onboard rate gyroscope and τ_t is the desired feedback time constant. All vectors are in vehicle's body-fixed frame. Thus, the desired torque, computed from the Euler's equation, follows as:

$$\boldsymbol{\tau}_d^{\boldsymbol{B}} = \boldsymbol{J}\dot{\boldsymbol{\omega}}_d^{\boldsymbol{B}} + \boldsymbol{S}\left(\tilde{\boldsymbol{\omega}}^{\boldsymbol{B}}\right)\boldsymbol{J}\tilde{\boldsymbol{\omega}}^{\boldsymbol{B}} \tag{5.10}$$

Lastly, the thrust converter computes the desired thrust force for each propeller as:

$$f_{P,i} = \frac{1}{4}\left(\begin{bmatrix} r_{i,y}^{-1} & -r_{i,x}^{-1} & h_i\kappa^{-1} \end{bmatrix}\boldsymbol{\tau}_d^{\boldsymbol{B}} + f_d\right) \tag{5.11}$$

where $r_{i,x}$ and $r_{i,y}$ are the components of $\boldsymbol{r}_i$, along the $\boldsymbol{e}_x^{\boldsymbol{B}}$ and the $\boldsymbol{e}_y^{\boldsymbol{B}}$ direction respectively. Handedness of the propeller i is denoted by h_i (1 for right-handed, -1 for left-handed), and κ is the propeller's torque coefficient.

Transition strategy between underwater and air operation

As is discussed in Section 5.3, the command-thrust mapping and the command signal operating range in water are significantly different from those in air. If an erroneous mapping were used, the mini UAUV would generate thrusts far away from desired, and fail to breach the water. As a result, the key to a successful water-air transition is to switch the command-thrust mapping from underwater mode to air mode precisely when the vehicle reaches the interface. Moreover, it is desirable to carry out the mapping switch when the propellers are exposed to air rather than fully submerged in water, as using the air command-thrust mapping underwater will generate an unexpected large thrust that can potentially destabilize the system. Based on above discussion, we propose a three-step transition strategy (Fig. 5.7):

1. Accelerate towards water surface with underwater command-thrust mapping.
2. Breach the water surface. Switch to air command-thrust mapping when all propellers are exposed to air.
3. Proceed flying in air with air command-thrust mapping.

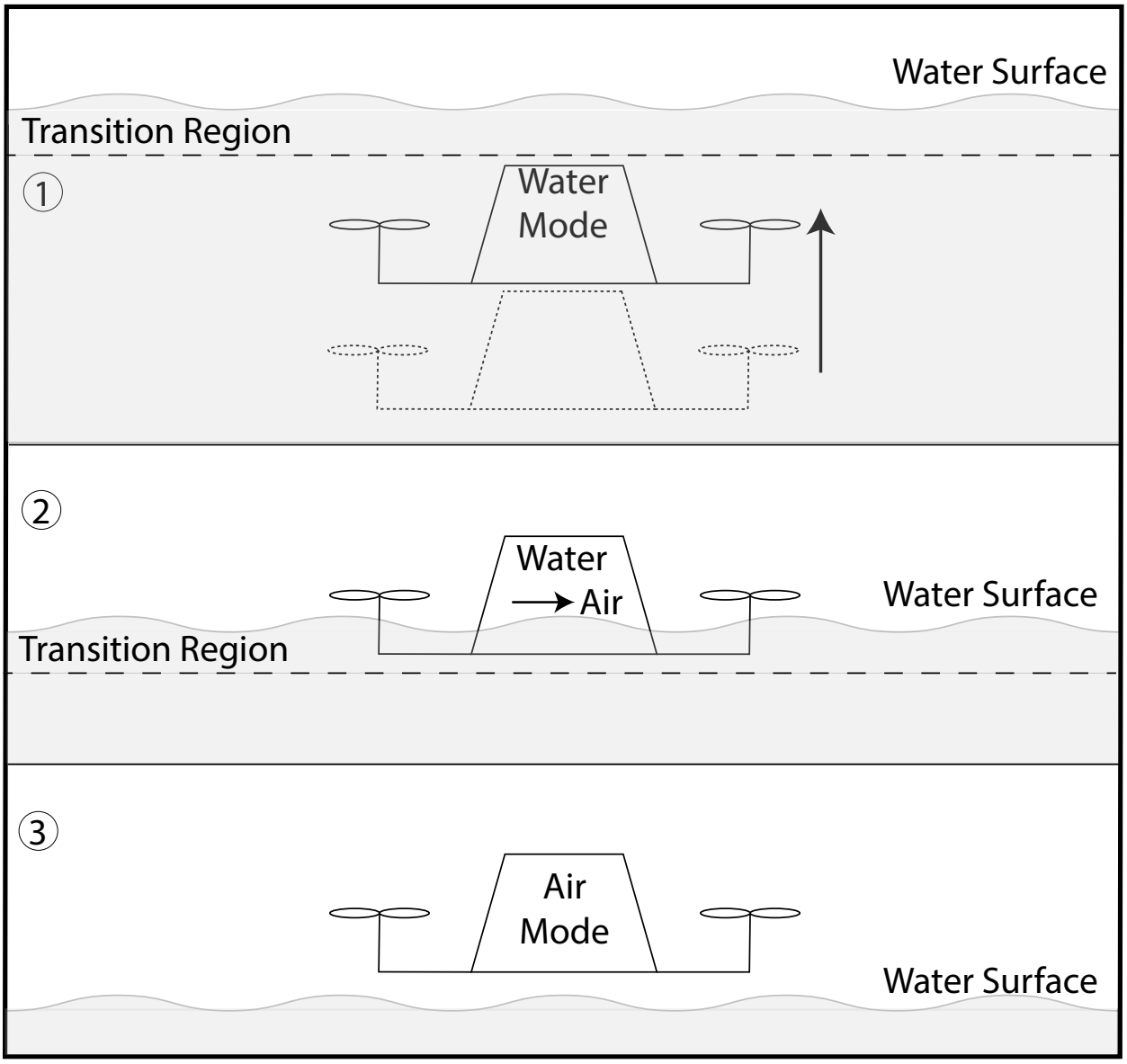

Figure 5.7: Illustration of mini UAUV's control strategy for water breaching. Top: the mini UAUV accelerates toward the water surface. Middle: the mini UAUV switch its control mode when all of its propellers have left water. Bottom: the mini UAUV continue its flight with the aerial mode.

While accelerating towards the water surface, the depth estimator is used to track the vertical position and velocity of the mini UAUV. However, once the mini UAUV enters the transition region, the location of which is characterized in Section 5.3, air entrainment alters the medium density and we can no longer reliably calculate depth based on the pressure reading. Hence, the depth estimator cannot tell us the exact time at which the mini UAUV breaches the water.

To solve the problem, we start a timer once the the mini UAUV reaches the transition region. During breaching, the mini UAUV has an strict time window in which the propellers are exposed to air. We can switch the mini UAUV to use the air command-thrust mapping when the timer reading falls in this window.

We estimate the time window with the force balance on the body in the vertical direction, which is given in Eq. (5.1). To simplify the model, we assume the mini UAUV as a prism. It has a base area A, and its volume is uniformly distributed along its height, h. We define $z_{surface}$ as the vertical position of the water surface and z_t as the vertical position of the prism's top surface, both in the world frame. Then, when $z_{surface} < z_t < z_{surface} + h$ the buoyancy force can be given as:

$$f_{B,z} = \rho g A(h + z_{surface} - z_t) \tag{5.12}$$

Meanwhile, the drag force in the vertical direction can be derived from Eq. (5.3) as:

$$f_{D,z} = -\frac{1}{2}\rho C_D(Re) A|\dot{z}_t|\dot{z}_t \tag{5.13}$$

Hence, we can estimate the vertical motion of the vehicle with the following ordinary differential equation:

$$\ddot{z}_t = -g + \frac{\rho g A(h + z_{surface} - z_t)}{m} - \frac{\rho C_D(Re) A|\dot{z}_t|\dot{z}_t}{2m} \tag{5.14}$$

Notice that propeller thrusts are ignored here, because when the vehicle operates in the air with the underwater command-thrust mapping, the thrusts generated are small compared to buoyancy and drag.

We can solve Eq. (5.14) numerically using the position and velocity at the start of the breaching event to obtain the estimated time window. Due to the uncertainty associated with the density of the surrounding fluid and the magnitude of the related forces, this is computed offline and used as a benchmark value after which the switching time is manually tuned.

5.5 Hardware design

To build a miniature UAUV capable of breaching the water surface, we need to fulfill the following design requirements:

First, the vehicle is small-scale, lightweight, and slightly negatively buoyant (i.e. the gravity force acting on it is larger than the buoyancy force), so that the vehicle can both sink underwater and operate in the air. Second, the whole vehicle system should be water-proof. Third, the onboard pressure sensor should have access to the pressure information of surrounding media so as to provide readings for the depth estimator.

In order to meet these requirements, we have created a system featuring following designs.

Vehicle frame and aviation stack

To minimize the size, we build a small vehicle frame and a highly compact aviation stack.

As can be seen in Fig. 5.1, the vehicle features a lightweight carbon fiber frame with an arm length of 65mm manufactured by water-jet and four EMAX 7500KV high thrust-to-weight ratio brushless motor.

The onboard aviation stack is composed of a Bitcraze Crazyflie 2.0 [75] chip as an integrated computation and sensing unit, and a 4-in1 DYS 18A BLHeli_S ESC to drive the brushless motors. Additional auxiliary circuits, including power regulators and voltage dividers are arranged on a custom-made printed circuit board. All electronic components are powered by a single 2-cell 800mAh LiPo battery. Laser-cut mounting adapters were designed to help assemble the electronic stack vertically and minimize the space it takes.

Waterproof housing with membrane

To waterproof the mini UAUV, we fabricate a housing shell encasing all on-board electronics via 3D-SLA-printing. An O-ring groove that docks a 55mm ID O-ring is engraved at bottom of the shell to provide sealing between the shell and carbon fiber frame. To pass ambient pressure information to the onboard pressure sensor, we build the top of the shell with a 1mm-thick latex membrane. As the latex membrane is highly flexible, it equalizes the air pressure in the shell to ambient water pressure. As a result, we are able to directly measure surrounding water pressure from inside the shell.

5.6 Experimental validation

To assess the ability of the mini UAUV to operate in water, air, as well as the transition regime, an experiment was conducted of the mini UAUV breaching the water surface in calm water. In doing so, the mini UAUV demonstrates the ability to transition between underwater and air operation autonomously.

Physical parameters

The physical parameters of the mini UAUV are listed in Table 5.1.

Table 5.1: Mini UAUV Physical Parameters

Parameter	Value
vehicle mass	$m = 0.202\,\text{kg}$
vehicle volume	$V = 186 \times 10^{-6}\,\text{m}^3$
moment of inertia along body x axis	$J_{xx} = 112 \times 10^{-6}\,\text{kg}\,\text{m}^2$
moment of inertia along body z axis	$J_{zz} = 187 \times 10^{-6}\,\text{kg}\,\text{m}^2$
maximum thrust per propeller	$f_{P_i,max} = 1.1\,\text{N}$
vehicle arm length	$\|\|\boldsymbol{r}_i\|\|_2 = 0.065\,\text{m}$
propeller torque constant	$\kappa = 808 \times 10^{-5}\,\text{m}$

Experiment setup and the test trajectory

The experiment is designed to verify the water breaching strategy proposed in Section 5.4. We command the mini UAUV to accelerate towards the free surface, breach the surface and then keep flying in air to a desired height.

We carry out the breaching strategy by programming the mini UAUV to follow a two phase trajectory. Initially, the vehicle is placed 0.15m below water surface, on the bottom of a water tank. When the program starts, the controller first commands the mini UAUV to track a trajectory with an initial desired vertical velocity of 1m/s and a desired vertical acceleration of 2m/s^2. This phase lasts for 1.3s and the trajectory is used to ensure that the mini UAUV smoothly rises through the water to the free surface. After 1.3s, the controller will check if the mini UAUV is successfully detected by the motion capture system. If yes, the trajectory enters phase two and the offboard controller commands a desired vertical position of 1m above water surface, with zero desired velocity and acceleration. Otherwise, an emergency stop of the mini UAUV is triggered.

The test is carried out in a lab space equipped with motion capture system and an off board controller sends radio commands to the mini UAUV at a rate of 50Hz.

When tracking the aforementioned trajectory, the mini UAUV breaches the water with a velocity of about 0.5m/s. Using an initial estimate from Eq. (5.14) and subsequent tuning, we program the controller to switch command-thrust mapping 0.08s after the mini UAUV detects that it reaches the transition region. Last, it is worth noting that we disable mini UAUV's yaw control and only control its roll and pitch during phase 1, because we don't have a reliable yaw measurement underwater. After the vehicle is detected by the motion capture system and enters phase 2, we re-enable the full attitude control.

Result and analysis

Fig. 5.8 shows the estimated vertical position of the mini UAUV during the test. We define zero datum at the water surface. The blue curve shows the estimate provided by mini UAUV's underwater depth estimator, whereas the red curve shows the estimate provided by the motion capture system. Due to the nature of these estimates, their accuracy depends

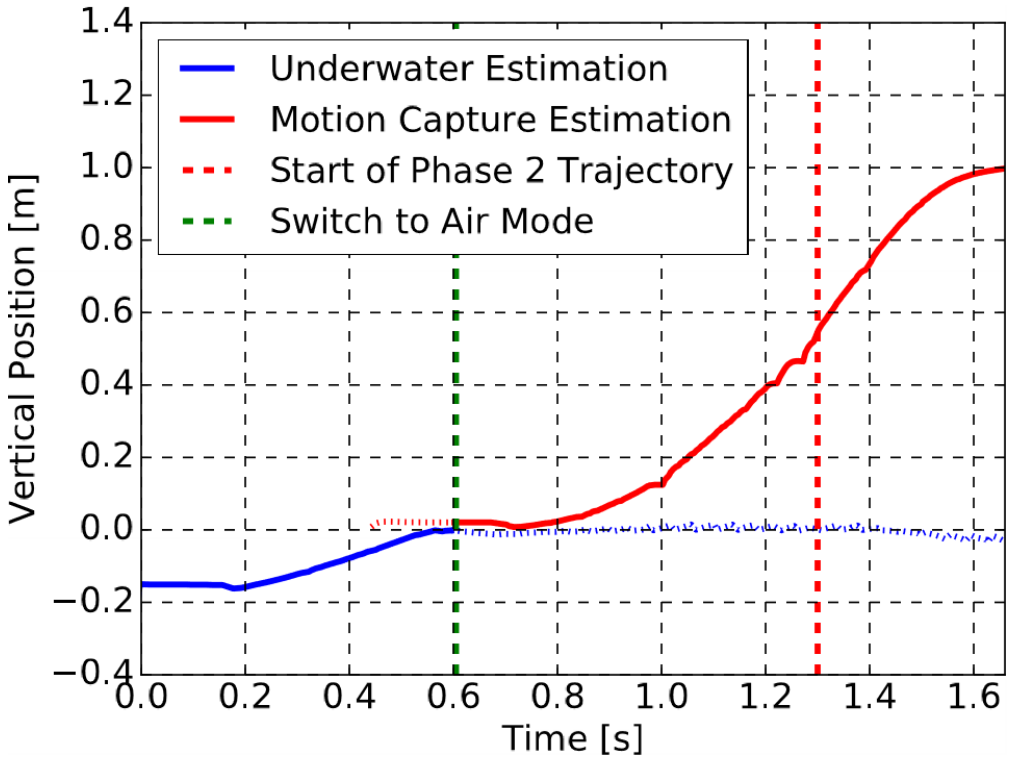

Figure 5.8: Estimated vertical position of the mini UAUV during the water breaching test.

on the medium in which the UAUV operates. In particular, when the UAUV is outside of water the pressure-based depth estimator is unable to differentiate changes in position. Furthermore, due to signal attenuation of the motion capture infrared signal, the motion capture system is unable to provide accurate estimates underwater. With this in mind, when position estimates cannot be trusted, the readout is plotted as dotted curve. The red vertical line shows when the vehicle enters phase 2 of the trajectory and the green vertical line denotes when the mini UAUV switches from underwater mode to air mode.

The mini UAUV reaches the transition region at about t = 0.5s. Instead of continuing to rise, the mini UAUV loses its initial momentum and stays on top of the free surface for about 0.2 seconds. During the staling time, 0.08s is spent switching the control mode, and multiple reasons can contribute to the remaining 0.12s. Potential causes for the delay are the time needed for the propellers to spin up and the generation of turbulence on the water surface from propeller-fluid interaction. Particularly, once the UAUV switches to air mode, the resulting increase in rotational speed induces significant water-air mixing in the fluid surrounding the propellers. Since the control strategy produces a command signal based on expected rotational speed and thrust, an unsteady operating medium would prevent the desired thrust from being generated. This phenomena is also captured for a single propeller in Fig. 5.3, which shows an increased variability in rotational speed during the transition regime.

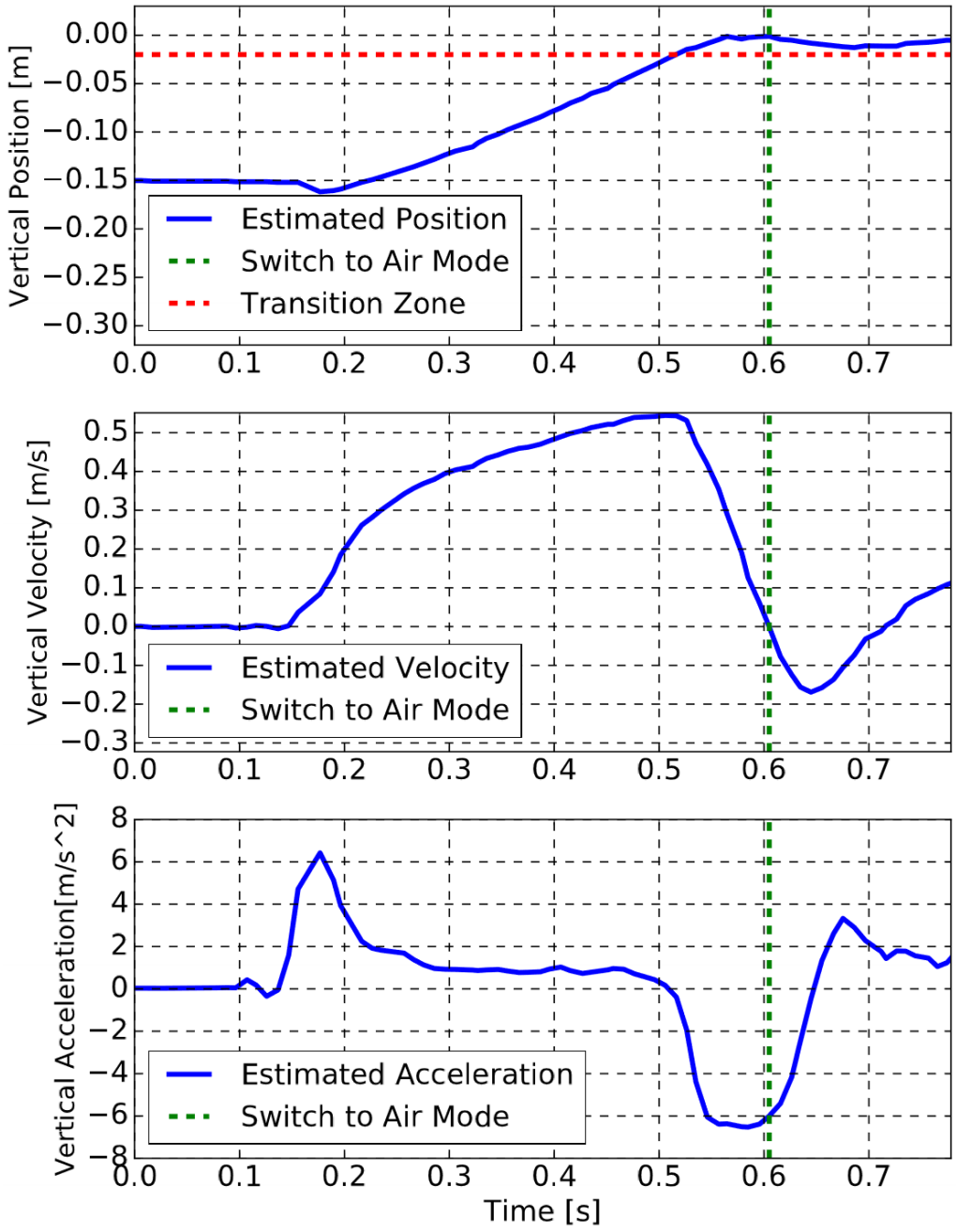

Figure 5.9: Estimate of the mini UAUV's vertical position, velocity and acceleration, both underwater and during the transition phase.

In addition to the above challenges, from Fig. 5.8 we can observe that the vertical position grows linearly, instead of quadratically to time when the mini UAUV implements the underwater acceleration trajectory, showing a notable tracking error. This indicates a discrepancy between our model and reality, which can be contributed to under-estimation of the drag effects underwater and not considering the added mass effect.

Fig. 5.9 illustrates in detail the estimated state of the mini UAUV's motion underwater and during the water-air transition phase. The estimate of vertical position and velocity is provided by the depth estimator described in Section 5.4 and is based on both barometer and IMU readings. The horizontal red dotted line marks the beginning of the transition region and the vertical green dotted line denotes where the mini UAUV switches from underwater mode to air mode. We observe that the switch to air mode occurs when the velocity of the UAUV is about 0m/s. This shows that the spin up of the propellers takes place when the mini UAUV is about to lose all its vertical momentum and starts dropping back to water. At about 0.7s, the velocity turns positive and the mini UAUV starts to climb, marking the successful propeller spin-up and the end of the breaching phase. The UAUV then follows the command to 1m above water surface.

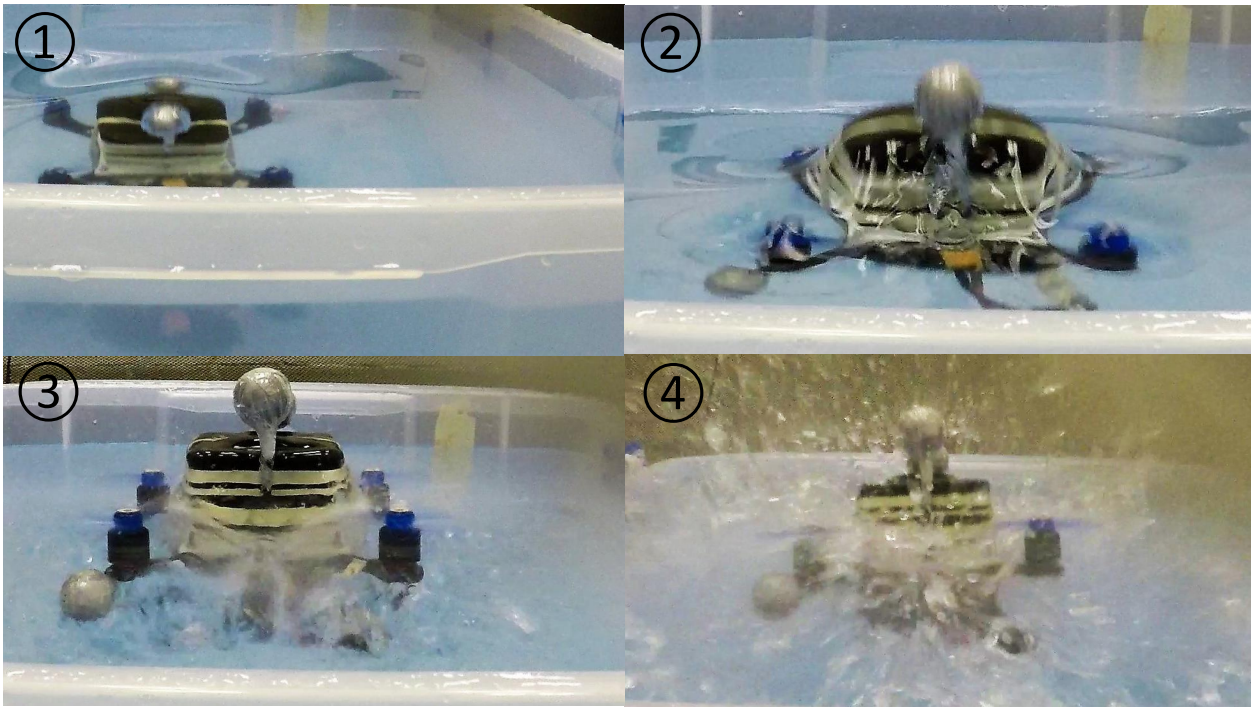

Figure 5.10: Image sequence showing the mini UAUV breaching the water surface.

Fig. 5.10 showcases an image sequence of the mini UAUV's water breach. In image 1, the vehicle accelerates towards water surface with underwater command-thrust mapping. In image 2, the top of the vehicle breaches water surface. In image 3, all propellers leave water

surface and the vehicle switches to air command-thrust mapping. In image 4, the vehicle fully leaves water and starts flying in the air. The experiment demonstrates that the mini UAUV is capable of transitioning from still water to air. Meanwhile, the tracking error during the experiment also points out that our model can be further improved by a better estimate of the drag effect when the mini UAUV is underwater and during transition phase.

The video of the experiment can be viewed at **youtu.be/y4-ZcgsTGAQ**.

5.7 Conclusion

In this chapter, we present a mechanically simple miniature UAUV that can autonomously breach a calm water surface without additional mechanical supports. It features an onboard Kalman Filter that provides information about the vehicle depth and its rate of change. With the waterproof design and a control strategy of accelerating towards water surface and then switching command-thrust mapping, the mini UAUV expands the operational environments of UAVs to multi-domain spaces with its amphibious mobility.

Chapter 6

Conclusion and future work

6.1 Conclusion

This dissertation explores the enhancement of UAVs' capabilities and operational environments through the integration of design, control, and planning algorithms. Specifically, we introduce two unique UAV designs capable of operating in cluttered and multi-domain environments, and a planning algorithm including collisions into its framework.

In Chapter 3, we introduce the collision-resilient tensegrity aerial vehicle. This UAV features an icosahedron tensegrity shell composed of rods suspended in a tension network of strings. This unique shell allows the vehicle to withstand high-speed impacts, thus opening up opportunities for UAV operations in cluttered environments without the need for complex collision avoidance algorithms. Additionally, we propose an autonomous re-orientation controller to enable post-collision flight resumption. With collision resilience and re-orientation control, the tensegrity aerial vehicles can operate safely in challenging, cluttered environments. We validate this concept through an experiment of the tensegrity vehicle navigating autonomously in a forest environment with tree obstacles previously unknown to the vehicle.

In the following chapter, we expand upon the concept of collision resilience by integrating collision directly into motion planning. By adapting the RRT* algorithm to accommodate collision, the planner gains two advantages. First, the feasible state space is expanded, allowing the planner to consider new trajectory candidates that were originally deemed infeasible, such as those exploiting collisions to quickly change movement directions. Second, sampled states are no longer discarded due to infeasibility caused by collisions, potentially increasing the rate of adding samples to the exploration tree and shortening the computation time. We illustrate these advantages with an example demonstrating the benefit of planning with collisions in a narrow tunnel environment. In addition, we experimentally track trajectories generated by our planner, showcasing how collisions can enhance the operation of aerial vehicles by using collisions to replace aggressive maneuvers.

In Chapter 5, we extend UAV operation to multi-domain environments. The UAUV we propose in this chapter has a simple mechanical structure similar to a traditional quadcopter.

It can operate both in air and water, and breach the still water surface with the assistance of a barometer-based depth estimator and a water-breaching strategy. This strategy leverages the additional vertical acceleration the vehicle can attain underwater due to buoyancy and determines an optimal time to switch the control mode from underwater to aerial operation.

In conclusion, the designs, the control strategies, and the planning algorithm presented in this dissertation extend the operational environments of UAVs to cluttered regions and multi-domain spaces. While some limitations persist, the groundwork laid in this dissertation opens avenues for future research, which we outline in the following section.

6.2 Future work

This dissertation has proposed and validated methods to enhance the operational capabilities of UAVs in complex environments. Nevertheless, there remains considerable scope for extending this line of research to address unexplored or partially addressed problems.

Firstly, in relation to the tensegrity aerial vehicle, there is significant potential for exploring new forms of aerial vehicles that utilize tensegrity structures. An avenue worth pursuing is the incorporation of the idea of modular robotic units, where actuators and batteries are integrated directly into the rigid rods of the tensegrity structures. By interconnecting these integrated rods in various string patterns, we can potentially create aerial robots with unique tensegrity structures. This can lead to modular aerial vehicles that can be assembled or disassembled for varying mission requirements, thus offering great versatility in UAV designs and applications.

A second research direction is related to the collision-inclusive motion planning algorithm, specifically addressing its dependence on collision models. Given the wide variety of surface materials in real-world environments, the planner may encounter significant difficulty in predicting the collision dynamics. Two strategies could potentially help with this problem. Firstly, for UAVs with cameras and strong onboard computational power, vision can be used to predict the characteristics of different surfaces and their corresponding collision dynamics. Secondly, it would be useful to design motion primitives that assist vehicles in recovering from collisions and reaching specific post-collision states. Despite the unpredictability of the collisions, such primitives help make the states tractable after collision-recoveries, and thus make the planning algorithm robust against collision model mismatches.

The UAUV design and water-breaching strategy also invite further study. Detailed analyses could help find the optimal breaching windows for vehicles across different scales. In parallel, the development of design tools to help select appropriate motors and propellers that support multi-media operation could significantly expedite the development process, ultimately leading to more efficient and versatile UAUV designs.

Developing UAVs capable of matching the abilities of natural fliers in complex environments presents a significant and ongoing challenge. This dissertation has demonstrated that an integrated approach, which combines design, control, and motion planning, can offer viable solutions for extending the operational environments of UAVs. These synergistic

methods may unlock new possibilities that would be unattainable through advancements in individual methods alone. That being said, the potential to enhance UAV performance in complex environments remains largely unexplored, and the work presented in this dissertation represents only an early step in this exciting direction.

Bibliography

[1] H. Kang, H. Li, J. Zhang, X. Lu, and B. Benes, "Flycam: Multitouch gesture controlled drone gimbal photography," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3717–3724, 2018.

[2] S. Siebert and J. Teizer, "Mobile 3d mapping for surveying earthwork projects using an unmanned aerial vehicle (uav) system," *Automation in construction*, vol. 41, pp. 1–14, 2014.

[3] J. Kim, S. Kim, C. Ju, and H. I. Son, "Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications," *IEEE Access*, vol. 7, pp. 105 100–105 115, 2019.

[4] BBC-Earth, "Torpedo gannet diving," YouTube, Nov 2014, [Accessed: 2023-07-10]. [Online]. Available: https://youtu.be/1Cp1n_vPvYY

[5] S. Wanless, T. Corfield, M. Harris, S. Buckland, and J. Morris, "Diving behaviour of the shag phalacrocorax aristotelis (aves: Pelecaniformes) in relation to water depth and prey size," *Journal of Zoology*, vol. 231, no. 1, pp. 11–25, 1993.

[6] BBC-Two, "Goshawk flies through tiny spaces in slomo," YouTube, Apr 2011, accessed: 2023-07-10. [Online]. Available: https://youtu.be/2CFckjfP-1E

[7] A. Briod, P. Kornatowski, J.-C. Zufferey, and D. Floreano, "A collision-resilient flying robot," *Journal of Field Robotics*, vol. 31, no. 4, pp. 496–509, 2014.

[8] Flyability, "Elios 3," https://www.flyability.com/elios-3.

[9] C. J. Salaan, K. Tadakuma, Y. Okada, Y. Sakai, K. Ohno, and S. Tadokoro, "Development and experimental validation of aerial vehicle with passive rotating shell on each rotor," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2568–2575, 2019.

[10] H. Jia, S. Bai, R. Ding, J. Shu, Y. Deng, B. L. Khoo, and P. Chirarattananon, "A quadrotor with a passively reconfigurable airframe for hybrid terrestrial locomotion," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 4741–4751, 2022.

[11] P. Sareh, P. Chermprayong, M. Emmanuelli, H. Nadeem, and M. Kovac, "Rotorigami: A rotary origami protective system for robotic rotorcraft," *Science Robotics*, vol. 3, no. 22, p. eaah5228, 2018.

[12] P. De Petris, H. Nguyen, M. Kulkarni, F. Mascarich, and K. Alexis, "Resilient collision-tolerant navigation in confined environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2286–2292.

[13] S. Mintchev, J. Shintake, and D. Floreano, "Bioinspired dual-stiffness origami," *Science Robotics*, vol. 3, no. 20, p. eaau0275, 2018.

[14] J. Jang, K. Cho, and G.-H. Yang, "Design and experimental study of dragonfly-inspired flexible blade to improve safety of drones," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4200–4207, 2019.

[15] J. Shu and P. Chirarattananon, "A quadrotor with an origami-inspired protective mechanism," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3820–3827, 2019.

[16] Z. Liu and K. Karydis, "Toward impact-resilient quadrotor design, collision characterization and recovery control to sustain flight after collisions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 183–189.

[17] R. de Azambuja, H. Fouad, Y. Bouteiller, C. Sol, and G. Beltrame, "When being soft makes you tough: A collision-resilient quadcopter inspired by arthropods' exoskeletons," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7854–7860.

[18] H. Zhou, A. R. Plummer, and D. Cleaver, "Distributed actuation and control of a tensegrity-based morphing wing," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 1, pp. 34–45, 2021.

[19] J. J. Rimoli, "On the impact tolerance of tensegrity-based planetary landers," in *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2016, p. 1511.

[20] A. Zhang, D. Hutchings, M. Gupta, and A. Agogino, "Orientation control of self-righting tensegrity landers," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 85451. American Society of Mechanical Engineers, 2021, p. V08BT08A025.

[21] K. Garanger, I. del Valle, M. Rath, M. Krajewski, U. Raheja, M. Pavone, and J. J. Rimoli, "Soft tensegrity systems for planetary landing and exploration," in *Earth and Space 2021*, 2021, pp. 841–854.

[22] V. SunSpiral, G. Gorospe, J. Bruce, A. Iscen, G. Korbel, S. Milam, A. Agogino, and D. Atkinson, "Tensegrity based probes for planetary exploration: Entry, descent and landing (edl) and surface mobility analysis," *International Journal of Planetary Probes*, vol. 7, p. 13, 2013.

[23] K. Kim, L.-H. Chen, B. Cera, M. Daly, E. Zhu, J. Despois, A. K. Agogino, V. SunSpiral, and A. M. Agogino, "Hopping and rolling locomotion with spherical tensegrity robots," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4369–4376.

[24] S. Mintchev, D. Zappetti, J. Willemin, and D. Floreano, "A soft robot for random exploration of terrestrial environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7492–7497.

[25] D. Zappetti, Y. Sun, M. Gevers, S. Mintchev, and D. Floreano, "Dual stiffness tensegrity platform for resilient robotics," *Advanced Intelligent Systems*, vol. 4, no. 7, p. 2200025, 2022.

[26] H. Cotgrove, "Tensegrity drones," https://www.haydencotgrove.com/tensegrity-drones, (Accessed on 01/02/2022).

[27] J. Zha, X. Wu, J. Kroeger, N. Perez, and M. W. Mueller, "A collision-resilient aerial vehicle with icosahedron tensegrity structure," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1407–1412.

[28] S. Savin, A. Al Badr, D. Devitt, R. Fedorenko, and A. Klimchik, "Mixed-integer-based path and morphing planning for a tensegrity drone," *Applied Sciences*, vol. 12, no. 11, p. 5588, 2022.

[29] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[30] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[31] D. Burke, A. Chapman, and I. Shames, "Generating minimum-snap quadrotor trajectories really fast," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1487–1492.

[32] Z. Wang, H. Ye, C. Xu, and F. Gao, "Generating large-scale trajectories efficiently using double descriptions of polynomials," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7436–7442.

[33] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE transactions on robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[34] N. Bucki and M. W. Mueller, "Rapid collision detection for multicopter trajectories," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7234–7239.

[35] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[36] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[37] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[38] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5054–5061.

[39] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "Rrt*-smart: Rapid convergence implementation of rrt* towards optimal solution," in *2012 IEEE International Conference on Mechatronics and Automation*. IEEE, 2012, pp. 1651–1656.

[40] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[41] Z. Tang, B. Chen, R. Lan, and S. Li, "Vector field guided rrt* based on motion primitives for quadrotor kinodynamic planning," *Journal of Intelligent & Robotic Systems*, pp. 1–15, 2020.

[42] M. Mote, J. P. Afman, and E. Feron, "Robotic trajectory planning through collisional interaction," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1144–1149.

[43] M. Mote, M. Egerstedt, E. Feron, A. Bylard, and M. Pavone, "Collision-inclusive trajectory optimization for free-flying spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 7, pp. 1247–1258, 2020.

[44] N. Wang and R. G. Sanfelice, "Hysst: A stable sparse rapidly-exploring random trees optimal motion planning algorithm for hybrid dynamical systems," *arXiv preprint arXiv:2305.18649*, 2023.

[45] Z. Lu and K. Karydis, "Optimal steering of stochastic mobile robots that undergo collisions with their environment," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 668–675.

[46] W. Stewart, W. Weisler, M. MacLeod, T. Powers, A. Defreitas, R. Gritter, M. Anderson, K. Peters, A. Gopalarathnam, and M. Bryant, "Design and demonstration of a seabird-inspired fixed-wing hybrid uav-uuv system," *Bioinspiration & biomimetics*, vol. 13, no. 5, p. 056013, 2018.

[47] T. Hou, X. Yang, H. Su, B. Jiang, L. Chen, T. Wang, and J. Liang, "Design and experiments of a squid-like aquatic-aerial vehicle with soft morphing fins and arms," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 4681–4687.

[48] Z. Zeng, C. Lyu, Y. Bi, Y. Jin, D. Lu, and L. Lian, "Review of hybrid aerial underwater vehicle: Cross-domain mobility and transitions control," *Ocean Engineering*, vol. 248, p. 110840, 2022.

[49] A. Durve and A. Patwardhan, "Numerical and experimental investigation of onset of gas entrainment phenomenon," *Chemical Engineering Science*, vol. 73, pp. 140–150, 2012.

[50] M. M. Maia, D. A. Mercado, and F. J. Diez, "Design and implementation of multirotor aerial-underwater vehicles with experimental results," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 961–966.

[51] D. A. M. Ravell, M. M. Maia, and F. J. Diez, "Modeling and control of unmanned aerial/underwater vehicles using hybrid control," *Control Engineering Practice*, vol. 76, pp. 112–122, 2018.

[52] P. L. Drews, A. A. Neto, and M. F. Campos, "Hybrid unmanned aerial underwater vehicle: Modeling and simulation," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4637–4642.

[53] A. A. Neto, L. A. Mozelli, P. L. Drews, and M. F. Campos, "Attitude control for an hybrid unmanned aerial underwater vehicle: A robust switched strategy with global stability," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 395–400.

[54] Z. Ma, J. Feng, and J. Yang, "Research on vertical air–water trans-media control of hybrid unmanned aerial underwater vehicles based on adaptive sliding mode dynamical surface control," *International Journal of Advanced Robotic Systems*, vol. 15, no. 2, p. 1729881418770531, 2018.

[55] H. Alzu'bi, I. Mansour, and O. Rawashdeh, "Loon copter: Implementation of a hybrid unmanned aquatic–aerial quadcopter with active buoyancy control," *Journal of Field Robotics*, 2018.

[56] "Seahawk alfa seawater mission test," YouTube, accessed: 2023-07-10. [Online]. Available: https://youtu.be/UDx4hGZBDyo

[57] C. Lyu, D. Lu, C. Xiong, R. Hu, Y. Jin, J. Wang, Z. Zeng, and L. Lian, "Toward a gliding hybrid aerial underwater vehicle: Design, fabrication, and experiments," *Journal of Field Robotics*, vol. 39, no. 5, pp. 543–556, 2022.

[58] X. Wu and M. W. Mueller, "Using multiple short hops for multicopter navigation with only inertial sensors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8559–8565.

[59] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Unmanned aerial vehicles (UAVs): Collision avoidance systems and approaches," *IEEE Access*, vol. 8, pp. 105 139–105 155, 2020.

[60] B. Jessen, "Orthogonal icosahedra," *Nordisk Matematisk Tidskrift*, pp. 90–96, 1967.

[61] S. Pellegrino and C. R. Calladine, "Matrix analysis of statically and kinematically indeterminate frameworks," *International Journal of Solids and Structures*, vol. 22, no. 4, pp. 409–428, 1986.

[62] F. P. Beer, E. R. Johnston, J. T. DeWolf, and D. F. Mazurek, *Mechanics of Materials, Chapter 4-Pure Bending*, 6th ed. McGraw-Hill, 2012.

[63] R. Goyal and R. E. Skelton, "Dynamics of class 1 tensegrity systems including cable mass," in *Earth and Space 2018: Engineering for Extreme Environments*. ASCE, 2018, pp. 868–876.

[64] R. Fenwick and D. Bull, "What is the stiffness of reinforced concrete walls," *SESOC journal*, vol. 13, no. 2, pp. 23–32, 2000.

[65] SciPy Developers, "scipy.integrate.radau," https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.Radau.html.

[66] M. D. Shuster *et al.*, "A survey of attitude representations," *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[67] J. Mattingley and S. Boyd, "Cvxgen: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[68] BetaFpv, "Pavo25 Frame Kit," https://betafpv.com/products/pavo25-frame-kit.

[69] M. W. Mueller, M. Hehn, and R. D'Andrea, "Covariance correction step for kalman filtering with an attitude," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2301–2306, 2017.

[70] J. Calsamiglia, S. W. Kennedy, A. Chatterjee, A. Ruina, and J. T. Jenkins, "Anomalous frictional behavior in collisions of thin disks," *Journal of Applied Mechanics*, vol. 66, no. 1, pp. 146–161, 1999.

[71] S. Waharte and N. Trigoni, "Supporting search and rescue operations with uavs," in *Emerging Security Technologies (EST), 2010 International Conference on*. IEEE, 2010, pp. 142–147.

[72] N. Metni and T. Hamel, "A uav for bridge inspection: Visual servoing control law with orientation limits," *Automation in construction*, vol. 17, no. 1, pp. 3–10, 2007.

[73] P. Ridao, M. Carreras, D. Ribas, and R. Garcia, "Visual inspection of hydroelectric dams using an autonomous underwater vehicle," *Journal of Field Robotics*, vol. 27, no. 6, pp. 759–778, 2010.

[74] B. R. Munson, T. H. Okiishi, A. P. Rothmayer, and W. W. Huebsch, *Fundamentals of fluid mechanics*. John Wiley & Sons, 2014.

[75] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in *Methods and Models in Automation and Robotics (MMAR), 2017 22nd International Conference on*. IEEE, 2017, pp. 37–42.