Leveraging Aerial Transformation for Enhanced Air-Ground Robotic Mobility

Thesis by

Ioannis M. Mandralis

In Partial Fulfillment of the Requirements for the

Degree of

Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY

Pasadena, California

2026

Defended October 14th 2025

ACKNOWLEDGEMENTS

I would like to express my gratitude and deepest appreciation to my advisors, Mory Gharib and Richard Murray, for their invaluable guidance, patience, generosity, and counsel. Having one great advisor is rare, but having two is even rarer—I have had the privilege of being a part of two fantastic groups and have been given countless opportunities to grow on both a personal and a scientific level. I am also grateful to the members of my thesis committee, Aaron Ames, John Dabiri, and Jane Bae, for their mentorship and many shared experiences that greatly enriched my time at Caltech.

Many collaborators have contributed to the content of this thesis. Alireza Ramezani inspired many novel research directions, fruitful collaborations, and provided invaluable advice. Reza Nemovi and Joshua Gurovich generously provided technical support, as well as companionship through deadlines and technical challenges. Adrian Ghansah, Eric Sihite, and Adarsh Salagame provided valuable technical input, support, and guidance. I would also like to thank the X1 Caltech–TII team and mainly Mory, Aaron, Joel, Palas, Adrian, Adarsh, Josh, and Reza for the fun times in demos, travel, meetings, and for coming together to bring our work out of the lab and into the real world. I am particularly grateful to the Onassis foundation for their support throughout my studies.

All members of the Gharib and Murray groups contributed to the completion of this thesis through their camaraderie, intellectual discussions, and collaborative spirit. I am particularly thankful to Alexandros Rosakis who gave me the opportunity to work beside him on the Ultrasound Shape Sensing project, and Alejandro Stefan-Zavala who shared with me his passion on fan-array modeling and control. I am also indebted to Gabriel Margaria and Severin Schumacher for their efforts which contributed to Chapters 5 and 6 of this thesis, and Vincent, Quentin, Simon, Steven, Diego, and Sophia, who I had the privilege to supervise. I thank Sarah Pontes and Karima Guthrie for their friendship, administrative support, and tireless positive spirit.

Lastly, I want to express my gratitude to my beloved family and friends—both old and new—that have contributed to this thesis through their patience, sacrifice, support, and friendship. Isabel, your companionship has kept me grounded, positive, and has provided me with invaluable joys beyond work. To my parents, Panayiota and Zenon, and sister, Katerina: thank you for supporting my move across the world, uplifting me in difficult times, and encouraging my ambition—you have been instrumental to my work. This thesis is dedicated to all of you!

ABSTRACT

Ground-aerial robots can extend endurance, versatility, and robustness by combining wheeled motion with flight, yet many flying-rolling robot designs add actuators that increase weight and reduce efficiency. Morphobots mitigate this by using multipurpose actuators and body shape change to switch modes on the ground, but unpredictable vehicle-ground interactions can be an obstacle to robust operation. This dissertation develops the *Aerially Transforming Morphobot* (ATMO), a quadcopter that reconfigures in flight to land on wheels, enabling reliable air-ground transitions, mode switching without the hindrances of ground-morphing, and improved agility. We present ATMO's design and performance characterization, analyze its dynamics—revealing transformation-induced couplings incompatible with standard quadcopter control—and introduce a model-predictive control framework that stabilizes ATMO through aerial transformation to execute dynamic transitions. We then compare this approach with a learning-based controller that uses deep reinforcement learning for end-to-end morpho-transition, validating both experimentally. Finally, we revisit ATMO's design using aerodynamic principles to expand morphing flight through wake vectoring, showing that passive structures in the rotor wake substantially increase available thrust authority. Overall, we demonstrate that aerial shape change improves agility and reliability, highlighting a new direction for research in ground-aerial robotics.

PUBLISHED CONTENT AND CONTRIBUTIONS

Ioannis Mandralis, Reza Nemovi, Alireza Ramezani, Richard M. Murray, and Morteza Gharib. "ATMO: an aerially transforming morphobot for dynamic ground-aerial transition". In: *Communications Engineering* 4.1 (Apr. 2025). ISSN: 2731-3395. DOI: `10.1038/s44172-025-00413-6`. URL: `http://dx.doi.org/10.1038/s44172-025-00413-6`. (published).
I.M participated in the project conception, performed experiments, wrote the manuscript, and analyzed data.

V. Gherold, I. Mandralis, E. Sihite, A. Salagame, A. Ramezani, and M. Gharib. "Self-Supervised Cost of Transport Estimation for Multimodal Path Planning". In: *IEEE Robotics and Automation Letters* 10.7 (2025), pp. 6872–6879. DOI: `10.1109/LRA.2025.3572792`. (published).
I.M participated in the project conception, participated in model development, supported experiments, contributed to manuscript writing and review, analyzed data, and provided supervision.

Ioannis Mandralis, Eric Sihite, Alireza Ramezani, and Morteza Gharib. "Minimum Time Trajectory Generation for Bounding Flight: Combining Posture Control and Thrust Vectoring". In: *2023 European Control Conference (ECC)*. 2023, pp. 1–7. DOI: `10.23919/ECC57647.2023.10178360`. (published).
I.M participated in the project conception, performed experiments, wrote the manuscript, and analyzed data.

Ioannis Mandralis, Severin Schumacher, and Morteza Gharib. Wake Vectoring for Efficient Morphing Flight. 2025. (under review).
I.M participated in the project conception, performed experiments, wrote the manuscript, and analyzed data.

Ioannis Mandralis, Richard M. Murray, and Morteza Gharib. "Quadrotor Morpho-Transition: Learning vs Model-Based Control Strategies". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2025. arXiv: `2506.14039 [cs.RO]`. (accepted).
I.M participated in the project conception, performed experiments, wrote the manuscript, and analyzed data.

Alejandro A. Stefan-Zavala, Isabel Scherl, Ioannis Mandralis, Steven L. Brunton, and Morteza Gharib. *Data-Driven Modeling for On-Demand Flow Prescription in Fan-Array Wind Tunnels*. 2024. eprint: `arXiv:2412.12309`. (accepted).
I.M participated in the project conception, participated in model development, supported experiments, and contributed to manuscript writing and review.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

PREFACE

In a world where Artificial Intelligence (AI) has fundamentally changed how we obtain information, write software, or solve scientific problems, collective imagination is captivated by what's next. Can the magic of AI be brought into the physical realm? Robotics and the sciences surrounding it are ideally poised to answer this question. A world populated by safe, autonomous robots holds the potential to fundamentally revolutionize society, removing the need for humans to perform laborious tasks, vastly multiplying industrial productivity, improving care for the sick or elderly, and transporting goods and people with higher efficiency.

Investment in humanoid robotics has accelerated, and, coupled with progress toward general-purpose embodied intelligence, promises substantial impact. At the same time, form should match function: not every task benefits from human-like morphology, and there is significant room for purpose-built autonomous systems that interact with the physical world in ways humans cannot.

Take drones, for example—these systems deliver substantial benefits due to their ability to navigate the airspace, reach and manipulate objects at great heights, and take off and land in severely constrained environments. Drones are already delivering real impact for advanced delivery systems and disaster response operations, as well as surveying, gathering, and distributing critical environmental information. However, flying robots remain limited by energy requirements, payload capacity, and manipulation and maneuvering capabilities that are still in their infancy. There are clearly important gains to be made in the design of flying robotic systems, as well as in the algorithms that enable useful physical interaction while ensuring safety near humans.

Addressing these limitations calls for robotic platforms that offload long-duration work to an energetically frugal ground mode while reserving flight for access, surveillance, and repositioning. Ground-aerial robots embody exactly this division of labor and are therefore well poised to increase the reliability and scope of autonomous missions. In this thesis we establish a new class of ground-aerial robots capable of aerial shape change and develop methods that leverage this shape change to enhance air-ground mobility.

Chapter 1

INTRODUCTION

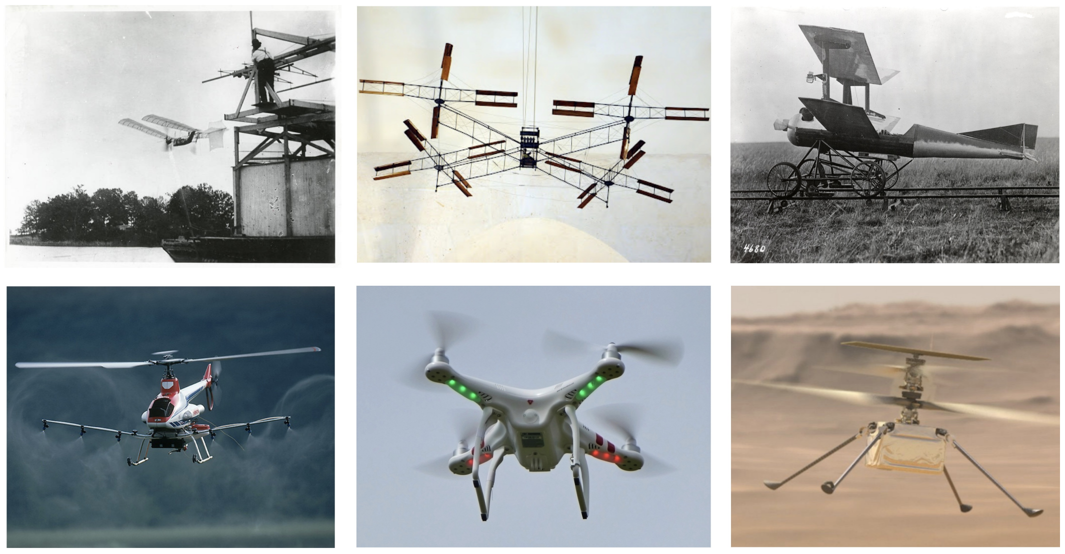


Figure 1.1: From the first powered flight to flight on Mars : a selection of unmanned aerial vehicles (UAVs) at different stages of their evolution. Top row: Samuel Langley's Aerodrome No. 5 (1896), The Bréguet-Richet Gyroplane (1907), The Kettering Aerial Torpedo "Bug" (1917). Bottom row: Yamaha R-50 (1987), DJI Phantom 1 (2013), NASA Mars Ingenuity Helicopter (2021).

1.1 Aerial Robotics: From Early Pioneers to Powered Flight on Mars

The Wright brothers' 1903 Flyer is rightly celebrated as the first manned, powered, heavier-than-air flight, marking the birth of modern aviation. Less widely noted is that unpiloted powered flight preceded this milestone: in 1896 Samuel Langley's Aerodrome No. 5 flew successfully [1]—seven years earlier. Powered by a compact steam engine driving two forward-facing propellers, the Aerodrome was launched from a catapult and flew without in-flight control, relying on its airframe configuration and a negative dihedral angle—modeled after the soaring of birds—for passive stability. Viewed through a robotics lens, this flying machine is a credible antecedent of the modern *aerial robot*, "a system capable of sustained flight with no direct human control and able to perform a specific task" [2]. Although the vocabulary of robotics would only emerge in the early twentieth century, the ambition to build self-regulating flying machines was already clear.

Building on the early explorations of flying machines, rotorcraft experiments were a natural next step. The Bréguet–Richet Gyroplane (1907) adopted a quad-rotor arrangement with counter-rotating propellers to balance the drag torque. Although stabilization and closed-loop control were still nascent, it demonstrated the feasibility of heavier-than-air lift by powered rotors, and can thus be credited as the precursor of modern quadcopter technology [3]. In parallel, unmanned fixed-wing systems were developed with operational aims in mind. World War I efforts such as the Kettering "Bug" (1918) reframed unmanned flight as a repeatable, pre-programmed procedure. The aircraft was launched from a track and stabilized by pneumatic and electrical controls, to ultimately guide it toward a target [4].

Widespread industrial adoption of unmanned aerial vehicles arrived much later. An early example was Yamaha's R-50/RMAX unmanned helicopters, released in the late 1980s–1990s. These were primarily used for agricultural spraying, surveying, and monitoring, establishing durable civil use cases [5]. Two decades later, the DJI Phantom (2013) was the first quadcopter that popularized the use of aerial robotics, by offering a ready-to-fly quadcopter equipped with automatic control, GPS, and camera capabilities—normalizing routine operation by non-experts and accelerating research and commercial adoption [6, 7].

Taken together, these examples mark a path from experimentation by early pioneers to routine civil use and widespread industrial adoption. In 2021, NASA/JPL's *Ingenuity*, extended this trajectory beyond Earth by achieving the remarkable feat of powered, controlled flight on Mars [8]. This expedition demonstrated how autonomy, propulsion systems, and mechanical design could be adapted to radically different, and often unknown environments. The final tally; Ingenuity survived for a total of seventy-two unpiloted flights on another planet.

This achievement inspires a continued effort in developing robotic systems with improved autonomy, endurance, and robustness. Here on Earth, terrestrial missions often demand capabilities that flight alone cannot supply—endurance over long traverses, safe operation near people and infrastructure, and reliable interaction with the ground. Indeed, predictions on the future of autonomous robotic systems point to the fact that many challenges facing today's autonomous systems may be relieved if we combine aerial and terrestrial capabilities [9, 10]. In the next section, we turn to the development of ground-aerial robotic systems, systems that leverage both modes of locomotion to broaden their operational envelope beyond single-mode flight.

1.2 Ground-Aerial Robotics

Ground aerial robotic systems are ideally poised to increase the reliability and scope of autonomous robotic missions. Whilst robots specially adapted to single locomotion types like quadcopters or legged robots may achieve excellent performance in their respective domains, they suffer from fundamental disadvantages. Aerial robots face important limitations due to battery life and payload capacity, whilst ground robots have limited ability to explore the aerial domain. Combining ground and aerial locomotion modes thus promises to deliver increased versatility, helping to transform applications such as last mile delivery [11, 12] or space exploration [13, 14].

The Benefit of Wheeled Locomotion

One way of achieving ground locomotion is by using wheels or rolling. Wheeled robots benefit from a low amount of energy consumed per unit distance traveled. This property, otherwise known as a low *cost of transport* [15], makes them ideal candidates for missions requiring long endurance.

Figure 1.2 shows theoretical estimates indicating that the overall range of a wheeled vehicle can be as much as ten times greater than an equivalent flying vehicle. As the fraction of time spent in flight grows, the benefit declines. For example, when spending 10% of time in flight, the maximum improvement in range is just below six times. A vehicle that spends 30% of time in flight sees an improvement in range between two and three times, depending on the ground speed. Although these estimates rely on simplified forward-flight and rolling-resistance models (detailed in Appendix A), similar estimates of the benefit of wheeled locomotion have been obtained experimentally [16]. Since wheeled-locomotion control is also much simpler than other locomotion types, for example walking or crawling, robots that combine flying and wheeled ground motion are becoming increasingly relevant to applications requiring long-endurance robotic autonomy.

Flying-Driving/Rolling Robots

These benefits have motivated researchers to develop novel flying-driving robot designs, varied in both their form factor as well as the choice of propulsion and actuation. The main challenge from a design point of view is finding how to effectively combine wheels and propellers into one economical and low-weight

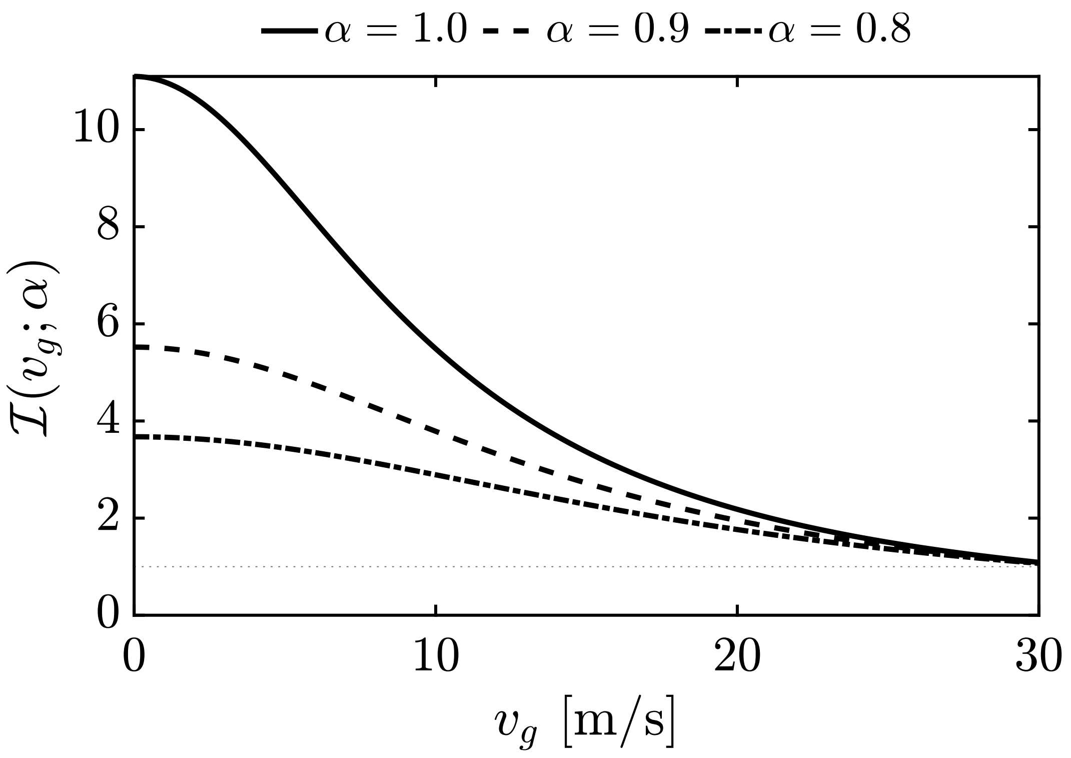


Figure 1.2: Range improvement when using bi-modal ground aerial locomotion compared to single-mode flight. The improvement in range, $\mathcal{I}(v_g;\alpha)$, is plotted as a function of the ground driving speed v_g for a few representative α values, i.e. the proportion of total time spent on the ground. In flight, the robot is assumed to fly at the speed that minimizes the cost of transport. The model uses parameters like mass, rotor diameter, number of rotors, and drag coefficient that are representative of the *Aerially Transforming Morphobot* presented in Chapter 2. For a full derivation the reader is referred to Appendix A.

package. A common approach found in the literature is to use passive wheels and achieve rolling by spinning the propellers when on the ground [20, 21, 22, 23]. In this method, the thrust generated by the propellers is used both for flight, and to initiate and sustain rolling when on the ground. This relieves the need for separate actuators for the wheels, but the use of the power-intensive propellers for both modes of locomotion reduces the overall energetic efficiency and practicality.

Another approach is to use actively driven wheels that are added onto the robot chassis [24, 25, 26]. Using specialized motors for driving increases the performance of ground locomotion, but can also increase the weight and overall form factor of the design, straining the propulsion system. In some cases, to reduce the weight of the wheels while protecting the propellers, spherical, motor-driven cages are placed

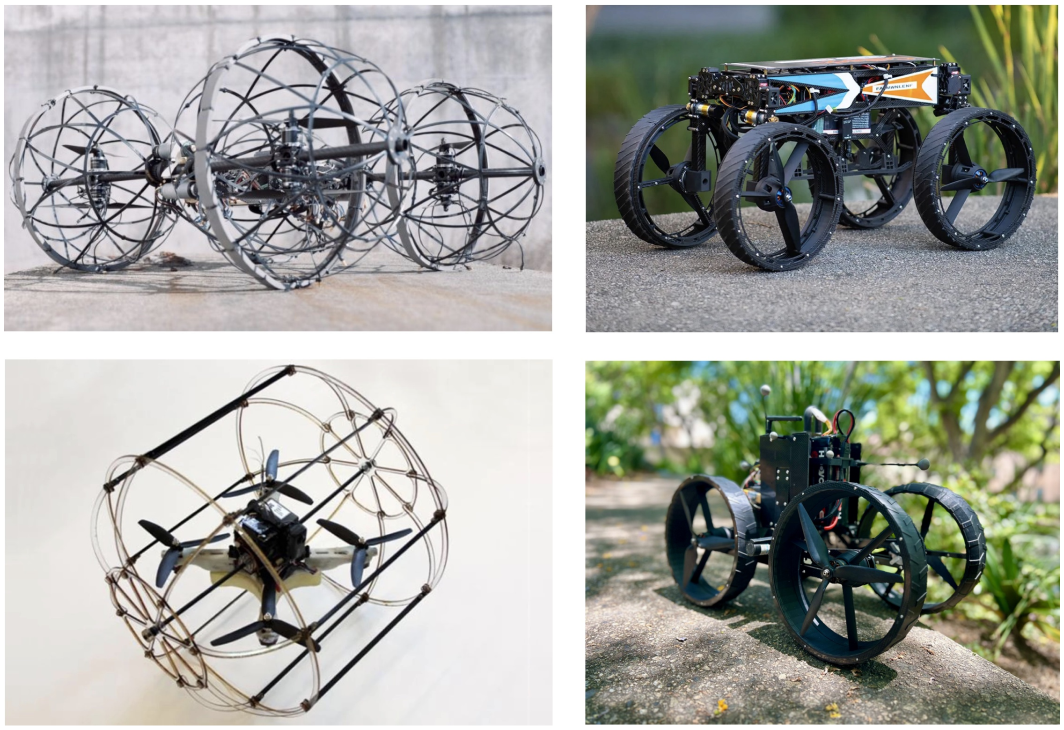


Figure 1.3: A selection of bi-modal flying-driving robots. Left column: Cages around propulsion to achieve driving. Drivocopter with actively driven wheels [17], HyTaQ which uses propellers for ground locomotion and flight [16]. Right column: Multi Modal Mobility Morphobot which uses shape change to achieve driving and flying [18] Aerially Transforming Morphobot—the platform presented in this thesis which uses aerial shape change to enhance multi-modal mobility [19].

around the propellers, enabling effective flight and ground mobility with minimal added structure [17, 27]. However, the spherical cage design increases the system volume and may be impractical in scenarios when a solid wheel may be needed, e.g. when driving over rough terrain. Some representative samples of flying-driving robot designs are depicted in the first column of Figure 1.3.

Morphobots and Shape Change

Most of the ground-aerial robots described above have relied on the use of redundant actuation. Other than the flying-driving robots that use the thrusters for both flight and driving locomotion, these designs employ multiple actuators that can perform one function only. However, these types of redundant robot designs often result in using more actuators and components than strictly necessary, increasing system weight and cost. A competing philosophy strives to reuse the same structural and actuation elements through shape change to generate different locomotion modes while reducing system weight and complexity [28, 18]. These robot designs, termed *Morphobots*, are generally thought to enhance the efficiency of mobile autonomous robots faced with changing, unstructured environments [29, 30, 31, 32, 33].

Recent work showed that using multi-functional appendages combined with body shape change resulted in increased *locomotion plasticity*—the ability of a single robot to reconfigure and re-purpose the same appendages to realize multiple distinct mobility modes [18]. The Morphobot used to demonstrate this, named M4 and depicted in Figure 1.3, incorporates propellers into wheels and uses servo motors with a pre-programmed joint-space sequence to enable switching between flying and driving modes. This dual actuator design results in weight and volume savings while the free motion of the wheel-thruster actuator enables simultaneous posture manipulation and thrust vectoring. This opens up new possibilities such as using thrust force from the propellers to increase the traction force when driving up steep inclines, or helping to balance when standing up on two wheels.

1.3 Aerial Transformation: Potential and Challenges

A key challenge for flying–driving Morphobots such as M4 is that mode switching has to occur on the ground, where vehicle–ground interactions are unpredictable and sometimes prohibitive. For example, debris or rough terrain can jam or overload appendages, preventing reconfiguration. Enabling transformation mid-air sidesteps these constraints: the robot can switch modes before touchdown, safeguarding critical actuators and yielding smoother, faster transitions. More broadly, aerial shape change can be used to pass through narrow openings or to deliberately alter control effectiveness, increasing agility and reducing maneuver time [34]. These capabilities are directly relevant to speed and precision-critical settings—from drone racing [35] to time-sensitive disaster response—and point to significant untapped potential for aerial transformation.

Despite these advantages, aerial transformation in Morphobots remains relatively underexplored. Recent work has shown quadcopters squeezing through narrow gaps [34, 36, 37, 38], perching to extend endurance [39, 40, 41], and exploiting morphology to tolerate actuator failures [42, 43], yet turning these demonstrations into a routine capability is nontrivial. During reconfiguration, mass distribution and inertias change, thrust vectors shift, and aerodynamic effects occur. Thus, the performance of controllers premised on a fixed geometry deteriorates; meeting mission requirements, such as maintaining precise hover while morphing, demands morphology-aware control. In practice this is less a matter of tuning and more of a complete redesign of the control stack.

Control Algorithms

The control problem in aerial transformation splits roughly into two regimes: (i) "in-plane" morphing, where thrusters move within the plane orthogonal to their thrust directions, and (ii) "out-of-plane" tilt, where thrust vectors depart from the body z-axis and induce strong cross-couplings. Both regimes must cope with shifting mass properties and wrench mappings during reconfiguration, but they place different demands on the control stack.

For in-plane reconfiguration, significant progress has been made on quadrotors that laterally reposition or pivot their thrusters to squeeze through gaps or to tune agility and efficiency. A feedback-linearizing, LQR body-rate controller was proposed in [38] where gains were adapted online to account for morphology-induced changes in center of gravity and inertia; desired torques and collective thrust are then mapped to actuator commands via a control-allocation matrix that depends on the instantaneous thruster angles. This structure integrates cleanly with the classical cascaded attitude and position loops used on fixed-frame quadrotors. A competing line of work employs adaptive body-rate control that updates parameters online to maintain performance given geometry changes [44].

Fewer studies address out-of-plane tilt, where thrust is no longer aligned with the body z-axis and standard cascaded PID architectures lose fidelity due to unmodeled lateral thrust components. In this domain, passively morphing designs that use springs to fold and traverse small gaps demonstrate the feasibility of out-of-plane geometries [36, 37]. However, sustained flight under out-of-plane tilt further requires configuration-aware allocation and attitude control, a topic that remains comparatively underdeveloped.

Finally, mode-transition control (e.g., flight-to-touchdown) introduces its own challenges. Proximity to the ground introduces ground-effect aerodynamics [45, 46, 47] that reshape the thrust profile and can complicate autonomous landing. Several approaches to mitigate this have been explored for quadcopters, including physics-based modeling with feedforward/feedback compensation or learning-based strategies that adapt to the changing flow field [48, 49, 50, 51]. Extending these ideas to ground–aerial systems with richer posture control and mid-air reconfiguration remains an open area.

1.4 Approach and Objectives

This thesis aims to develop the control methods, robot hardware, and design principles that make aerial transformation a valuable tool for air–ground mobility. Our ultimate goal is to enable novel ground-aerial robotic capabilities—switching modes before touchdown, protecting actuators from uncertain ground contact, and deliberately modulating control effectiveness to improve agility and efficiency—while deepening our understanding of the physical principles, control methods, and aerodynamics necessary to accomplish this.

Our approach is deeply rooted in experimentation. We design and build a Morphobot specialized for mid-air reconfiguration, *the Aerially Transforming Morphobot* (ATMO), which builds on previous robot designs but incorporates key simplifications that facilitate aerial transformation. Centered around this robotic platform we conduct two parallel investigations. On the one hand we investigate the platform's dynamics, characterizing and contrasting it to existing aerial robots such as quadcopters, and push its limits by developing optimization and learning-based controllers that account for the morphology-dependent dynamics. On the other hand we scrutinize the robot design, and apply aerodynamic principles to expand its feasible operating region and improve morphing flight efficiency. In both cases, our major claims are supported by benchtop experimentation, physical simulation, system integration testing, and full scale robot experiments.

Our objectives are: (i) develop and validate control strategies that enable morphing flight and reliable air-ground transitions using morphing; (ii) quantify how mid-air reconfiguration impacts agility while identifying practical regimes where it is beneficial; (iii) establish the dynamics and aerodynamics of aerial morphing via first-principles modeling, experimental load cell testing, flow visualization, and computational studies; (iv) understand how fluid dynamics can be harnessed to improve thrust recovery and control authority during aerial shape change.

Wherever possible we have released the tools developed in this thesis to the community via open-sourcing—facilitating reproduction and adoption of our work by researchers making steps into the field of aerial transformation control.

1.5 Thesis Outline and Contributions

Chapter 2 presents the Aerially Transforming Morphobot (ATMO)—a novel flying-driving Morphobot specially designed for mid-air transformation. The major design choices, such as choice of morphing mechanism, electronics stack, and actuator sizing for both morphing and wheel-drive systems are presented. The kinematics of morphing are derived and the propulsion system is characterized experimentally. We then investigate the aerodynamics of *morpho-transition*, or the act of transitioning from flight to ground through a phase of mid-air shape change, using benchtop load-cell measurements and qualitative flow visualization. These experiments show that ground-proximity aerodynamics are altered substantially when propellers are tilted, and that the four-rotor, inward-facing configuration used by ATMO exhibits interaction effects that are not captured by standard ground-effect models.

Chapter 3 develops a first-principles dynamic model for a *morphing quadcopter*– our modeling abstraction of ATMO. A near-hover linearization reveals a fundamental structural difference between the morphing quadcopter and the classical quadcopter: out-of-plane morphing introduces coupling between translational states and attitude (notably roll), invalidating standard separations exploited by off-the-shelf cascaded quadcopter controllers. This analysis motivates a re-design of the control architecture. The chapter also establishes the dynamic model of morphing flight that is used by the controllers developed in subsequent chapters.

Building on these insights, **Chapter 4** introduces a model predictive controller (MPC) that stabilizes ATMO across three regimes: conventional quadcopter flight, morphing flight, and morpho-transition. The controller employs a state and configuration-dependent cost to account for the shifting control effectiveness as the robot morphs and includes constraints that reflect actuator limits. A supervisory logic coordinates the air-to-ground transition maneuver, where ATMO morphs near the ground to land on wheels, using configuration and state thresholds informed by the aerodynamic and dynamic analyses of Chapters 2 and 3. A set of experiments demonstrates closed-loop stability and repeatable transitions, demonstrating tracking performance without the need for inner tracking control loops.

Chapter 5 explores a complementary strategy based on deep reinforcement learning (RL) for end-to-end control of morpho-transition. Leveraging a massively parallel simulation environment and targeted domain randomization, we train policies that transfer zero-shot to hardware for the transition task. A key practical component is the explicit treatment of actuator dynamics and delays in training, which we contrast with the MPC design that achieves hardware transfer without an explicit delay model due to its feedback structure. This study highlights which randomizations and observation choices are most salient for successful transfer, providing guidelines for sim-to-real transfer in learning-based aerial transformation control.

Chapter 6 revisits ATMO's design and aims to address the central drawback of mid-flight reconfiguration: tilting thrusters during flight results in reduced vertical thrust and control authority. We develop a method that uses a passive deflecting surface to redirect the rotor wake of the tilted thruster and to partially recover the thrust lost during morphing. This concept is implemented using internal deflectors that are integrated directly into the robot chassis and that intercept and redirect the rotor wake downward, recovering momentum that would otherwise be wasted. This electronics-free approach achieves up to 40% recovery of vertical thrust in configurations where useful vertical force would otherwise be minimal, substantially extending hover margin and maneuvering capability during transformation. We characterize the effect experimentally and provide design guidelines such as the influence of the deflector exit angle for application to related morphing platforms.

Finally, **Chapter 7** develops a theoretical framework for aerial posture control aimed at agility and time-critical maneuvers. Generalizing beyond ATMO's single primary morphing degree of freedom, we consider a morphing quadcopter model that can rotate each arm with two degrees of freedom and study how deliberate posture changes reshape control authority and maneuver time. We solve the minimum time trajectory generation problem under this generalized dynamic model and demonstrate, in simulation, how careful aerial posture control can reduce maneuver duration.

Chapter 2

THE AERIALLY TRANSFORMING MORPHOBOT: DESIGN AND AERODYNAMICS

This chapter incorporates material from the following publication:

Ioannis Mandralis et al. "ATMO: an aerially transforming morphobot for dynamic ground-aerial transition". In: *Communications Engineering* 4.1 (Apr. 2025). ISSN: 2731-3395. DOI: `10.1038/s44172-025-00413-6`. URL: `http://dx.doi.org/10.1038/s44172-025-00413-6`. (published).

The integration of ground and aerial locomotion into a single, dynamically transforming robotic platform introduces substantial mechanical design challenges. Additional actuation inevitably increases system weight and complexity, while careful selection of transformation actuators is required to ensure reliable and sustained operation. This chapter provides a detailed breakdown of the robot's key subsystems, as well as a characterization of performance, used for the development of model-based controllers in subsequent chapters. We begin by outlining the major design decisions behind the morphing mechanism, the wheel drive, and the onboard electronics. The propulsion system is then described and characterized, and experimentally obtained thrust and moment coefficients for the selected motor–propeller combination are reported. The chapter concludes with an investigation of the aerodynamics of the system when morphing near the ground. We present quantitative data on how ground proximity and propeller tilting affect overall thrust production, offering insight into how ground-effect aerodynamics are modulated by the robot's unique aerial configuration.

2.1 Robot Overview

The Aerially Transforming Morphobot (ATMO), depicted in Figure 2.1, was specially designed to achieve bi-modal flying and driving locomotion while solving the problem of mid-air transformation and smooth transition between modes. Building on the Multi-Modal Mobility Morphobot [18], we substantially streamline the architecture: where the prior system uses eight posture-manipulation actuators, ATMO achieves morphing with a single mechanism. This consolidation simplifies the state

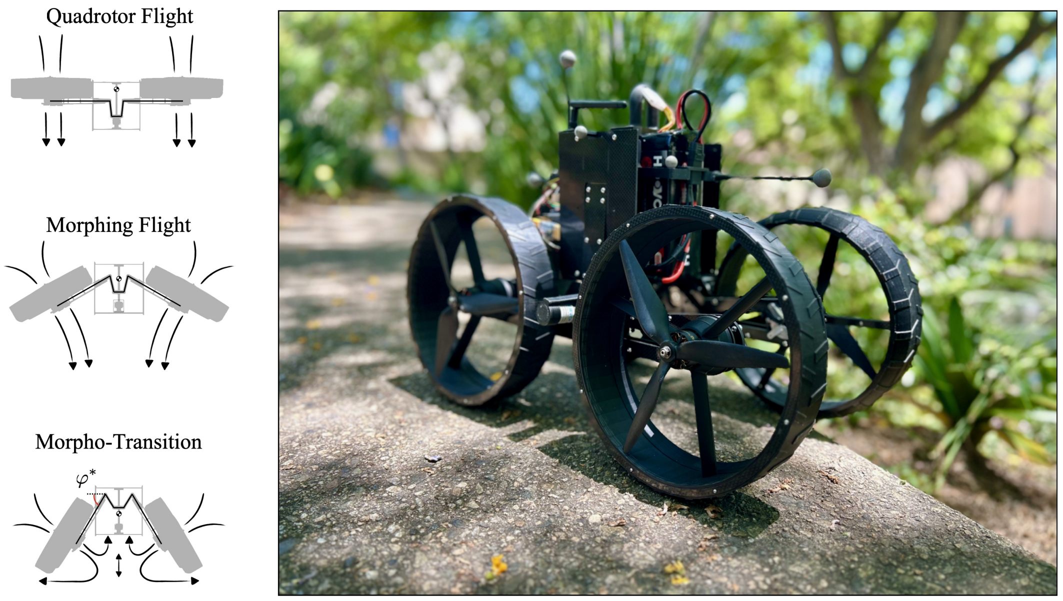


Figure 2.1: Aerially Transforming Morphobot (ATMO) Overview. A picture of ATMO in an outdoor environment in driving mode next to a schematic of the robot capabilities.

machine and control architecture, reduces mass, and lowers potential failure points.

To enable both flight and ground locomotion with minimal hardware, we adopt the wheel–thruster actuator introduced in [18], which integrates a propeller–motor unit into a driven wheel. This dual-purpose actuator supplies thrust in the air and wheeled motion on the ground, allowing ATMO to reconfigure between flying and driving through transformation. In the aerial configuration, ATMO behaves as a conventional quadrotor, commanding propeller speeds for thrust and attitude control; in the ground configuration, the wheels are rotated to produce forward motion and steering.

The final result is a compact robot that weighs 5.5 kg (including the battery). It stands 16 cm tall and 65 cm wide in aerial configuration, and 33 cm tall and 30 cm wide in ground configuration. The vehicle mass is kept low by fabricating the chassis, arms, and wheel components from 2-mm carbon-fiber plates and 3D-printed parts. A fiber-inlay process on a Markforged 3D printer with Onyx thermoplastic offers an excellent strength-to-weight ratio and ease of use, enabling rapid prototyping and fabrication. ATMO is equipped with an onboard computer running a custom controller, as well as onboard sensors for state estimation and fusion. A video overview of the robot's capabilities is available here.

2.2 Morphing Mechanism

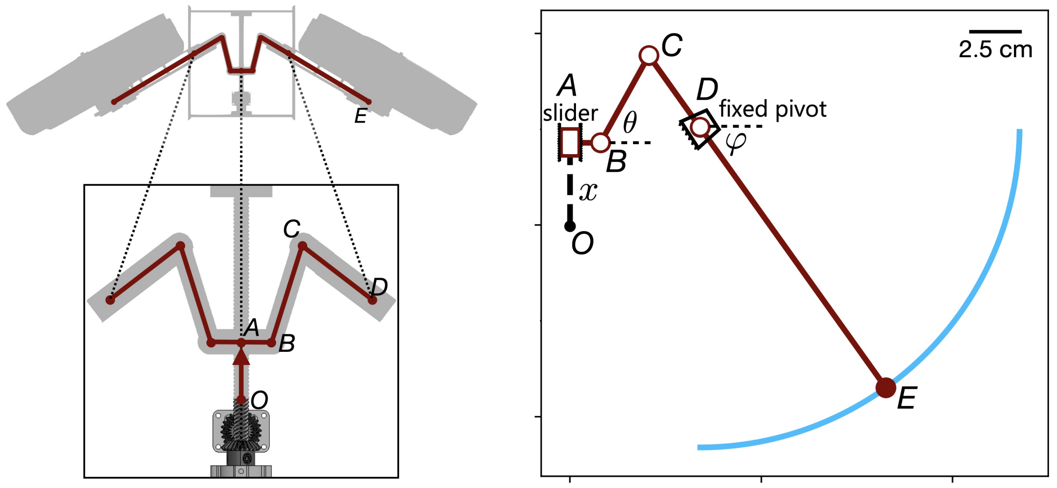


Figure 2.2: A closeup schematic of the morphing mechanism geometry and relevant variables. (left) ATMO is shown in a morphed configuration in flight. Underneath, the tilt actuator box is enlarged and the joints are labeled. The tilt mechanism is actuated by two co-rotating bevel gears spun by a DC motor. The spinning causes joint A to translate on the OA axis, inducing mechanism motion. (right) The right half of the symmetric mechanism is shown with all the joints labeled as well as joint E which represents the center of rotation of the propeller. The path taken by joint E as joint A moves from bottom to top is traced in blue. The key kinematics parameter is the tilt angle φ which varies from $\varphi = 0$ in flight configuration to $\varphi = \frac{\pi}{2}$ in drive configuration. θ is an internal angle of the mechanism and x is the displacement of A with respect to O.

The morphing mechanism, depicted in Figure 2.2, dynamically controls the tilt angle φ of four wheel-thruster pairs using a single motor. It works using a closed-loop kinematic linkage actuated by a motor that rotates a central power screw. This rotation results in linear motion of joint A which is then converted to rotational motion of joints B, C, D, and E—tilting the four wheel-thruster actuators simultaneously.

The power screw is of length 100 mm and made of stainless steel with lead $l = 8$ mm per revolution (i.e. total linear distance traveled by the nut due to one rotation), outer diameter $d = 8$ mm, and pitch $p = 2$ mm. The nut which rotates on the power screw is made of brass, leading to a friction coefficient of approximately $f = 0.4$ between the brass and the steel surfaces. Links AB, BC, and CD are machined from stainless steel. The power screw is fit with a bevel gear which attaches to the shaft using a set screw and rotated by an identical bevel gear attached to the motor shaft. The point D can rotate freely due to two ball bearings attached to each side of the robot chassis which enable rotational motion.

By virtue of the properties of the power screw mechanism, the posture of the robot locks in place even when the tilt motor is not being actuated. This allows ATMO to fly in morphed configurations without actively providing power to the motor—reducing the overall power draw of the system and improving battery life and range. The self-locking property also protects the system from the possibility of mid-flight actuator failure, which could result in sudden loss of desired posture. Overall, using a single actuator to control the body shape change, results in a simple mechanical system with few failure points, while the symmetric and simultaneous tilting simplifies the aerial transformation control problem and limits synchronization issues between either side of the robot.

Kinematics

Using knowledge of the position of the pivot point D relative to the mechanism origin O, the kinematics of the closed loop linkage are computed by solving the following system for θ, φ as a function of the linear displacement of the first link $x = ||OA||$,

$$\begin{bmatrix} D_x \\ D_y \end{bmatrix} = \begin{bmatrix} h \\ x \end{bmatrix} + \begin{bmatrix} \cos\theta & \cos\varphi \\ \sin\theta & -\sin\varphi \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}, \tag{2.1}$$

where θ is an internal mechanism angle, φ is the tilt angle, and $h = ||AB||$, $d_1 = ||BC||$, and $d_2 = ||CD||$ are the mechanism link lengths as depicted in Figure 2.2. The numerical values of the parameters of the morphing mechanism are given in Table 2.1.

h	1.6 cm
d_1	5.2 cm
d_2	4.6 cm
D_x	6.8 cm
D_y	5.1 cm

Table 2.1: Kinematic parameters of morphing mechanism.

The kinematic equations were validated using a DC motor with an encoder to rotate the power screw in discrete rotation bouts and measuring the angle φ using an electronic angle gauge. The number of encoder counts of the motor, which produced $n_e = 1632.67$ counts per revolution, were recorded using an Arduino micro-controller. The kinematic equations were also solved numerically for the tilt angle (φ), using the relationship $x = lN$, where N is the number of revolutions of the power screw, and $l = 8$ mm is the power screw lead value. The tilt angle was then plotted against the number of encoder counts, n, using the conversion $n = n_e N$. The analytical prediction of the angle is compared against measured tilt angle values at various encoder counts and shown in Figure 2.3. The good agreement validates the kinematic equations of motion.

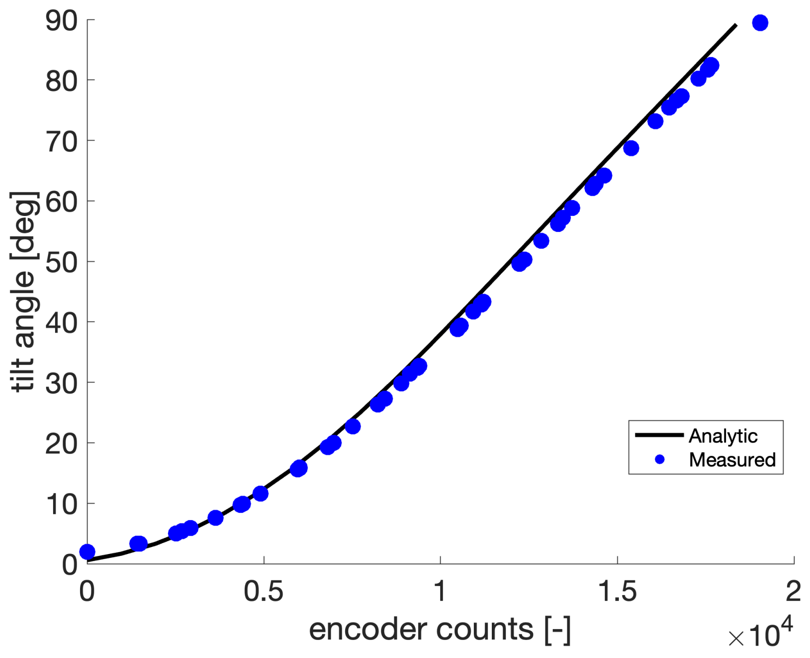


Figure 2.3: Validation of kinematic equations against encoder data and ground truth angle measurements. An experimental validation of the closed loop kinematic equations which predict the tilt angle φ (in solid black) against the number of encoder counts of the actual motor (blue markers).

The validated kinematic equations were used to obtain the transformation speed as a function of the motor rotational speed, ω. Implicitly differentiating Equation (2.1) and solving for $\dot{\varphi}$ yields

$$\dot{\varphi} = \frac{l\omega \sin\theta}{||CD|| \sin(\theta + \varphi)}, \tag{2.2}$$

where $\dot{x} = l\omega$.

Motor Selection

To select the motor which spins the power screw, we must consider the amount of torque needed to rotate the screw when the mechanism is under external load. For motor sizing, we consider the worst case scenario where all four thrusters are producing the maximum thrust $T_{\max}$. The external load from the thrusters produces

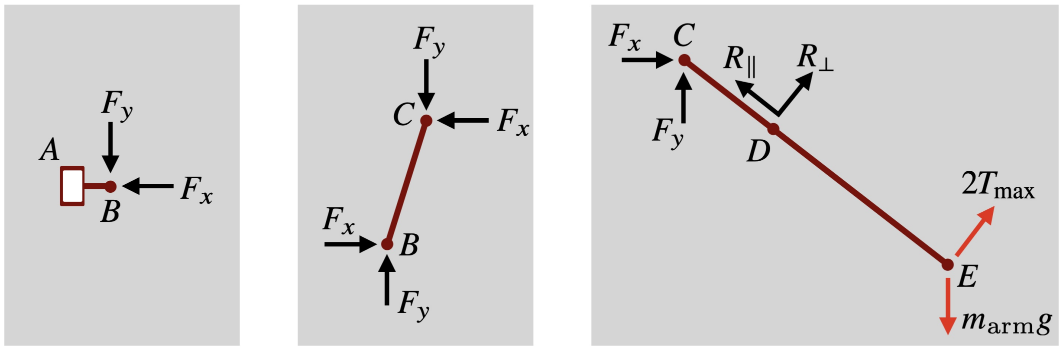


Figure 2.4: Free body diagram of symmetric half of morphing mechanism.

a force of magnitude F on the central joint A, which is directly related to the torque required to actuate the mechanism using power screw theory [52].

To compute the loading force F, a free body analysis of the symmetric half of the mechanism is performed. As can be seen in Figure 2.4, two external forces act on link CDE: the force of two thrusters at maximum thrust $2T_{\max}$, and the weight of the arm $m_{\text{arm}}g$, which is assumed to be concentrated at point E. These external loads result in a reaction force F_y at joint B. Assuming all links are in static equilibrium, a balance of moments on link CDE around the pivot point D yields

$$\sum_{CDE} \tau_D = \frac{||DE||}{||CD||}(2T_{\max} - m_{\text{arm}}g\cos\varphi) - (F_y\cos\varphi + F_x\sin\varphi) = 0. \tag{2.3}$$

The moment balance on link BC around joint B further relates F_y and F_x:

$$\sum_{BC} \tau_B = ||BC||(F_y\cos\theta - F_x\sin\theta) = 0 \tag{2.4}$$

$$\implies F_y = F_x\tan\theta. \tag{2.5}$$

Combining Equations (2.5) and (2.3) allows the total vertical force acting on link A to be expressed in terms of $\varphi, \theta, T_{\max}, ||DE||, ||CD||$ and m_{arm}

$$\boxed{F = 2\frac{||DE||}{||CD||}\frac{(2T_{\max})\tan\theta - m_{\text{arm}}g\cos\varphi}{\tan\theta\cos\varphi + \sin\varphi}.} \tag{2.6}$$

To get the total axial loading force F we have used the fact that each half of the morphing mechanism contributes a force F_y to the joint A i.e. $F = 2F_y$. Taking into account a friction coefficient of $f = 0.40$, the screw lead $l = 8$ mm, the outer diameter $d = 8$ mm, and pitch $p = 2$ mm, i.e. $d_m = d - p/2 = 7$ mm, and ignoring the small frictional force caused by the rotation of the power screw in the bearings, the torque τ_R required to raise link A against the external load predicted by power screw theory is

$$\tau_R = \frac{F d_m}{2}\left(\frac{\pi f d_m + l}{\pi d_m + f l}\right). \tag{2.7}$$

Likewise, to lower link A in the same direction as the external load F, the torque τ_L is

$$\tau_L = \frac{F d_m}{2}\left(\frac{\pi f d_m - l}{\pi d_m + f l}\right). \tag{2.8}$$

These two torques are plotted as functions of the tilt angle φ in Figure 2.5, using $T_{\text{max}} = 2.9$ kg, $m_{\text{arm}} = 1.537$ kg, $g = 9.81$ ms^{-2}, $||DE||= 13.67$ cm, and $||CD||=$ 4.60 cm. To remove the dependency of F on θ, the latter was expressed as a function of φ by rearranging Equation (2.1):

$$\theta = \arccos\left[\frac{D_x - h - \cos\varphi d_2}{d_1}\right]. \tag{2.9}$$

Our analysis shows that the torque required by the motor to move link A against the influence of the axial load is just under 10 kg cm, while moving link A in the same direction as the axial load is positive at all angles—indicating that the mechanism is indeed self-locking, as desired.

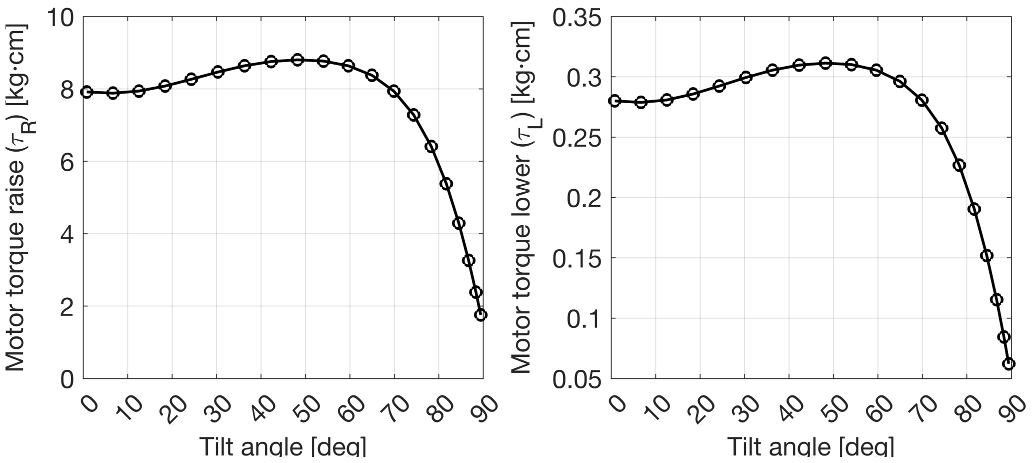


Figure 2.5: Motor torque required for folding under maximum thrust load.

Given this information, the Pololu 47:1 Metal Gearmotor with a stall torque of around 15 kg cm was selected, giving the system a considerable transformation safety factor. This motor is very lightweight and cheap, improving overall flight performance; using servo-motors instead of a custom tilt mechanism is possible but would require heavier and more expensive actuators to deliver the torque required to transform the robot in the worst case scenario.

Transformation Speed

Finally, we can compute the time it takes to transform from flight to drive configuration by solving the ordinary differential equation,

$$\dot{\varphi} = \frac{l\omega_n \left(1 - \frac{\tau_R(\theta,\varphi)}{\tau_s}\right) \sin\theta}{||CD||\sin(\theta + \varphi)}, \tag{2.10}$$

which is obtained from Equation (2.2) using $\omega = \omega_n(1 - \frac{\tau_R}{\tau_s})$. Here, ω_n is the motor no-load speed and τ_s is the motor stall torque. These values are obtained from the motor manufacturer: $\omega_n = 220$ RPM and $\tau_s = 15$ kg cm. The dependency on θ was removed using Equation (2.9). Equation (2.10) was integrated numerically using the `ode45` solver. The total time required to vary the tilt angle from $\varphi = 0°$ to $\varphi = 90°$ is approximately $T = 2.0$ s as shown in Figure 2.6.

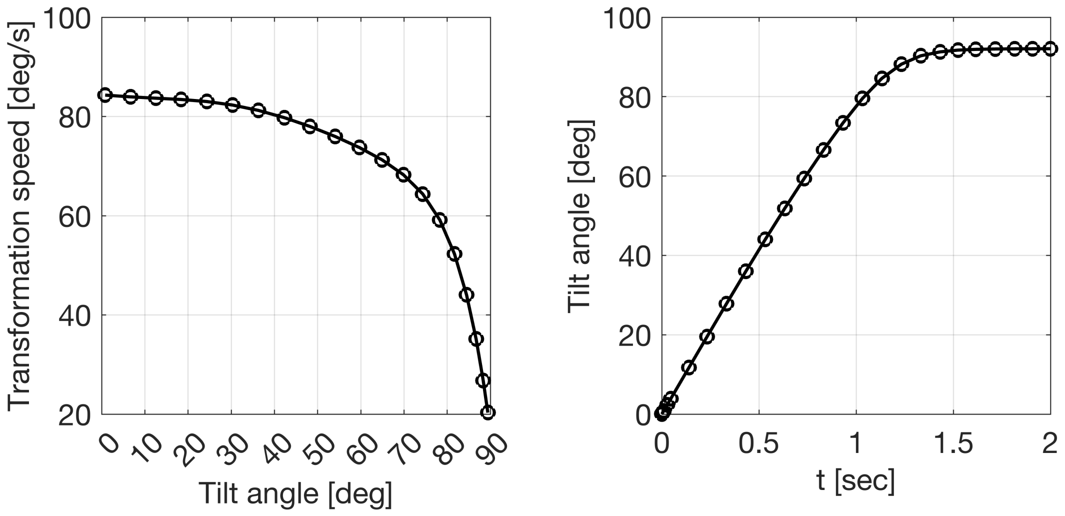


Figure 2.6: (left) Transformation speed versus tilt angle. Time taken to transform from flight to drive configuration using selected motor.

2.3 Propulsion System

The wheel-thruster actuators each contain a motor-propeller combination, which rotates independently from the wheel and produces the thrust necessary for flight. A three-blade propeller design with diameter $D_p = 22.86$ cm, propeller pitch $\alpha_p = 12.7$ cm, and a brushless motor with $k_V = 1155$ (RPM per Volt under no load conditions) were used. The motors are controlled using the APD80F3v2 Electronic Speed Controllers (ESCs) which receive Pulse Width Modulation (PWM) signals from a CubeOrange flight controller running the PX4 firmware [53] and power each brushless motor. A 6S1P battery pack with 8 Ah capacity and 100 C burst discharge rate was used, allowing the battery to discharge up to 800A of current in short bursts. The wheel around the propeller plays the dual role of enabling driving but also acting as a shroud for protection. There are potential performance benefits to this configuration [54] but characterizing and optimizing these has been left for future work.

To characterize the thrust performance of the propulsion system, we performed experimental load cell tests. The motor-propeller combination was mounted onto an RC Benchmark load cell and the thrust and moment parallel to the propeller rotation were measured for rotational speeds between zero and maximum. The motor-propeller combination produced $T_{\max} = 2.9$ kg of maximum thrust and $M_{\max} = 0.5$ Nm of maximum torque. The maximum rotational speed was $16'388$ RPM. Thus, in fully charged battery conditions, the lift-to-weight ratio of the robotic system is approximately 2.

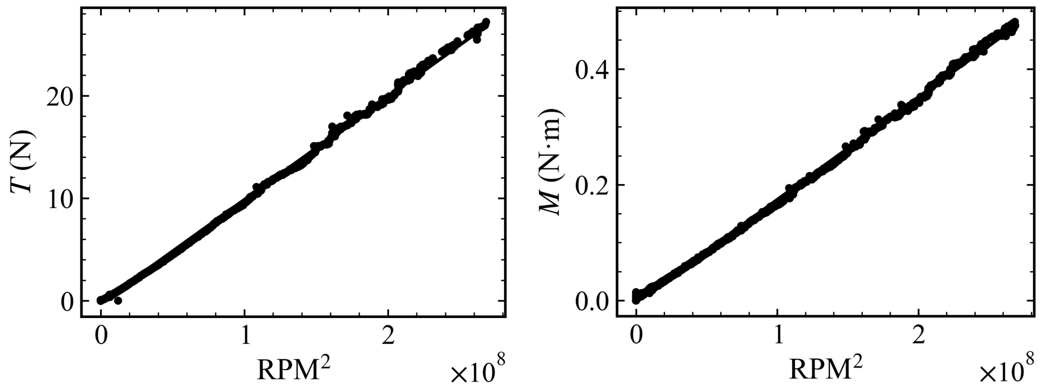


Figure 2.7: Thrust characterization of motor-propeller combination onboard the Aerially Transforming Morphobot. Thrust and moment values are plotted against the square of the propeller rotational speed.

To obtain thrust and moment coefficients, the propeller rotational speed was swept between zero and maximum and the thrust and moment were continuously measured and displayed as a function of the square of the rotational speed, Ω^2, in Figure 2.7.

The relationship between thrust/moment and the squared rotational speed is linear:

$$T(\Omega) = \tilde{k}_T\Omega^2 \equiv k_T\Omega^2 \equiv \bar{k}_T\frac{\Omega^2}{\Omega^2_{\text{max}}} \tag{2.11}$$

$$M(\Omega) = \tilde{k}_M\Omega^2 \equiv k_T k_M\Omega^2 \equiv \bar{k}_T\bar{k}_M\frac{\Omega^2}{\Omega^2_{\text{max}}}. \tag{2.12}$$

The coefficients are identified from the data: $\tilde{k}_T = 1.04 \times 10^{-7}$ N per RPM and $\tilde{k}_M = 1.82 \times 10^{-8}$ Nm per RPM. We use several equivalent parameterizations of this relationship throughout the thesis. A form convenient for some derivations employs k_T and k_M, with $k_T = \tilde{k}_T$ and $k_M = \tilde{k}_M/\tilde{k}_T$. In terms of the normalized propeller speed $\Omega^2/\Omega^2_{\text{max}}$ (where $\Omega_{\text{max}} = 16{,}388$ RPM), we define $\bar{k}_T = \tilde{k}_T\,\Omega^2_{\text{max}}$ and $\bar{k}_M = \tilde{k}_M/\tilde{k}_T$.

2.4 Wheel Drive and Chassis

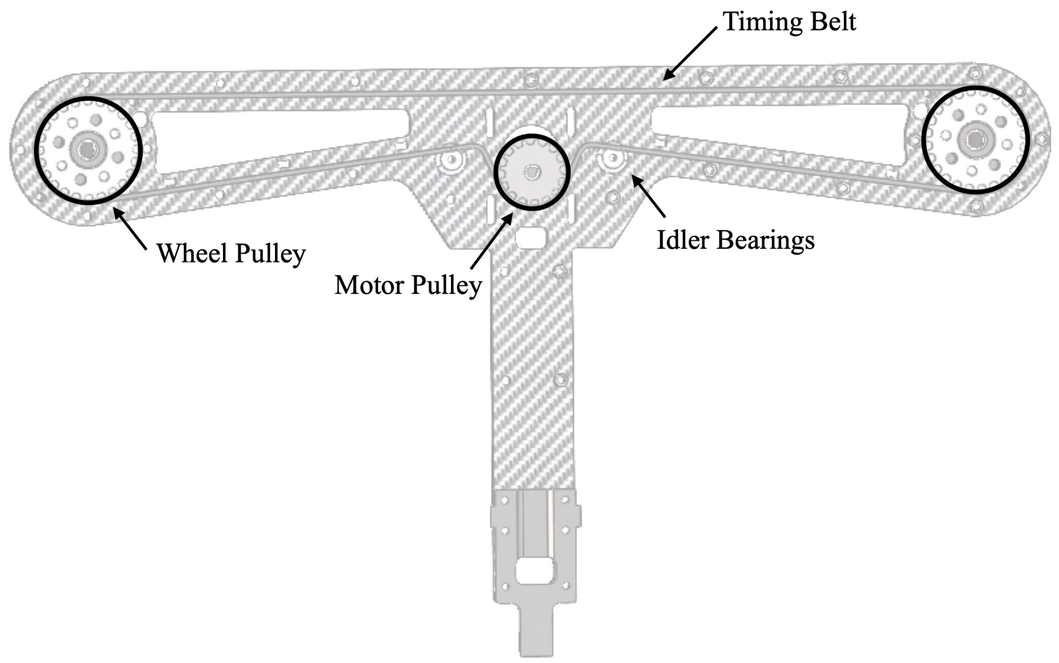


Figure 2.8: Cross section of the wheel drive mechanism. A belt-pulley system is held between two flat carbon fiber plates and actuated by a central motor that spins the motor pulley.

Driving is achieved by two belt-pulley systems on either side of the robot, enabling differential drive steering. The wheel drive mechanism is depicted in Figure 2.8. The wheels are 3D printed in PLA material and embossed with narrow grooves to increase ground traction for differential drive steering. The 3D printed frame is reinforced by thin rings of carbon fiber for additional rigidity. There are two brushed DC motors which drive the motor pulley on either side of the robot. The timing belt transfers the motion to two wheel pulleys, rotating both wheels on either side of the robot simultaneously. Differential drive steering is performed by direct pulse-width-modulation (PWM) control of the DC motors using a radio controller or a ROS2 control node.

Motor Selection

To select the drive motors, we consider the total force that must be applied to accelerate the robot by $a_d = 2.5$ m/s^2, taking into account the gear ratio between the motor pulley and the wheel pulley $G = 1.5$, the wheel radius $R_w = 0.125$ m, and the wheel mass $m_w = 0.30$ kg. We assume a coefficient of rolling resistance of $C_{rr} = 0.1$, and a conservative drivetrain efficiency of $\eta_g = 0.8$. Under these assumptions, the total force required to accelerate the robot by a_d is

$$F_d = ma_d + C_{rr}mg, \tag{2.13}$$

and the torque that must be applied on either side of the robot to achieve this is

$$T_d = \frac{1}{2}F_d R_w + m_w R_w a_d, \tag{2.14}$$

where the moment of inertia of the two wheels on each side has been considered, assuming they are solid disks (i.e. inertia is $I_w = \frac{1}{2}m_w R_w^2$), and the angular acceleration was computed from the desired acceleration assuming no slip: a_d/R_w. Finally, the motor torque is obtained using the mechanism gear ratio, G, and the assumed drivetrain efficiency, η_g:

$$T_m = \frac{T_d}{G\eta_g}. \tag{2.15}$$

Using the numerical values, the motor torque is $T_m = 10.93$ kg cm. We selected the Servocity spur gear motors that can deliver up to 22 kg cm of torque, at a rotational rate of 170 revolutions per minute, resulting in a safety factor of ≈ 2.

2.5 Electronics

The electronics stack is based on a central onboard computer, the NVIDIA Jetson Orin Nano. This uses the Robot Operating System 2 (ROS2) [55] to communicate with the motor drivers actuating the tilt actuator and driving motors, as well as the flight controller, and the electronic speed controllers (ESCs) which actuate the thruster-propeller pairs. ROS2 enables running a central, real-time, control loop that

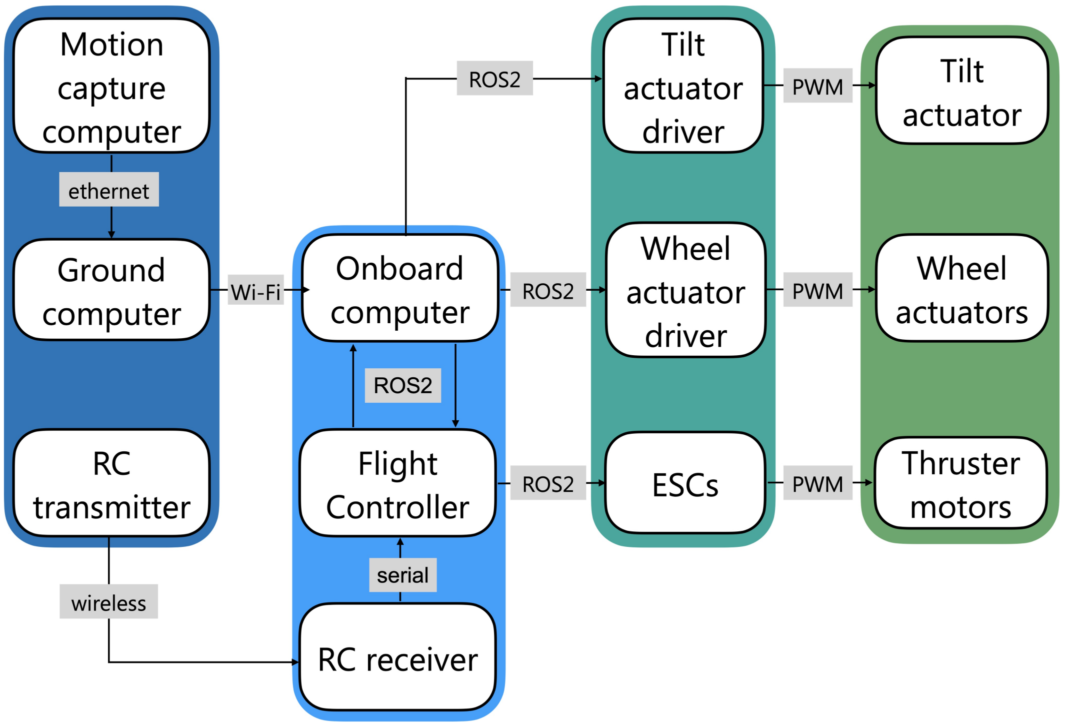


Figure 2.9: Electronics architecture and connections onboard robotic system. On the left are ground control components. These communicate to the onboard components by Wi-Fi or wireless (radio protocol) transmission. The onboard computer receives radio signals from the flight controller and communicates to all the motor drivers using ROS2.

receives inputs from an RC transmitter/receiver pair, as well as pose (3D position and orientation) data from a motion-capture ground computer running the Optitrack Motive software. With this information, the central control loop does the necessary computations and sends signals to the motor drivers through ROS2, which in turn send pulse-width modulation signals (PWM) to the respective actuators. More specifically, the pose information being streamed by the Motion Capture ground computer is sent to the CubeOrange flight controller (flashed with a custom version of the PX4 autopilot firmware) via the UXRCE-DDS protocol which translates ROS2 messages to PX4 compatible uORB messages. This pose data is given as an input to the Extended Kalman Filter (EKF) which runs onboard the CubeOrange. This fuses the Motion Capture pose information with the onboard IMU information, as well as other available sensor information (magnetometer, barometer, etc.) to provide a real-time state estimate. The PX4 EKF runs a delayed fusion time horizon EKF to allow for different time delays relative to the IMU.

Onboard Computer	NVIDIA Jetson Orin Nano
Flight Controller	CubeOrange+
Tilt Motor	Pololu 12V High Power 47:1 (4845)
Wheel Motors	Servocity Econ Motors 170 RPM
DC Motor Drivers	Roboclaw 2x15A motor controller
Thruster Motors	T-Motor Cine Series 1155 Kv
ESCs	APD 80F3v2
Propellers	HQ 9x5x3
Battery Pack	Hoovo 6S1P 8000 mAh 100C

Table 2.2: Electronic components list. A list of the electronic components used for the onboard control, communication, and actuation.

2.6 Near Ground Transformation Aerodynamics

Approaching the ground is known to alter the thrust characteristics of rotorcraft [56, 57, 58, 59]. This phenomenon is typically known as the ground effect and is defined as the percent of additional thrust gained due to ground proximity. In a seminal study by Cheeseman and Bennet, a simple theoretical model of ground effect for a single rotor was obtained by representing the rotor as a source and using the method of images to model the interaction with the ground [60]. Using this simplified model, the relationship between the thrust in ground effect T_{IGE} and the thrust outside of ground effect T_{OGE} is predicted as

$$\frac{T_{\text{IGE}}}{T_{\text{OGE}}} = \frac{1}{1-(R/4z)^2}, \tag{2.16}$$

where R is the propeller radius, and z is the distance from the propeller to the ground. This theory predicts experimental measurements reasonably well for a single rotor in ground effect, but falls short in predicting the behavior of multiple rotors in ground effect. ATMO's aerodynamic configuration, i.e. four inward-facing propellers approaching the ground, has not yet been studied.

To bridge this gap, we measured how the thrust produced by ATMO changed as it approached the ground while the tilt angle was varied. Our testing rig is summarized in Figure 2.10. To position ATMO at the desired distances from the ground, we mounted it onto a robotic arm that enables free translation and rotation. A single-axis S-type load cell was placed in between ATMO and the mounting point on the robotic arm to measure the thrust values. The load cell was calibrated against known weights and its axis was carefully aligned with the direction of gravity using an electronic angle gauge as reference. The distance between ATMO's base frame and the ground was streamed in real time using the OptiTrack measurement system.

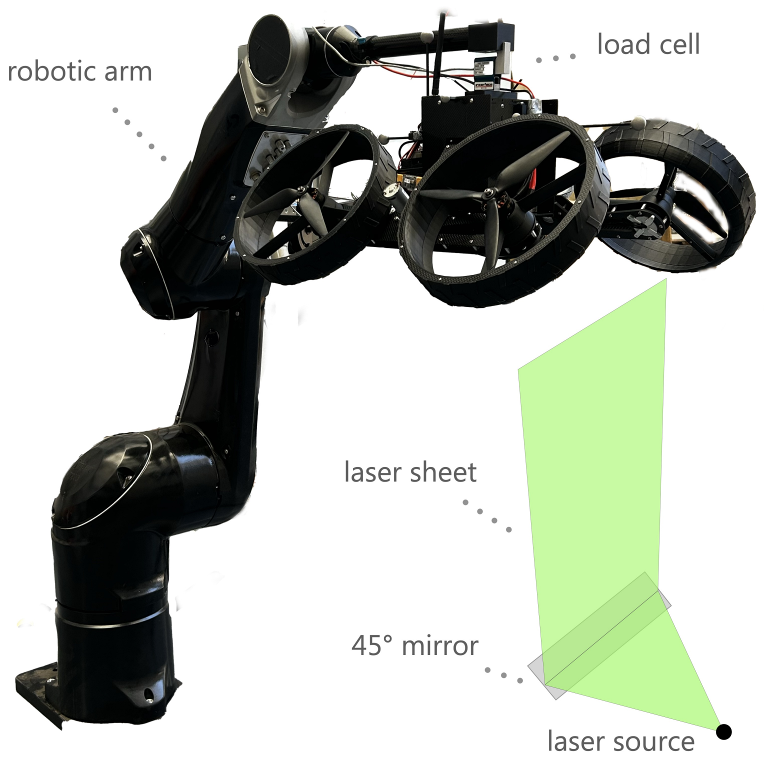


Figure 2.10: Aerodynamics of aerial transformation experimental setup. A six axis robotic arm is used to adjust the three dimensional position of ATMO. A load cell is in series between the robotic arm and ATMO. A laser sheet is positioned underneath the plane of the front two thrusters for imaging purposes. To achieve this a laser source is shone on a 45 degree mirror generating a vertical sheet which constitutes the imaging plane.

The tilt angle was computed from the encoder counts using the kinematic equations described in Section 2.2 and recorded after each experiment.

For each data point, the thrusters were set to 50% of the maximum thrust value. The thrust, T, measured by the load cell was recorded for $t_f = 15$ second intervals to produce the set $\{T(t, \varphi, z)|\ t \in [0, t_f]\}$ for each φ and z considered. After each trial the average thrust $\bar{T}(z, \varphi)$ was recorded, as well as the standard deviation in time of the thrust during the trial. The normalized thrust values reported in Figure 3 were obtained as $\frac{\bar{T}_{\text{IGE}}(z,\varphi)}{\bar{T}_{\text{OGE}}(\varphi)}$ where $\bar{T}_{\text{OGE}}(\varphi)$ is the average thrust measured at the load cell far from the ground. In order to maintain the voltage steady across experiments, a 24V power supply was used instead of the onboard 6S1P LiPo battery.

Our measurements are reported in Figure 2.11. The ground effect for angles $\varphi = 40°, 50°, 60°$, is evidenced by the increasing thrust as the robot approached the

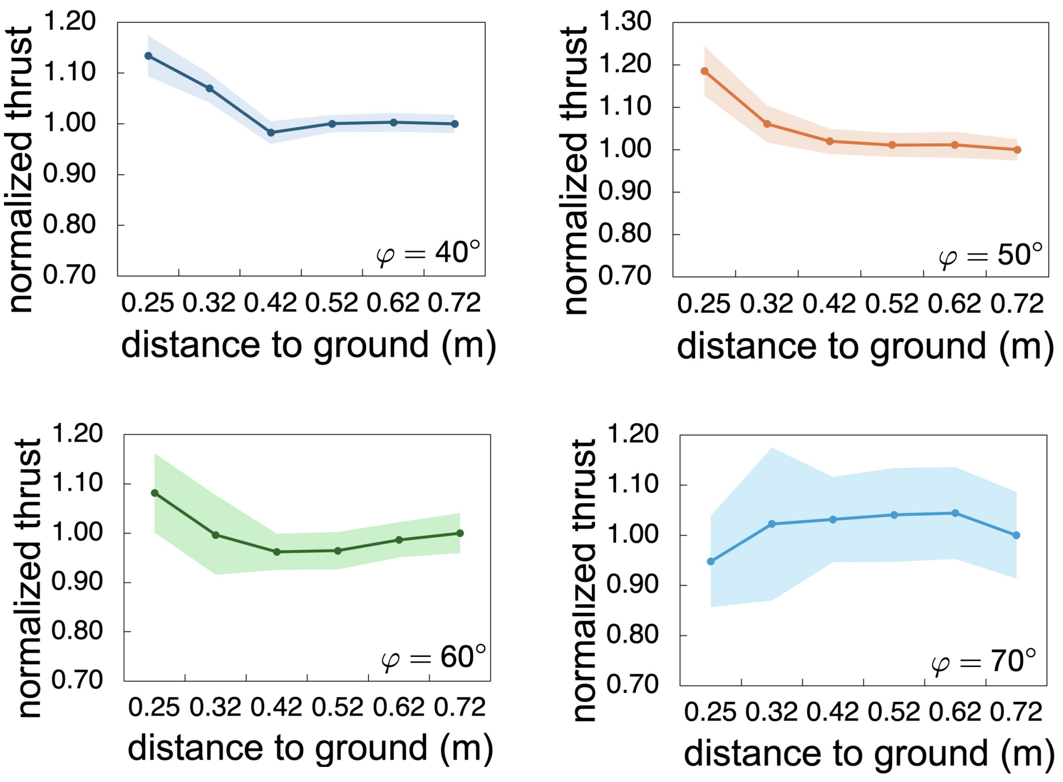


Figure 2.11: Ground effect subject to propeller tilting. Results from the load cell testing for $\varphi = 40°, 50°, 60°, 70°$. The overall thrust (measured at the load cell) normalized by the thrust produced in the same configuration far from the ground is plotted at six different heights $z = 0.25, 0.32, 0.42, 0.52, 0.62, 0.72$m. The standard deviation of the thrust during the measured time horizon is shaded in lighter color for each experiment.

ground. The largest ground effect was found at $\varphi = 50°$ with an increase of thrust of almost 20%. The trend of increased thrust with ground effect did not persist at larger tilt angles. For $\varphi = 70°$, approaching the ground resulted in a loss of thrust. Entering this state is dangerous since the loss of thrust will result in a larger than anticipated impact velocity. Furthermore, the time variability of the thrust was much higher for $\varphi = 70°$ compared to the other angles as seen by observing the increase in area of the shaded portion in Figure 2.11 for $\varphi = 70°$. Taking into account the standard deviation, we observe that the thrust force can fall to as low as 85% of the thrust away from the ground as the robot approaches the ground for $\varphi = 70°$.

We performed smoke-visualization of the flow field at different angles to examine the underlying factors that influence landing stability. The smoke visualization was done using a laser placed on the flat ground shining onto a 45° mirror in order to create a laser sheet in the plane of the front two robot thrusters. The smoke was supplied using standard fog machines into one of the middle planes between the front two thrusters. The fog machines were held on the robot axis of symmetry between the two front thrusters to ensure that smoke was entrained into both sets of propellers. The movement of the smoke was observed over all possible tilt angles as seen in Supplementary Videos 4 and 5.

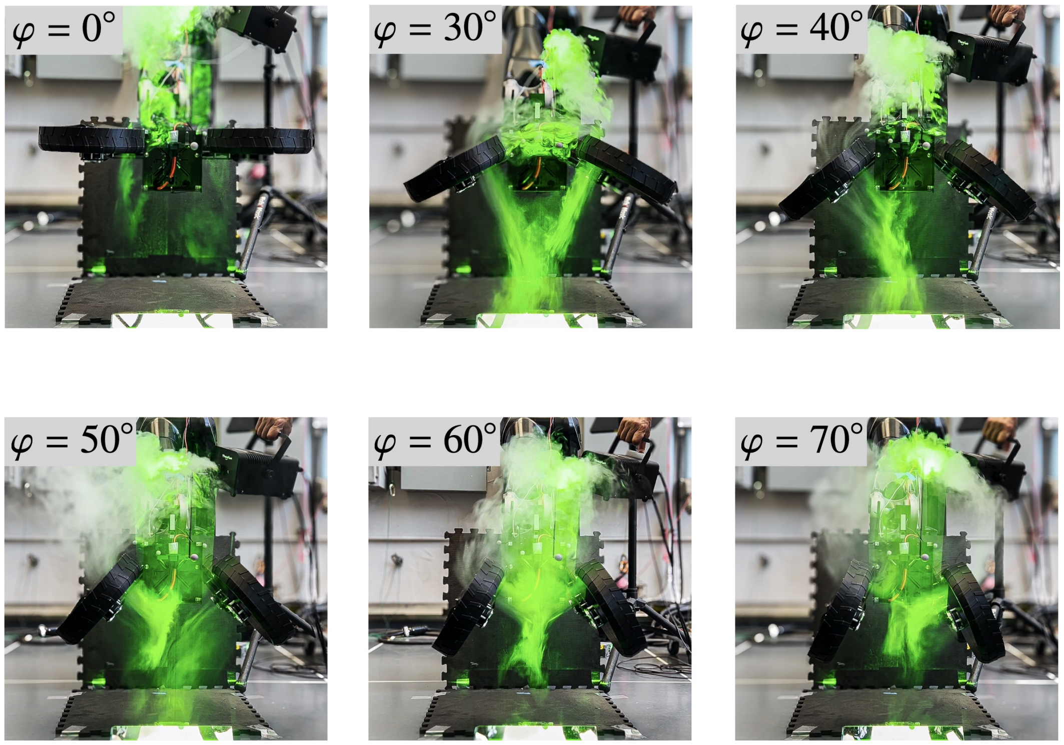


Figure 2.12: Smoke visualizations of the aerodynamic flow field with the robot stationary at different tilt angles. For φ = 0° two streams of air are flowing vertically through the wheel-thrusters. For φ = 30° the two jets are reoriented and mix to form one downward oriented stream. At φ = 70° the two jets impinge to form a stagnation point with an unstable region where the flow may be directed either upwards or downwards. Arrows indicated the overall direction of the flow as observed in the videos of the visualization (see supplementary materials).

The smoke visualization presented in Figure 2.12 reveals that up to φ = 60°, the interaction of the two rotor flows results in a net downward transfer of momentum since the streams of air combine by mixing and accelerating away from the robot body. Conservation of linear momentum dictates that the net thrust is oriented upwards. Beyond φ = 60°, e.g. at φ = 70°, this trend reverses; the interaction of the two air jets becomes unstable, resulting in a significant portion of the flow being re-oriented in the undesired direction and a loss of thrust. Furthermore, the instability of the air jet interaction, which can be seen in Supplementary Videos 7 and 8, causes a greater time variability of the thrust, increasing dynamic uncertainty.

2.7 Summary

This chapter presented the core systems that constitute the Aerially Transforming Morphobot (ATMO). We introduced a compact morphing mechanism with inherent self-locking that reduces actuation complexity and system failure points. We derived the kinematic and dynamic models needed to analyze the mechanism, and characterized the propulsion system. Finally the near ground transition aerodynamics were documented, revealing a novel relationship between propeller tilting and ground effect. In the next chapter, we model the dynamics of ATMO from first principles building on the propulsion characteristics and mechanism kinematics presented here. The near ground morpho-transition aerodynamics are revisited in Chapter 4 when developing a model predictive controller that can achieve dynamic ground aerial transition by morphing near the ground and landing on wheels.

Chapter 3

DYNAMICS OF AERIALLY TRANSFORMING QUADCOPTERS

Successful control of the Aerially Transforming Morphobot (ATMO) requires understanding its dynamics and how it differs from conventional quadcopters. In this chapter, we develop a dynamic model of ATMO using the *morphing quadcopter* abstraction. We begin by reviewing propeller aerodynamics, motor dynamics, and the translational and rotational equations for a fixed-frame quadcopter, then extend these dynamics to a morphing quadcopter that can tilt its propellers beneath its body. Using a near-hover linearization, we analyze both systems and show that the cascaded control architecture widely used in quadcopter control is not applicable to the morphing case. This analysis reveals a fundamental structural change in the dynamics induced by out-of-plane tilting. Finally, we introduce the *critical angle*—the tilt at which thrusters saturate when attempting to hover—a key parameter for control and stabilization revisited in subsequent chapters. In Appendix B we have included a proof of *differential flatness* for the morphing quadcopter model. Differential flatness is an important property of nonlinear dynamical systems that can be exploited to develop fast trajectory generation algorithms. The proof further highlights the inherent structural differences between quadcopter and morphing quadcopter dynamics.

3.1 Propeller Aerodynamics

A spinning propeller produces forces and torques by moving air through the *propeller disk* i.e. the area swept by the rotating wing. These forces are highly dependent on the geometry of the propeller and the flight regime, for example, whether the aircraft is in hover or translational flight. In general four forces are significant: the thrust force $\boldsymbol{T}$, the drag torque $\boldsymbol{M}$, the hub force $\boldsymbol{H}$, and the rolling torque $\boldsymbol{R}$. For a propeller in hover, the two dominant forces are the thrust force and the drag torque,

$$T = \frac{1}{2}c_T\rho A(\Omega R)^2 \equiv k_T\Omega^2 \tag{3.1}$$

$$M = \frac{1}{2}c_M\rho A(\Omega R)^2 R \equiv k_T k_M\Omega^2, \tag{3.2}$$

where R is the propeller radius, $A = \pi R^2$, Ω is the propeller rotational speed, and c_T and c_M are the thrust and moment coefficients respectively. In Equations (3.1) and (3.2), all of the constants have been grouped into two constants k_T and k_M which depend on the propeller geometry and the Reynolds number. The thrust and moment coefficients were reported in Chapter 2.

In forward flight the hub force and the rolling moment become significant. The hub force opposes the horizontal flight direction and the rolling moment acts around the body $\boldsymbol{x}_B$ axis. They are given by

$$H = \frac{1}{2} c_H \rho A (\Omega R)^2 \tag{3.3}$$

$$R = \frac{1}{2} c_R \rho A (\Omega R)^2 R. \tag{3.4}$$

The coefficients c_H and c_R depend on the advance ratio $V/(\Omega R)$ where V is the forward velocity. For low advance ratios these forces are typically neglected since the thrust and drag forces dominate [61].

3.2 Motor Dynamics

The motors which drive the propeller shafts in order to generate the desired thrust can be represented as electro-mechanical subsystems which receive a desired angular velocity value $\Omega_d \in [0, \Omega_{\max}]$ and spin the propellers at an angular rate Ω which is the measured output. Although motor drives can be formally modeled using the physical principles of electrical motors, it is common to instead assume the dynamics are a first order system with time constant T_m. Denoting $\boldsymbol{\Omega}_d$ as the vector of desired motor RPMs and $\boldsymbol{\Omega}$ as the vector of propeller rotational rates, the motor dynamics can be written in vector form as

$$\dot{\boldsymbol{\Omega}} = \frac{1}{T_m}(\boldsymbol{\Omega}_d - \boldsymbol{\Omega}). \tag{3.5}$$

The constants $\Omega_{\max}$ and T_m depend on the motor-propeller combination and are determined via experimentation.

3.3 Quadcopter Dynamics

A quadcopter is a particular instantiation of a *rotorcraft* i.e. an aircraft which produces lift by spinning a propeller in the horizontal plane. The quadcopter uses four spinning propellers that are arranged on the vertices of a rectangular frame and spun by a direct-drive mechanism. The propellers are arranged in pairs of two and are counter-rotating to balance the torque on the body.

The quadcopter is modeled as a rigid and symmetric structure whose center-of-gravity is coincident with the origin of the body frame $\{\boldsymbol{x}_B, \boldsymbol{y}_B, \boldsymbol{z}_B\}$. The force of

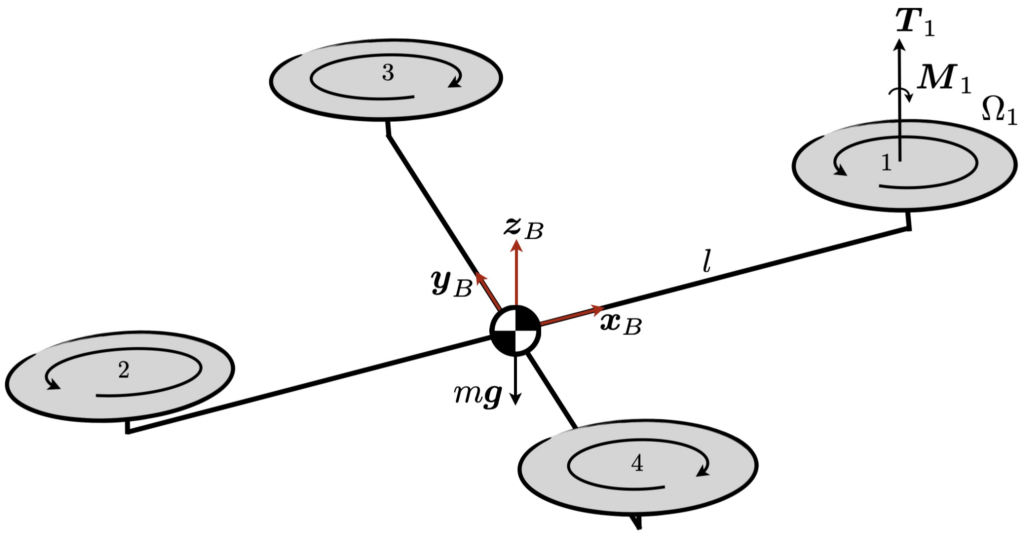


Figure 3.1: Schematic of a quadcopter in flight. The base frame is aligned with the center of gravity. There are four propellers arranged in pairs with opposite directions of rotation.

gravity is $m\boldsymbol{g}$, where $\boldsymbol{g}$ is expressed in the world frame. Each spinning propeller produces a thrust force $\boldsymbol{T}_i$ aligned with the the body-frame axis z_B and a drag moment $\boldsymbol{M}_i$ which opposes the direction of the propeller rotational velocity. The propeller rotational velocity is denoted Ω_i. The body-frame axis $\boldsymbol{x}_B$ points in the direction of the first propeller. Propellers 1 and 2 are assumed to rotate counter-clockwise while propellers 3 and 4 rotate clockwise. A schematic of a quadcopter in flight is depicted in Figure 3.1.

It is assumed that the translational velocity of the quadcopter is low allowing us to neglect fuselage drag as well as the hub force and rolling moment. The propeller is assumed rigid and interaction with the ground or other surfaces is neglected. The translational dynamics of the quadcopter can hence be written as

$$m\ddot{\boldsymbol{p}} = -mg z_W + \left(\sum_{i=1}^{4} T_i\right) z_B, \tag{3.6}$$

where $\ddot{\boldsymbol{p}}$ is the position of the quadcopter center-of-gravity in the world frame $\{\boldsymbol{x}_W, \boldsymbol{y}_W, z_W\}$. The rotational dynamics can be written as

$$\boldsymbol{I}\dot{\omega} + \omega \times \boldsymbol{I}\omega = \sum_{i=1}^{4} \left[\boldsymbol{\rho}_i \times T_i z_B + M_i z_B\right] \equiv \boldsymbol{\tau}, \tag{3.7}$$

where the angular momentum due to the spinning propellers has been neglected since the propeller inertias are typically very small. The angular velocity of the quadcopter with respect to the world frame, $\boldsymbol{\omega}$, is expressed in the body frame. Furthermore, the body frame is assumed to be aligned with the principal axes of inertia of the quadcopter and the body inertia is denoted by $\boldsymbol{I}$. The external torque, $\boldsymbol{\tau}$, acting on the right hand side of Equation (3.7) is composed of the torque due to the thrust forces which act at vector displacement $\boldsymbol{\rho}_i$ away from the center-of-gravity, as well as the drag torques caused by the propeller spinning.

We obtain a complete representation of the quadcopter dynamics by writing Equations (3.6) and (3.7) in state-space form. The degrees of freedom of the quadcopter are its position $\boldsymbol{p}$, and its *attitude* i.e. the orientation of the quadcopter body relative to the world frame. To parameterize this space of rotations we use Euler ZYX angles and denote the yaw, pitch, and roll angles by $\boldsymbol{\theta} = (\psi, \theta, \phi)^\top$. The rotation from the body to the world frame is given by the matrix $\boldsymbol{R}$, explicitly stated in Appendix C. Thus, in state-space form, the quadcopter equations of motion are

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{3.8}$$

$$\dot{\boldsymbol{v}} = -g z_W + \left(\frac{1}{m}\sum_{i=1}^{4} T_i\right) z_B \tag{3.9}$$

$$\dot{\boldsymbol{\theta}} = \boldsymbol{E}^{-1}\boldsymbol{R}\omega \tag{3.10}$$

$$\dot{\omega} = \boldsymbol{I}^{-1}\left(\boldsymbol{\tau} - \omega \times \boldsymbol{I}\omega\right) \tag{3.11}$$

$$\dot{\boldsymbol{\Omega}} = T_m^{-1}(\boldsymbol{\Omega}_d - \boldsymbol{\Omega}). \tag{3.12}$$

In Equation (3.10) we have used the relationship between the angular velocity of the quadcopter expressed in the body frame and the time derivatives of the yaw, pitch, and roll angles which is given in Appendix C. Substituting the thrust and torque in terms of the propeller rotational speeds using Equations (3.1)–(3.2) results in the quadcopter dynamics in state-space form with $\boldsymbol{x} = (\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{\theta}, \omega, \boldsymbol{\Omega}) \in \mathbb{R}^{16}$ and control inputs $\boldsymbol{u} = \boldsymbol{\Omega}_d \in \mathbb{R}^4$:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). \tag{3.13}$$

3.4 Morphing Quadcopter Dynamics

Similar to the quadcopter, a morphing quadcopter is controlled primarily via the spinning of four propellers, but the quadcopter frame can now change shape mid-flight. We consider specifically the case where the quadcopter can fold the right and left hand side of the frame downward and symmetrically like the Aerially Transforming Morphobot, as sketched in Figure 3.2. Although other types of morphing quadrotor configurations have been considered in the literature, as was

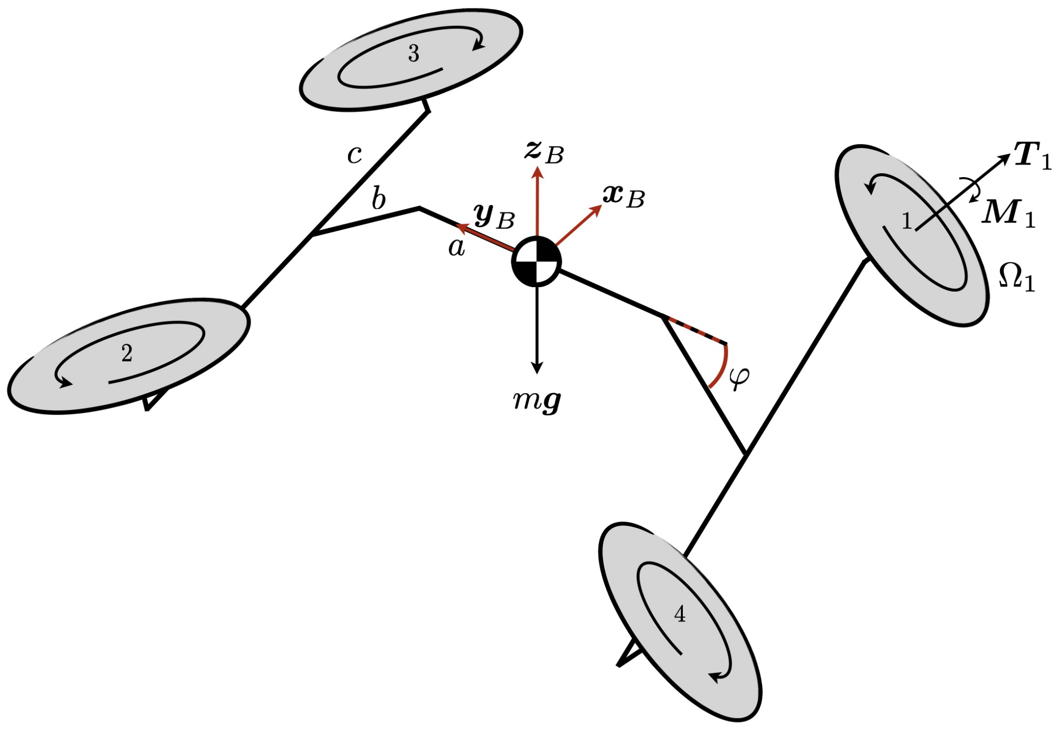


Figure 3.2: Schematic of the morphing quadcopter model in flight with forces and geometry labeled. The base frame is defined in line with the point about which the arms tilt. A representative center-of-gravity is depicted but shifts during flight depending on the configuration. The half-width of the morphing-quadcopter body is a, the arm length is b, and the half length of each arm structure is c. Four counter-rotating propellers are used.

described in Chapter 1, the specific case of symmetric, out-of-plane propeller motion has not been studied.

The morphing quadcopter comprises seven inertial bodies: the base link, two arms that pivot about the base, and the four spinning propellers. The wheels surrounding the propellers are excluded from the dynamic model because they do not contribute to stabilization during morphing flight. Using the wheels as additional control inputs could aid stabilization, but we leave this to future work. As the vehicle changes shape during morphing, both the center-of-mass location and the inertia matrix vary with configuration. A systematic way to capture these effects is the floating-base multibody formulation, which is widely used for quadrupedal and bipedal robots and is supported by a well-developed theory [62]. In this form, the equations of motion are

$$\boldsymbol{M}(\boldsymbol{q})\,\dot{\mathbf{w}} + \boldsymbol{b}(\boldsymbol{q}, \mathbf{w}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{f} + \boldsymbol{J}_{\text{ext}}^{T}\boldsymbol{F}_{\text{ext}}, \tag{3.14}$$

and include the following variables:

$\boldsymbol{M}(\boldsymbol{q})$	$\in \mathbb{R}^{n_q \times n_q}$	Mass matrix (orthogonal)
$\boldsymbol{q}$	$\in \mathbb{R}^{n_q}$	Generalized coordinates
$\mathbf{w}$	$\in \mathbb{R}^{n_q}$	Generalized velocity
$\dot{\mathbf{w}}$	$\in \mathbb{R}^{n_q}$	Generalized acceleration
$\boldsymbol{b}(\boldsymbol{q}, \mathbf{w})$	$\in \mathbb{R}^{n_q}$	Coriolis and centrifugal terms
$\boldsymbol{g}(\boldsymbol{q})$	$\in \mathbb{R}^{n_q}$	Gravitational terms
$\boldsymbol{f}$	$\in \mathbb{R}^{n_f}$	Generalized forces and torques
$\boldsymbol{F}_{\text{ext}}$	$\in \mathbb{R}^{n_c}$	External forces
$\boldsymbol{J}_{\text{ext}}$	$\in \mathbb{R}^{n_c \times n_q}$	Geometric Jacobian of location of external forces.

The morphing quadcopter has three translational and three rotational degrees of freedom, excluding the tilt angle. This leads to the generalized coordinates and velocities: $\boldsymbol{q} = (\boldsymbol{p}, \boldsymbol{\theta})$ and $\mathbf{w} = (\boldsymbol{v}, \boldsymbol{\omega})$ where $\boldsymbol{p}$ is the position of the robot base frame. The position of the base frame is chosen to be at the same height as the pivot on the base link to simplify the equations of motion, as depicted in Figure 3.2. In general the location of the base frame and the location of the center-of-gravity do not coincide. $\boldsymbol{\theta}$ is the orientation of the robot relative to the inertial world frame, expressed as a vector of Euler ZYX angles, $\boldsymbol{v}$ is the velocity of the center of mass, and $\boldsymbol{\omega}$ is the angular velocity of the robot relative to the world frame expressed in the body frame. In the morphing quadcopter model no external forces are considered so $\boldsymbol{F}_{\text{ext}} \equiv \mathbf{0}$.

The generalized forces $\boldsymbol{f}$ are constructed by projecting the force and moment generated by the spinning propellers into the generalized coordinate space. This is achieved using the geometric position and rotation Jacobians, $\boldsymbol{J}_{P_i}$ and $\boldsymbol{J}_{R_i}$, calculated at the center of mass frames of rotor $1, \ldots, 4$, and expressed in the world frame. $\boldsymbol{J}_{\text{armr/arml}}$ and $\boldsymbol{J}_{\text{armr/arml}}$ are likewise the Jacobians of the center-of-gravity of right or left arm link expressed in the world frame. Taking into account the action of each propeller on the rotor link as well as the reaction on the arm, we obtain

$$\boldsymbol{f}_1 = \left[\left(\boldsymbol{J}_{P_1}^T - \boldsymbol{J}_{P_{\text{armr}}}^T\right) - k_M\left(\boldsymbol{J}_{R_1}^T - \boldsymbol{J}_{R_{\text{armr}}}^T\right)\right] k_T \Omega_{\max}^2 u_1 \boldsymbol{n}_1 \tag{3.15}$$

$$\boldsymbol{f}_2 = \left[\left(\boldsymbol{J}_{P_2}^T - \boldsymbol{J}_{P_{\text{arml}}}^T\right) - k_M\left(\boldsymbol{J}_{R_2}^T - \boldsymbol{J}_{R_{\text{arml}}}^T\right)\right] k_T \Omega_{\max}^2 u_2 \boldsymbol{n}_2 \tag{3.16}$$

$$\boldsymbol{f}_3 = \left[\left(\boldsymbol{J}_{P_3}^T - \boldsymbol{J}_{P_{\text{arml}}}^T\right) + k_M\left(\boldsymbol{J}_{R_3}^T - \boldsymbol{J}_{R_{\text{arml}}}^T\right)\right] k_T \Omega_{\max}^2 u_3 \boldsymbol{n}_3 \tag{3.17}$$

$$\boldsymbol{f}_4 = \left[\left(\boldsymbol{J}_{P_4}^T - \boldsymbol{J}_{P_{\text{armr}}}^T\right) + k_M\left(\boldsymbol{J}_{R_4}^T - \boldsymbol{J}_{R_{\text{armr}}}^T\right)\right] k_T \Omega_{\max}^2 u_4 \boldsymbol{n}_4. \tag{3.18}$$

Here $\boldsymbol{n}_i$ is the axis of rotation of the i-th propeller, function of the tilt angle φ, and Equations (3.1) and (3.2) have been used with the auxiliary control variables,

$$u_i = \frac{\Omega_i^2}{\Omega_{\max}^2} \in [0, 1]. \tag{3.19}$$

The total generalized force is given by the sum of the four propeller generalized forces:

$$\boldsymbol{f} = \sum_{i=1}^{4} \boldsymbol{f}_i. \tag{3.20}$$

The tilt angle φ is not considered as a direct actuated variable as would be common practice in a robot dynamics setting. Instead, φ is a parameter and enters the dynamic equations as a kinematic variable modeled as a pure integrator. This reflects the physical properties of the non-compliant, self-locking morphing mechanism described in Section 2.2.

The mass matrix $\boldsymbol{M}(\boldsymbol{q})$, coriolis and centrifugal terms $\boldsymbol{b}(\boldsymbol{q})$, and the gravitational terms $\boldsymbol{g}(\boldsymbol{q})$ in Equation (3.14) can be derived using a number of rigid body dynamics libraries. We opted to use the open-source package proNEu [63] which provides explicit symbolic representations—needed for high performance implementation of a model-predictive controller described in the next chapter—of the dynamic equations of motion, given a kinematic tree and appropriate definitions of external wrenches. Using this library, the equations of motion for the morphing quadcopter can be obtained in state space form

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{3.21}$$

$$\dot{\boldsymbol{\theta}} = \boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{\omega} \tag{3.22}$$

$$\mathrm{w} = \boldsymbol{M}(\boldsymbol{q}, \varphi)^{-1}\left[\boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}, \varphi) - \boldsymbol{b}(\boldsymbol{q}, \mathrm{w}, \varphi) - \boldsymbol{g}(\boldsymbol{q}, \varphi)\right] \tag{3.23}$$

$$\dot{\varphi} = \dot{\varphi}_{\max} u_{\varphi}, \tag{3.24}$$

where $\boldsymbol{u} = (u_1, u_2, u_3, u_4)$ and u_φ is the normalized transformation speed—a control input specified externally. Note that the configuration dependent transformation speed derived in Chapter 2, Equation (2.2), was not used. Our experiments showed that it was sufficient to use the pure integrator reduced-order model for control-purposes, sidestepping the need for additional system identification and modeling uncertainty. The maximum transformation speed $\dot{\varphi}_{\max}$ was identified from the results in Figure 2.6.

Equations (3.21)–(3.24) are expressed in terms of the robot's inertial parameters. For the model used in the next chapter's model-predictive controller, we instantiate these dynamics with the parameter values identified in Section D. All aerodynamic effects due to approaching the ground, walls, or other obstacles, as well as the unknown effect of the interacting rotor flows, were neglected due to difficulty in producing simple analytical models of these phenomena over the desired range of operating conditions.

3.5 Quadcopter Control

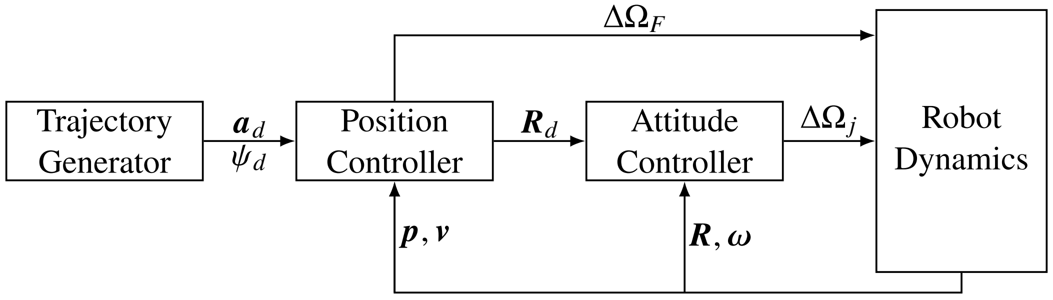


Figure 3.3: Cascaded control architecture that uses time scale separation and the decoupled dynamics of attitude and position to control a typical quadcopter aerial robot. The trajectory generator provides the desired acceleration $\boldsymbol{a}_d$ which is fed into a position controller that converts it into a desired attitude $\boldsymbol{R}_d$ and a collective thrust $\Delta\Omega_F$. The desired attitude is used in an attitude controller to produce the body torques or $\Delta\Omega_\phi, \Delta\Omega_\theta, \Delta\Omega_\psi$ which are also fed into the robot dynamics. The robot provides state feedback to the position and attitude controllers, thereby closing the loop.

The quadcopter's dynamics written in Equations (3.8)–(3.12) have a special form which is particularly amenable to a cascaded control scheme. This can be revealed by linearizing the equations of motion around the hover operating point (following the work presented in [64]) which corresponds to $\boldsymbol{v} = 0$, ϕ, θ small, $\psi = \psi_d$, $\boldsymbol{\omega} = 0$, and $\Omega_i \simeq \Omega_h$, where ψ_d is the desired yaw and Ω_h is the nominal rotational speed at hover:

$$\Omega_h = \sqrt{\frac{mg}{4k_T}}. \tag{3.25}$$

Since all the control inputs are close to this nominal value, we introduce small deviations $\Delta\Omega_F, \Delta\Omega_\phi, \Delta\Omega_\theta, \Delta\Omega_\psi$ defined by the following system:

$$\begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & 1 \\ 1 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta\Omega_F + \Omega_h \\ \Delta\Omega_\phi \\ \Delta\Omega_\theta \\ \Delta\Omega_\psi \end{bmatrix}. \tag{3.26}$$

The body torque can be written in terms of the propeller rotational speeds and approximated under the near hover assumption as

$$\boldsymbol{\tau} = \begin{bmatrix} lk_T(\Omega_3^2 - \Omega_4^2) \\ lk_T(-\Omega_1^2 + \Omega_2^2) \\ k_T k_M(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \simeq \begin{bmatrix} 4lk_T\Omega_h\Delta\Omega_\phi \\ 4lk_T\Omega_h\Delta\Omega_\theta \\ 8k_T k_M\Omega_h\Delta\Omega_\psi \end{bmatrix}, \tag{3.27}$$

by ignoring second order terms. The Euler rotational equations in component form and assuming diagonal inertia matrix,

$$\begin{cases} I_{xx}\dot{p} &= \tau_x - qr(I_{zz} - I_{yy}) \\ I_{yy}\dot{q} &= \tau_y - pr(I_{xx} - I_{zz}) \\ I_{zz}\dot{r} &= \tau_z, \end{cases} \tag{3.28}$$

can be rearranged by substituting the new control inputs, ignoring second order terms, and further assuming that the yaw rate r is small,

$$\dot{p} = \frac{4lk_T\Omega_h}{I_{xx}}\Delta\Omega_\phi \tag{3.29}$$

$$\dot{q} = \frac{4lk_T\Omega_h}{I_{yy}}\Delta\Omega_\theta \tag{3.30}$$

$$\dot{r} = \frac{8k_T k_M\Omega_h}{I_{zz}}\Delta\Omega_\psi. \tag{3.31}$$

Applying the near-hover assumption to the overall thrust allows us to rewrite it as

$$\sum_{i=1}^{4} T_i \simeq (4k_T\Omega_h^2 + 8k_T\Omega_h\Delta\Omega_F). \tag{3.32}$$

The acceleration equation (3.9) thus becomes

$$\dot{v}_1 = g(\theta \cos \psi + \phi \sin \psi) \tag{3.33}$$

$$\dot{v}_2 = g(\theta \sin \psi - \phi \cos \psi) \tag{3.34}$$

$$\dot{v}_3 = 4\sqrt{\frac{k_T g}{m}} \Delta\Omega_F, \tag{3.35}$$

where the explicit form of the rotation matrix, Equation (C.1), has been used as well as the relation $\boldsymbol{R} = [\boldsymbol{x}_B, \boldsymbol{y}_B, \boldsymbol{z}_B]$. Second order terms have again been omitted.

The body rates (3.29)–(3.31) vary linearly (locally) with the control deviations $\Delta\Omega_\phi$, $\Delta\Omega_\theta$, and $\Delta\Omega_\psi$. Since, in the near-hover state, $\dot{\phi} \simeq p, \dot{\theta} \simeq q, \dot{\psi} \simeq r$, this would suggest the use of a proportional-integral derivative controller to correct errors on the desired attitude:

$$\Delta\Omega_\phi = -k_p^\phi(\phi - \phi_d) - k_d^\phi(\dot{\phi} - \dot{\phi}_d) - k_i^\phi \int_0^t (\phi - \phi_d) d\tau \tag{3.36}$$

$$\Delta\Omega_\theta = -k_p^\theta(\theta - \theta_d) - k_d^\theta(\dot{\theta} - \dot{\theta}_d) - k_i^\theta \int_0^t (\theta - \theta_d) d\tau \tag{3.37}$$

$$\Delta\Omega_\psi = -k_p^\psi(\psi - \psi_d) - k_d^\psi(\dot{\psi} - \dot{\psi}_d) - k_i^\psi \int_0^t (\psi - \psi_d) d\tau. \tag{3.38}$$

Furthermore, when $\varphi = 0$, Equations (3.56)–(3.58) can be inverted:

$$\phi_d = \frac{1}{g}\left(\dot{v}_1^d \sin \psi_d - \dot{v}_2^d \cos \psi_d\right), \tag{3.39}$$

$$\theta_d = \frac{1}{g}\left(\dot{v}_1^d \cos \psi_d + \dot{v}_2^d \sin \psi_d\right), \tag{3.40}$$

$$\Delta\Omega_F = \frac{1}{4}\sqrt{\frac{m}{k_T g}}\, \dot{v}_3^d. \tag{3.41}$$

Thus, the desired attitude can be obtained from the desired accelerations which are outputs of the position controller. These can be obtained using a proportional-integral derivative controller on the position error $\boldsymbol{p}_d - \boldsymbol{p}$. This type of controller was proposed in [64] and is widely used for quadcopters.

3.6 Morphing Quadcopter Control

In contrast, the morphing quadcopter does not benefit from the neat decomposition between the position and attitude dynamics. To show this we take the morphing quadcopter dynamics Equations (3.42)–(3.45) and rewrite them for a particular tilt angle φ. Assuming the base frame coincides with the instantaneous center-of-gravity position for a given φ allows us to write Equations (3.42)–(3.45) as

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{3.42}$$
$$\dot{\boldsymbol{v}} = -g\boldsymbol{z}_W + \frac{T(\varphi, \boldsymbol{\Omega})}{m}\boldsymbol{z}_B + \frac{F(\varphi, \boldsymbol{\Omega})}{m}\boldsymbol{y}_B \tag{3.43}$$
$$\dot{\boldsymbol{\theta}} = \boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{\omega} \tag{3.44}$$
$$\dot{\boldsymbol{\omega}} = I(\varphi)^{-1}\left[\boldsymbol{\tau}(\varphi, \boldsymbol{\Omega}) - \boldsymbol{\omega} \times I(\varphi)\boldsymbol{\omega}\right], \tag{3.45}$$

where T is the component of total propulsive force on the body z axis $\boldsymbol{z}_B$, and F is the component of the total propulsive force on the body y axis $\boldsymbol{y}_B$. No propulsive force can be applied on the x axis due to the configuration of the robot's propellers as well as the axis of propeller tilting. A regular quadcopter only produces a propulsive force along its body z axis so $F \equiv 0$. The acceleration equation as well as the rotation equation are functions of the tilt angle φ and changing the robot tilt angle results in a change in the inertia matrix $I(\varphi)$ and the magnitude of the forces $T(\varphi), F(\varphi)$, as well as the body torque $\boldsymbol{\tau}(\varphi)$.

The body torque can be expressed in terms of the propeller rotational speeds and the tilt angle as

$$\boldsymbol{\tau}(\varphi, \boldsymbol{\Omega}) = \begin{bmatrix} k_T(b + a\cos\varphi)\left[-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2\right] \\ k_T(c\cos\varphi - k_M\sin\varphi)\left[-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2\right] \\ k_T(c\sin\varphi + k_M\cos\varphi)\left[-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right] \end{bmatrix}, \tag{3.46}$$

where a, b, c are the link lengths as defined in Figure 3.2. The magnitudes of the z and y body forces are likewise given by

$$T(\varphi, \boldsymbol{\Omega}) = k_T(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)\cos\varphi \tag{3.47}$$
$$F(\varphi, \boldsymbol{\Omega}) = k_T(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2)\sin\varphi. \tag{3.48}$$

The inertia matrix $I(\varphi)$ is a matrix function of the individual body masses and inertias whose explicit form is not needed for the present calculations. Note that the limit $\varphi \to 0$ in the equations of motion (3.42)–(3.45) results in the equations of motion of a quadcopter (Equations 3.8–3.11) since $F \to 0$ and the body torque and forces tend to equivalent forms.

Near Hover Linearization

To demonstrate the difference between the quadcopter and the morphing quadcopter we again linearize the equations of motion around the hover operating point as done in Section 3.5. Now the nominal hover thrust depends on the instantaneous tilt angle and is given by

$$\Omega_h = \sqrt{\frac{mg}{4k_T \cos\varphi}}. \tag{3.49}$$

We again introduce small deviations $\Delta\Omega_F$, $\Delta\Omega_\phi$, $\Delta\Omega_\theta$, $\Delta\Omega_\psi$ defined by the following system:[1]

$$\begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \Delta\Omega_F + \Omega_h \\ \Delta\Omega_\phi \\ \Delta\Omega_\theta \\ \Delta\Omega_\psi \end{bmatrix}. \tag{3.50}$$

Under these assumptions, the torque produced on the center of mass by the rotating propellers is given by

$$\begin{aligned} \tau_x &\simeq k_T(b + a\cos\varphi) \cdot 8\Omega_h\Delta\Omega_\phi \\ \tau_y &\simeq k_T(c\cos\varphi - k_M\sin\varphi) \cdot 8\Omega_h\Delta\Omega_\theta \\ \tau_z &\simeq k_T(c\sin\varphi + k_M\cos\varphi) \cdot 8\Omega_h\Delta\Omega_\psi, \end{aligned}$$

where all second order terms have been ignored. Substituting into the rotational dynamics, assuming diagonal inertia matrix, ignoring second order terms, and again assuming that the yaw rate r is small yields

$$\dot{p} = \frac{8k_T(c + b\cos\varphi)\Omega_h}{I_{xx}(\varphi)}\Delta\Omega_\phi, \tag{3.51}$$

$$\dot{q} = \frac{8k_T(c\cos\varphi - k_M\sin\varphi)\Omega_h}{I_{yy}(\varphi)}\Delta\Omega_\theta, \tag{3.52}$$

$$\dot{r} = \frac{8k_T(c\sin\varphi + k_M\cos\varphi)\Omega_h}{I_{zz}(\varphi)}\Delta\Omega_\psi. \tag{3.53}$$

Applying the near-hover assumption to the functions T and F allows us to rewrite them as

$$T(\varphi) \simeq (4k_T\Omega_h^2 + 8k_T\Omega_h\Delta\Omega_F)\cos\varphi \tag{3.54}$$

$$F(\varphi) \simeq (8k_T\Omega_h\Delta\Omega_\phi)\sin\varphi. \tag{3.55}$$

[1]This definition differs from the control deviations for a quadcopter in Section 3.5 because of the different frame configuration.

The acceleration equation (3.43) thus becomes

$$\dot{v}_1 = g(\theta\cos\psi + \phi\sin\psi) - 4\sqrt{\frac{k_T g}{m}}\frac{\sin\varphi}{\sqrt{\cos\varphi}}\sin\psi\Delta\Omega_\phi, \tag{3.56}$$

$$\dot{v}_2 = g(\theta\sin\psi - \phi\cos\psi) + 4\sqrt{\frac{k_T g}{m}}\frac{\sin\varphi}{\sqrt{\cos\varphi}}\cos\psi\Delta\Omega_\phi, \tag{3.57}$$

$$\dot{v}_3 = 4\sqrt{\frac{k_T g}{m}}\sqrt{\cos\varphi}\Delta\Omega_F, \tag{3.58}$$

where the explicit form of the rotation matrix in Appendix C has been used, as well as the relation $\boldsymbol{R} = [\boldsymbol{x}_B, \boldsymbol{y}_B, \boldsymbol{z}_B]$. Second order terms have again been omitted.

Near Hover Control for $\varphi \neq 0$

Two structural problems in the robot dynamics arise when $\varphi \neq 0$. Firstly, the basic proportional-integral controller, Equations (3.36)–(3.38), cannot be used to control the body rates since the coefficients of the control deviations in the body rate equations (3.51)–(3.53) are functions of the tilt angle. For example, the pitch rate $\dot{q}$, (3.52), depends on the tilt angle through the changing body inertia $I_{yy}(\varphi)$, the moment due to the vectoring of the propulsive force $c\cos\varphi$, and the vectoring of the aerodynamic drag moment $-k_M\sin\varphi$. Thus, ensuring stability would require the use of gain scheduling [65, 66], which requires significant engineering effort and cannot provide formal guarantees, despite often working extremely well.

The second issue is that the linearized accelerations (3.56)–(3.58) are dependent on $\Delta\Omega_\phi$. Hence, the attitude and the position dynamics are coupled; applying a control input $\Delta\Omega_\phi$ affects both the roll rate as well as the x and y accelerations directly. Furthermore, since the linearized accelerations now contain four unknowns, $\Delta\Omega_\phi, \Delta\omega_F, \phi, \theta$, for three equations, there is no straightforward way to invert the accelerations and build a position controller. Overall, this means that the cascaded control architecture that performs position and attitude control in separate modules, shown in Figure 3.3, can no longer be applied to symmetrically morphing robotic systems like ATMO.

This can be understood on an intuitive level by examining the forces which occur during morphing flight. In Figure 3.2, the robot is depicted in a flying and tilted configuration. If a roll torque is applied to the robot by applying more thrust on the left side (propellers 2 and 3) than the right side (propellers 1 and 4), an unwanted horizontal thrust force is also created in the opposite direction. Indeed, if the robot attempts to roll in order to translate, this results in a horizontal component of the thrust force of magnitude in the undesired direction, as obtained via the near-hover linearization in Section 3.6.

3.7 The Critical Angle

Having established the structural challenges of near-hover control for $\varphi \neq 0$, we now turn to another feasibility limitation imposed by aerial morphing; tilting reduces the vertical component of thrust. Thus, beyond a certain angle, even full throttle cannot sustain weight. We formalize this boundary with the *critical angle*:

$$\boxed{\cos \varphi_c = \frac{mg}{4k_T \,\Omega_{\max}^2}.} \tag{3.59}$$

This expression follows from the standard static thrust model $T_i = k_T \Omega_i^2$ and the assumption that each rotor's vertical contribution decays as $\cos \varphi$ under symmetric tilt. At $\Omega_{\max}$ for all four rotors, the total vertical thrust is $4k_T \Omega_{\max}^2 \cos \varphi$; equating this to mg yields (3.59). For any $\varphi > \varphi_c$, the hover equilibrium is inadmissible, irrespective of control law, since the available vertical thrust is insufficient to balance gravity. For ATMO, with its thrust-to-weight ratio ≈ 2.1, (3.59) gives $\varphi_c \approx 60°$.

As the tilt angle approaches the critical value, objectives that are sensible in hover—such as tight altitude and attitude regulation—become physically unattainable. A controller that continues to demand hover will drive the actuators to saturation, exhausting control authority for disturbance rejection. Practically, φ_c delineates the boundary where hover operation is possible, thus serving as an important planning constraint. For tilts below φ_c a zero-impact touchdown is, in principle, achievable by appropriate control, whereas beyond φ_c it becomes infeasible.

3.8 Summary

We reviewed the quadcopter equations of motion and showed—via a near-hover linearization and analysis of the linearized dynamics—why they are well suited to cascaded control. We then derived a morphing-quadcopter model inspired by ATMO, providing two complementary formulations. The first captures the shifting center of gravity and inertia using floating-base multi-body dynamics; this form underpins the model predictive controller developed in the next chapter. The second, valid for a fixed tilt angle, is used to expose intrinsic differences from a standard quadcopter. We show that, unlike the fixed-frame quadcopter, the morphing quadcopter admits shape-dependent attitude dynamics and a coupling between the position and attitude dynamics through the roll control deviation. These properties indicate that a classical cascaded structure is not appropriate for control of ATMO. Finally, we introduced the critical angle—the tilt at which thrusters saturate when hovering. In the next chapter, we develop a model predictive controller that addresses the control challenges presented in this chapter and accounts for the full complexity of the nonlinear morphing quadcopter dynamics.

Chapter 4

BI-MODAL GROUND AERIAL LOCOMOTION CONTROL

This chapter incorporates material from the following publication:

Ioannis Mandralis et al. "ATMO: an aerially transforming morphobot for dynamic ground-aerial transition". In: *Communications Engineering* 4.1 (Apr. 2025). ISSN: 2731-3395. DOI: `10.1038/s44172-025-00413-6`. URL: `http://dx.doi.org/10.1038/s44172-025-00413-6`. (published).

In this chapter, a controller for the stabilization of the Aerially Transforming Morphobot across the different phases of bi-modal ground-aerial locomotion is presented. We first review the field of optimization-based control which enables the control of nonlinear systems while systematically taking into account input saturation and state constraints. The controller objectives are layed out and the proposed solution is described. The controller design relies on a unifying flight-transition module that enables quadcopter flight, morphing flight, as well as smooth ground-aerial transitions using a single model predictive control scheme. The implementation of the controller on the embedded hardware onboard the Aerially Transforming Morphobot is discussed. Finally, the controller is validated by experiments in Caltech's CAST flight arena.

4.1 Optimization-Based Control

In recent years, the improvements in the speed of central processing units (CPUs) have made it possible to solve complex optimization problems fast enough for real-time control. Rather than using optimization as a way to design control algorithms offline, optimization algorithms are increasingly being used online to adapt to the underlying changes in the dynamics of robotic systems. This has made possible in particular a branch of control called *model-predictive control* or MPC.

Unlike many classical control methods, model predictive control explicitly incorporates the system model, performance objectives, and physical limitations into the design process, significantly expanding the operational envelope of the controller. This makes it particularly well suited for morphing robotic flight where safety and maneuverability must be balanced in real time. Below we briefly review the main goals of model-predictive control in the context of the control of a robotic system, following the presentation in [67].

At its core, model predictive control relies on the solution of an *optimal control* problem. Optimal control is a systematic framework for determining the control inputs that minimize a specified performance criterion while respecting the dynamics and constraints of a dynamical system. The study of optimal control theory, deeply rooted in the calculus of variations, is a subject with rich history whose proper treatment is out of the scope of the present thesis. For a full treatment, the reader is referred to [68, 69].

Optimal Control

Optimal control consists of minimizing a cost functional that assigns a value, or goodness, to state and input trajectories while ensuring that the state and input trajectories are *dynamically feasible* i.e. they respect the system dynamics. More specifically, a minimization over state and input trajectories—the functions $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$—is performed:

$$\begin{aligned} &\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \int_0^T L(\boldsymbol{x}(t), \boldsymbol{u}(t))dt + V(\boldsymbol{x}(T)), \\ &\text{subject to: } \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). \end{aligned} \tag{4.1}$$

Constraints on the states and inputs along the trajectory and at the beginning and end of the trajectory can also be specified. Optimal control is best understood as a trajectory generation method rather than a direct control law. By solving an optimization problem over the system dynamics and constraints, it yields an open-loop sequence of inputs and states that minimize a prescribed cost function.

However, such open-loop solutions are inherently sensitive to disturbances, time delays, and modeling mismatch, and thus cannot serve as robust controllers on their own. Only when optimal control is embedded in a *receding horizon framework*, where the problem is resolved at each time step based on updated state information, does it provide the feedback necessary for closed-loop stability and robustness. This formulation, widely known as Model Predictive Control (MPC), bridges trajectory optimization with real-time control.

Receding Horizon Control

The key idea behind receding horizon control is to solve an optimal control problem, apply the optimal control input for a short time interval δt, solve the optimization problem again based on the most recent state update, and reapply the optimal control input for the fixed time interval. More formally, this procedure can be written at each given time t_i as

$$\boldsymbol{u}_{[t_i, t_i+\delta T]} = \arg\min_{(\boldsymbol{x}, \boldsymbol{u})} \int_{t_i}^{t_i+T} L(\boldsymbol{x}, \boldsymbol{u})\, d\tau + V(\boldsymbol{x}(t_i + T)) \tag{4.2}$$

subject to

$$\boldsymbol{x}(t_i) = \text{current state}, \tag{4.3}$$

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}), \tag{4.4}$$

$$g_j(\boldsymbol{x}, \boldsymbol{u}) \leq 0, \quad j = 1, \ldots, r, \tag{4.5}$$

$$\psi_k(\boldsymbol{x}(t_i + T)) = 0, \quad k = 1, \ldots, q. \tag{4.6}$$

The functions $g_j(\cdot)$ and $\psi_k(\cdot)$ are the path and terminal constraints, respectively. By executing this loop fast enough feedback can be achieved. Stability can be guaranteed under certain conditions such as the choice of cost function, time horizon, and update frequency [67], making model predictive control a particularly attractive controller for robotic systems.

The optimal control problem (4.2)–(4.5) has a closed-form, analytical solution in few cases of interest. An important example is the case when the stage cost L is quadratic, the dynamics are linear, $T = \infty$, and there are no state or input constraints. This case, known as the *linear quadratic regulator*, admits a closed form, constant gain control policy by application of Pontryagin's Maximum Principle [67, 68]. However, in most applications of interest, including for the Aerially Transforming Morphobot, state and input constraints are of key importance, and dynamics are often nonlinear. In Section 4.6, numerical approaches to the solution of the optimal control problems that do not admit closed form, analytical solutions are discussed.

4.2 Controller Description

In this section, a model predictive controller, designed to control the Aerially Transforming Morphobot across all phases of bi-modal ground-aerial locomotion, is described. The controller objectives are:

1. **Objective 1: Quadcopter flight.** When $\varphi = 0$, ATMO should fly like a regular quadcopter. As described in Chapter 3, Section 3.5, this is usually achieved using a cascaded control architecture, typically implemented in out-of-the-box flight controllers like the PX4 Autopilot framework [53]. However,

these flight controllers are unable to handle the morphing flight or morpho-transition tasks. To avoid switching controllers mid-flight, which increases control complexity and failure points, one unified controller should handle quadcopter flight as well as the other flight modes.

2. **Objective 2: Morphing flight.** For tilt angles $0 < \varphi < \varphi_c$, ATMO should be able to hover, track desired trajectories, and effectively reject disturbances. This is desirable for flying through narrow gaps, changing the body inertia to improve flight performance, and for recovering from actuator failures by reconfiguring mid-flight. As elaborated in Chapter 3, thruster tilting causes a dynamic coupling between roll and horizontal translational motion. Furthermore, as the tilt angle approaches the critical angle, as defined in Equation (3.59), there is less control authority leftover from hover control to achieve other control objectives. The controller should explicitly account for the coupled and critical dynamics as the robot changes shape.

3. **Objective 3: Morpho-transition.** When transitioning between flight and ground phases dynamically, aerial transformation is required. As it is desirable to land on the robot's wheels with the largest possible tilt angle, this may require approaching or surpassing the critical angle, $\varphi \geq \varphi_c$. This can be dangerous due to the loss of position and velocity control when attempting to compensate for gravity. Near the ground, the system dynamics become even more complex due to ground-proximity aerodynamics and rotor wake interactions, as shown in Section 2.6. The controller should be able to achieve safe transitions even as the physical demands on the system increase.

4. **Objective 4: Ground locomotion.** Once on the ground, the controller should be able to navigate freely using differential drive steering. To achieve the fastest transition possible, the controller should detect when the robot is on the ground and switch to a driving controller.

The control architecture designed to achieve Objectives 1–4 is summarized in Figure 4.1. The control system is based on a central module which receives a state estimate $\hat{\boldsymbol{x}}$, the current tilt angle $\hat{\varphi}$, a reference state and input $\boldsymbol{x}_{\text{ref}}, \boldsymbol{u}_{\text{ref}}$ and generates controls $\boldsymbol{u}_a$ and $\boldsymbol{u}_g$. The state estimate is computed from a position and orientation measurement, $\boldsymbol{p}, \boldsymbol{\theta}$, received from either a motion capture system, or another localization system such as visual-inertial odometry [70] or Global Navigation Satellite System (GNSS). The position and attitude measurements are fed into an Extended Kalman

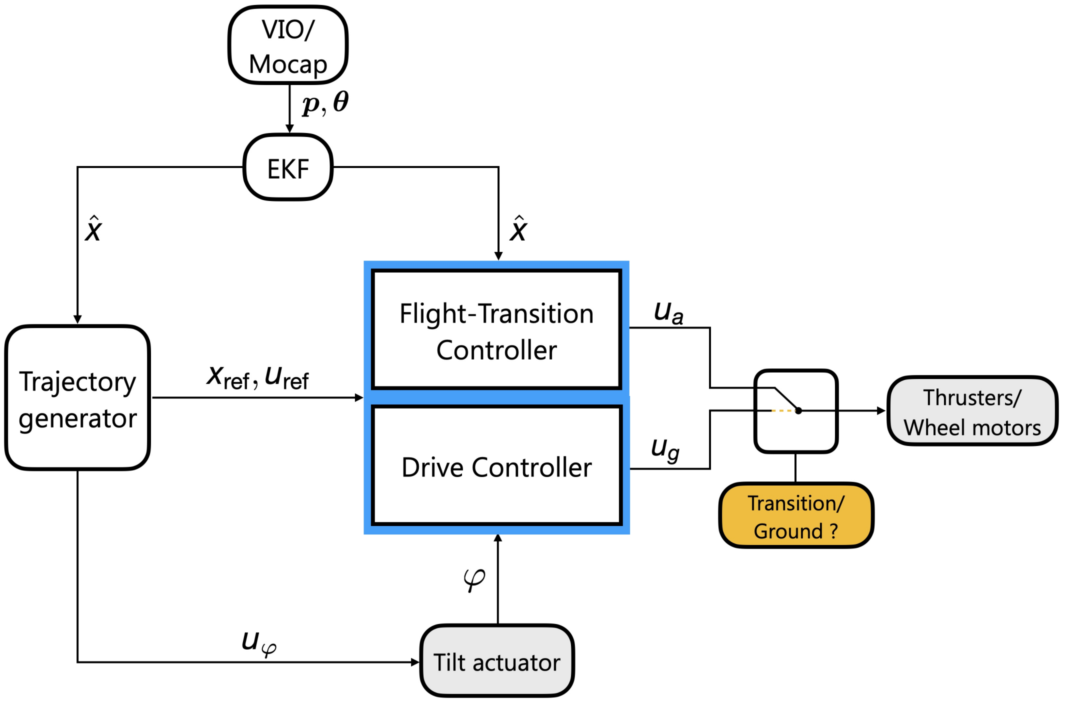


Figure 4.1: Bi-modal ground aerial locomotion control architecture: control diagram with switching logic and signal flow between the central modules.

Filter (EKF) that runs onboard the PX4-based flight controller and updates the state estimate by fusing high-rate inertial data from the inertial measurement units (IMU) with other lower rate sensors [53].

The controls $\boldsymbol{u}_a$ are generated by the flight-transition controller and are the normalized desired propeller rotational speeds, as explained in Section 3.2. The remaining controls $\boldsymbol{u}_g$ are generated by the drive controller and correspond to the normalized rotational rates of the left and right wheel-spinning motors. Whether the commands are actually sent to the thruster motor electronic speed controllers or wheel motor drivers is determined by whether or not a ground contact or transition state were estimated.

This switching logic is implemented with a state machine that uses the current system state to determine whether the robot is grounded or not, and cuts or allows the signal flow to the wheel motors accordingly. To enable smooth driving as soon as the robot touches the ground, the wheel motors are activated shortly before impact and the drive controller begins sending wheel spin commands in order to track the reference position. Similarly, once the robot touches the ground, the thrust commands no longer reach the thruster motors, ensuring safety.

The robot tilt angle is controlled by a normalized tilt velocity signal u_φ that is supplied as a throttle command from the trajectory generator module to the Roboclaw microcontroller which then actuates the tilt actuator using a pulse-width modulation signal. The tilt mechanism in turn produces angular feedback, φ, to the controller using the mechanism kinematics and the DC motor encoder count (Equation 2.1).

4.3 Flight-Transition Controller

The central controller responsible for Objectives 1–3 is the flight-transition controller. This controller applies the receding horizon control framework to track trajectories in position, velocity, and tilt angle space during aerial transformation. The usual choice of cost function for trajectory tracking, for example when designing a nonlinear MPC controller for a quadcopter [71, 72, 73], is

$$L(\boldsymbol{x}, \boldsymbol{u}) = (z - z_{\text{ref}})^\top \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R} \end{bmatrix} (z - z_{\text{ref}}). \tag{4.7}$$

where $z = (\boldsymbol{x}^\top, \boldsymbol{u}^\top)^\top$ and $\boldsymbol{Q} \succeq 0, \boldsymbol{R} \succ 0$, thereby penalizing deviations from the desired reference trajectory.

However, a fixed quadratic cost is often insufficient to reliably govern tracking across all operating regimes of the aerial robot, and advanced tracking controllers may need online variation of the cost function to achieve maximum performance. For example, to achieve tracking control for a quadcopter with a failed actuator, the cost on yaw tracking is set to zero to account for the loss of yaw controllability [74]. Likewise, the actor-critic model predictive control framework supplements an actor network with a differentiable MPC head and uses deep reinforcement learning to vary the cost function online, enabling mode switching behavior when passing through gates in autonomous drone racing [75].

Similar to the examples mentioned above, our experimentation revealed that a fixed quadratic cost function did not produce well-tuned tracking performance across tilt angles $0 \leq \varphi \leq \varphi_c$ for the Aerially Transforming Morphobot. Three key observations were noted:

1. **Observation 1:** As the tilt angle increased and approached φ_c the attitude tracking performance deteriorated. It was difficult to achieve good attitude tracking performance for $\varphi = 0$ and other angles between 0 and $\approx 0.8\varphi_c$, simultaneously.

2. **Observation 2:** When the tilt angle approached or surpassed the critical angle, the optimization problem became numerically ill-conditioned. This resulted in irrecoverable failure of the quadratic program solver `HPIPM` [76] (see Section 4.6 for more details).

3. **Observation 3:** When on the ground, requiring position or attitude tracking led to controller instability. Indeed morphing quadcopter dynamics are no longer a suitable representation of the robot dynamics when in contact with the ground.

Motivated by the above observations, we developed a cost function which changes online to adjust the priorities on position, velocity, attitude, and angular rate tracking as the tilt angle, and distance from the ground, vary. We found that, as the robot was transitioning between flight and ground locomotion, it was necessary to reduce the cost on position and attitude tracking to zero to ensure numerical stability of the model predictive controller.

This was achieved by a convex combination of two cost functions, each tuned for adequate tracking performance in different flight phases, weighted by the factor $\alpha(z,\varphi)$ which depends on the current altitude z and the tilt angle:

$$L(\boldsymbol{x},\boldsymbol{u}) = \alpha(z,\varphi)\underbrace{L_1(\boldsymbol{x},\boldsymbol{u})}_{\text{flying}} + (1-\alpha(z,\varphi))\underbrace{L_2(\boldsymbol{x},\boldsymbol{u})}_{\text{transitioning}}. \tag{4.8}$$

Cost function (4.8) is composed of two terms, L_1, L_2, which are the tracking costs for quadcopter flight and morpho-transition, respectively:

$$L_i(\boldsymbol{x},\boldsymbol{u}) = (z - z_{\text{ref}})^\top \begin{bmatrix} \boldsymbol{Q}_i & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_i \end{bmatrix} (z - z_{\text{ref}}), \tag{4.9}$$

for $i = \{1, 2\}$. The quadratic form matrices are diagonal and chosen as

$$\boldsymbol{Q}_1 = \text{diag}(1,1,1,10,10,18,0.1,0.1,0.8,1.5,2.7,2.0), \tag{4.10}$$

$$\boldsymbol{Q}_2 = \text{diag}(0,0,0,0,0,0,0,0,0,1.5,2.7,2.0), \tag{4.11}$$

$$\boldsymbol{R}_1 = \boldsymbol{R}_2 = \text{diag}(.1,.1,.1,.1). \tag{4.12}$$

The matrix $\boldsymbol{Q}_1$ was tuned on the hardware to ensure tight tracking for quadcopter flight phase when $\varphi = 0$. The matrix, $\boldsymbol{Q}_2$ weights the body rates p, q, r the same as for $\boldsymbol{Q}_1$ but sets the cost to all the remaining states to zero. This reflects the loss of controllability in $x, y, z, \dot{x}, \dot{y}, \dot{z}$ as the tilt angle approaches the critical configuration.

The cost is varied according to the tilt angle and the altitude with the blending factor $\alpha(z, \varphi)$,

$$\alpha(z, \varphi) = f(z) \cos \varphi, \tag{4.13}$$

$$f(z) = \begin{cases} 1.0 & \text{if } z \geq z^*, \\ \frac{z - z_g}{z^* - z_g} & \text{if } z_g \leq z \leq z^*, \\ 0.0 & \text{otherwise,} \end{cases} \tag{4.14}$$

where z_g is the ground height and z^* is a pre-set transition height. The blending factor is proportional to two terms. The first term $\cos \varphi$ decays to zero as $\varphi \to 90°$, and the second term $f(z)$, active only in the transition region, ensures that $\alpha \to 0$ as the robot approaches the ground. The transition was set to begin at $z^* = 0.45$ m above the ground. As seen in Section 2.6, this is the height at which the ground effect begins to impact the overall thrust level, increasing the thrust at the same propeller rotational rate. By setting the transition region to this height, greater tilt angles at landing can be achieved by taking advantage of the positive effect of the ground effect up to $\varphi^* = 70°$.

Equipped with cost function (4.8), the optimal control problem solved by the flight transition controller is

$$\begin{aligned} \min_{\boldsymbol{x}, \boldsymbol{u}_a} \quad & \int_0^T [\alpha(z, \varphi)\, L_1(\boldsymbol{x}, \boldsymbol{u}_a) + (1 - \alpha(z, \varphi)) L_2(\boldsymbol{x}, \boldsymbol{u}_a)]\, \mathrm{d}t \\ \text{s.t.} \quad & \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_a, \varphi) \quad \forall\, t \in [0, T], \\ & 0 \leq u_a^i \leq 1 \quad \forall\, i \in \{1, \dots, 4\}. \end{aligned} \tag{4.15}$$

Here, T is the horizon of the optimization, and u_a^i are the normalized thrust values of each thruster, constrained between 0 and 1 to account for actuator saturation. For the system dynamics constraint, $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_a, \varphi)$, the full-order rigid body dynamics of the morphing quadcopter, described by Equations (3.21)–(3.24), were used. The state-varying loss function reflects the loss of control authority which occurs at high tilt angles and ensures numerical stability of the optimization problem across all flight phases. The numerical implementation of the optimal control problem in Equation 4.15 is discussed in Section 4.6.

4.4 Drive Controller

When on the ground, the driving controller receives a desired linear and angular velocity (v_g, ω_g), and computes the wheel rotation rates of the left and right wheels

$\boldsymbol{u}_g = (\omega_L, \omega_R)$, using a simple model of a differential drive vehicle. The dynamics of the ground position and orientation (x, y, ψ) due to the wheel spinning are given by

$$\dot{x} = \frac{1}{2}R_w(\omega_L + \omega_R)\cos\psi \equiv v_g \cos\psi, \tag{4.16}$$

$$\dot{y} = \frac{1}{2}R_w(\omega_L + \omega_R)\sin\psi \equiv v_g \sin\psi, \tag{4.17}$$

$$\dot{\psi} = \frac{R_w}{2l_b}(\omega_L - \omega_R) \equiv \omega_g, \tag{4.18}$$

with $v_g = \frac{1}{2}R_w(\omega_L + \omega_R)$ and $\omega_g = \frac{R_w}{2l_b}(\omega_L - \omega_R)$. R_w and l_b denote the wheel radius and half wheel base respectively. Given a desired ground twist (v_g, ω_g) we calculate the associated wheel speeds as $\omega_R = \frac{1}{R_w}(v_g - l_b\omega_g)$ and $\omega_L = \frac{1}{R_w}(v_g + l_b\omega_g)$. These are normalized by the maximum wheel rotation rate, and the normalized signal is supplied to the wheel drive motors by sending a throttle signal to the Roboclaw micro-controller, which converts this to a pulse-width modulation signal that drives the motors.

4.5 Trajectory Generation

The trajectory generator outputs the desired state and input trajectory and sends the reference points in real time to the robot through the ROS2 network. For the experiments described in Section 4.7, control commands were supplied for the 3D position of the robot as, $\boldsymbol{p}^{\text{ref}}(t) = \boldsymbol{p}(t) + \boldsymbol{v}T_s$, where T_s is the sampling time of the controller. The velocity commands $\boldsymbol{v} = (v_x, v_y, v_z)$ are received from a radio controller and limited to 1 m/s for safety. The attitude and angular velocity reference states are set to zero throughout the experiment.

The tilt angle velocity reference is assigned according to the phases of the dynamic wheel landing task. The important parameters are the height above the ground, z, and whether the robot is grounded or not (grounded: $\lambda = 1$, not grounded: $\lambda = 0$). Additionally, if the robot is on the ground the user can indicate that the robot should take off by setting the overall throttle c to be greater than 50% throttle. This logic is encoded overall as the following tilt velocity reference:

$$\dot{\varphi}_{\text{ref}}(z, c, \lambda) = \begin{cases} \dot{\varphi}_{\max} & z < z_{\varphi},\ \lambda = 0, \\ -\dot{\varphi}_{\max} & c \geq 0.5,\ \lambda = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4.19}$$

The in-flight body posture is limited to $\varphi_{\max} = 50^\circ$, when not in transition close to the ground, ensuring that at least 35% of the total thrust is always available for disturbance rejection. This value is chosen to ensure that 35% of thrust is available for disturbance rejection i.e. the thrust at the maximum tilt angle should be 1.35 times greater than the weight of the robot:

$$\frac{T_{\max} \cos \varphi_{\max}}{mg} = 1.35. \tag{4.20}$$

For a thrust to weight ratio of 2.1, $T_{\max}/mg = 2.1$ then $\varphi_{\max} = 50°$. The tilt angle is controlled by receiving the normalized tilt angle velocity command u_φ directly from the trajectory generator. The tilt angle φ is sensed using the mechanism kinematics and sent back to the controller. It is also used to limit the tilt angle to $\varphi \leq \varphi_{\max} = 50°$ in flight and $\varphi \leq \varphi_{\max} = 70°$ in the transition phase.

4.6 Controller Implementation

The control architecture summarized in Figure 4.1 is implemented using a set of `ROS2` nodes that run the different control modules at a fixed rate. The full code-base developed is available here. The most demanding part of the controller is the flight-transition controller that must solve an optimal control problem at a sufficiently high-rate to ensure stability, requiring a high-performance implementation.

Multiple Shooting Discretization

To resolve the flight transition controller online, we chose to employ the multiple shooting method [77] due to its high-performance implementation in the software package Acados [78]. Acados, optimized for embedded devices like the NVIDIA Jetson onboard ATMO, is also compatible with symbolic modeling languages like `CasADi` [79], enabling the dynamics constraint, derived analytically and from first principles in Equations (3.21)–(3.24) to be used directly. Unlike most modern multi-body libraries (e.g., Pinocchio, RBDL, Drake, MuJoCo, Isaac Lab), which evaluate the dynamics numerically at runtime, the symbolic formulation facilitates tight integration with optimal control solvers by allowing pre-computation of derivatives, sparsity patterns, and efficient code generation.

In multiple shooting, the infinite dimensional optimal control problem of Equation (4.15) is reduced to a finite dimensional optimization problem by first discretizing the time interval $[t_0, t_N]$ into time points $t_0 ... t_N$, where $T = t_N - t_0$ is the optimization time horizon. In each sub-interval the control vectors are approximated by a finite set of parameters using basis functions, e.g. piecewise polynomials. In the implementation of the flight-transition controller, the control trajectory is approximated by a piecewise constant parameterization.

At each time point a discrete state and control variable is considered: $\boldsymbol{x}_0, ..., \boldsymbol{x}_N$ and $\boldsymbol{u}_0, ..., \boldsymbol{u}_N$ and the system dynamics are enforced by propagating the dynamics forward on each sub-interval $[t_k, t_{k+1})$ using a fourth-order Runge-Kutta numerical integration scheme: $\boldsymbol{x}_{k+1} = \boldsymbol{\phi}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)$. This allows the optimal control problem in Equation (4.15) to be transformed into a nonlinear program (NLP):

$$\min_{\substack{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_N, \\ \boldsymbol{u}_0,\ldots,\boldsymbol{u}_{N-1}}} \sum_{k=0}^{N-1} l(\boldsymbol{x}_k, \boldsymbol{u}_k) + V(\boldsymbol{x}_N) \tag{4.21}$$

subject to

$$\boldsymbol{x}_0 = \text{current state} \tag{4.22}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{\phi}_k(\boldsymbol{x}_k, \boldsymbol{u}_k), \quad k = 0, \ldots, N-1, \tag{4.23}$$

$$0 \leq \boldsymbol{u}_k \leq 1, \quad k = 0, \ldots, N-1, \tag{4.24}$$

where $l(\boldsymbol{x}_k, \boldsymbol{u}_k) = (t_{k+1} - t_k)L(\boldsymbol{x}_k, \boldsymbol{u}_k)$. This scheme is equivalent to performing the single-shooting method on each sub-interval in parallel [80], but studies show that it leads to significantly improved convergence behavior [81]. The conditions (4.23) are called the matching conditions and ensure that the solution is continuous and respects the dynamics at the grid points.

Sequential Quadratic Programming

To solve the nonlinear program in equations (4.21)–(4.24), Sequential Quadratic Programming (SQP)—a widely used numerical method for solving nonlinear optimization problems—is used. The central idea of SQP is to iteratively approximate the nonlinear problem with a sequence of quadratic programs (QPs), each constructed by locally linearizing the constraints and forming a quadratic approximation of the cost function around the current iterate. Thus Equations (4.21)–(4.24) are linearized around the current iterate of the SQP $\{\{\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k\}_{k=0}^{N-1}, \bar{\boldsymbol{x}}_N\}$, and the nonlinear program is converted into the following quadratic program:

$$\min_{\substack{\boldsymbol{\delta x}_0,\ldots,\boldsymbol{\delta x}_N, \\ \boldsymbol{\delta u}_0,\ldots,\boldsymbol{\delta u}_{N-1}}} \sum_{k=0}^{N-1} \left(\frac{1}{2}\boldsymbol{\delta z}_k^\top \boldsymbol{H}_k \boldsymbol{\delta z} + \boldsymbol{g}_k^\top \boldsymbol{\delta z}\right) + \frac{1}{2}\boldsymbol{\delta x}_N^\top \boldsymbol{H}_N \boldsymbol{\delta x}_N + \boldsymbol{g}_N^\top \boldsymbol{\delta x}_N \tag{4.25}$$

subject to

$$\boldsymbol{\delta x}_{k+1} = \boldsymbol{A}_k \boldsymbol{\delta x}_k + \boldsymbol{B}_k \boldsymbol{\delta u}_k + \boldsymbol{\phi}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) - \bar{\boldsymbol{x}}_k, \quad k = 0, \ldots, N-1. \tag{4.26}$$

The Hessians and gradients resulting from the linearization about the previous SQP iterate are given by

$$\boldsymbol{H}_k = \nabla^2_{\boldsymbol{z}} l(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k), \tag{4.27}$$

$$\boldsymbol{H}_N = \nabla^2_{\boldsymbol{x}} V(\bar{\boldsymbol{x}}_N), \tag{4.28}$$

$$\boldsymbol{g}_k = \nabla_{\boldsymbol{z}} l(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k), \tag{4.29}$$

$$\boldsymbol{g}_N = \nabla_{\boldsymbol{x}} V(\bar{\boldsymbol{x}}_N), \tag{4.30}$$

and the linearized dynamics matrices are given by

$$\boldsymbol{A}_k = \frac{\partial \boldsymbol{\phi}_k}{\partial \boldsymbol{x}_k} \tag{4.31}$$

$$\boldsymbol{B}_k = \frac{\partial \boldsymbol{\phi}_k}{\partial \boldsymbol{u}_k}. \tag{4.32}$$

In the above, the shorthand $\boldsymbol{z} = [\boldsymbol{x}, \boldsymbol{u}]^\top$ and $\boldsymbol{\delta z} = [\boldsymbol{\delta x}, \boldsymbol{\delta u}]^\top$ has been employed. Once the quadratic program (QP) (4.25)–(4.26) has been solved, the iterate $\{\boldsymbol{\delta z}_i\}_{i=1}^N$ is used to update the optimal trajectory $\{\{\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k\}_{k=0}^{N-1}, \bar{\boldsymbol{x}}_N\}$. This procedure is implemented using a numerical implementation of the `HPIPM` algorithm [76] with partial condensing [82].

Real-Time Iteration

To ensure that the SQP can be solved fast enough for real-time control, the real-time iteration (RTI) approach is taken. Unlike the full SQP scheme, which requires multiple QP solves and nonlinear evaluations per sampling instant until convergence, SQP-RTI performs only a single QP solve per iteration. Although this introduces an approximation relative to the fully converged solution, it significantly reduces the computational requirements. This property renders SQP-RTI particularly suitable for real-time model predictive control for the ATMO platform, where stringent timing constraints preclude the use of fully converged nonlinear optimization at every sampling step, yet an accurate treatment of nonlinear dynamics and constraints remains essential.

Controller Parameters

Using the above numerical implementation we are able to run the model predictive controller at a frequency of 150Hz on the onboard computer. To achieve this multiple techniques are combined. First, the cost function is updated only once every 10 iterations of the model predictive control scheme to ensure minimal computational overhead, as well as giving sufficient time for the optimizer to warm up the solution.

Second, the horizon time is chosen as $T = 2.0$ seconds and $N = 20$ collocation points—striking a balance between controller accuracy and computational overhead. Finally, the RTI scheme is found to significantly improve optimization time.

4.7 Experimental Validation

The bi-modal locomotion controller was implemented on the Aerially Transforming Morphobot and its performance was evaluated on various tasks. First, a dynamic wheel landing maneuver, or morpho-transition maneuver, was performed in the Caltech CAST flight arena. In this experiment the controller was used to track a reference trajectory in space that consisted of a descent with some forward motion while tilting the wheel-thrusters, landing on the wheels, and finally driving forward, as described in Section 4.2. For state estimation, the position and attitude measurements were supplied using a motion capture system.

Snapshots from the resulting experiment are shown in Figure 4.2 and a video of the maneuver can be found in the supplementary materials (Video 2). As can be seen labeled in Figure 4.2 on the $\varphi(t)$ graph, the final tilt angle at landing is $\varphi_g = 65°$, showing that the robot is successfully able to land at a tilt angle that is past the critical angle. This is achieved due to the change of cost function in the transition phase evidenced by the sharp drop of $\alpha(t)$.

By virtue of the changing cost function the robot can continue tilting its wheel-thrusters while still maintaining the desired attitude by reducing the emphasis on holding vertical position or velocity and prioritizing only attitude stabilization. Ultimately, the tilt angle at landing is $\Delta\varphi = 15°$ above the maximum in-flight tilt angle, while the maximum mean normalized thrust during the maneuver is $\bar{u} = 0.7$. We hypothesize that this is possible in part due to the ground effect that was measured in Figure 3(C).

Note that, the robot shows a noticeable lateral drift during the maneuver (Video 2). This might have been due to wind disturbances present in the CAST flight arena or controller tuning. We repeated the experiment with improved tuning of the cost function parameters and report noticeably less lateral drifting, as can be seen in Video 4 of the supplementary materials. The safety tether was also removed, showing that additional forces were not applied to the robot during the maneuver.

We also validated our control method by performing a driving takeoff followed by a dynamic wheel landing. In this maneuver the robot was tasked with taking off while driving, dynamically transitioning to flight by deploying as a quadrotor, and finally

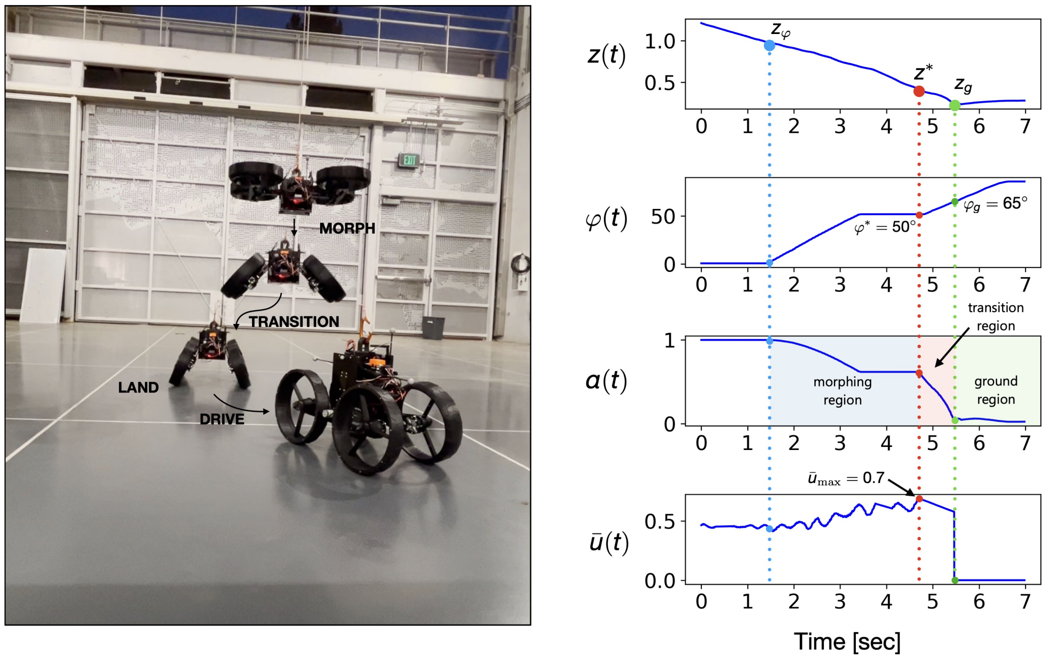
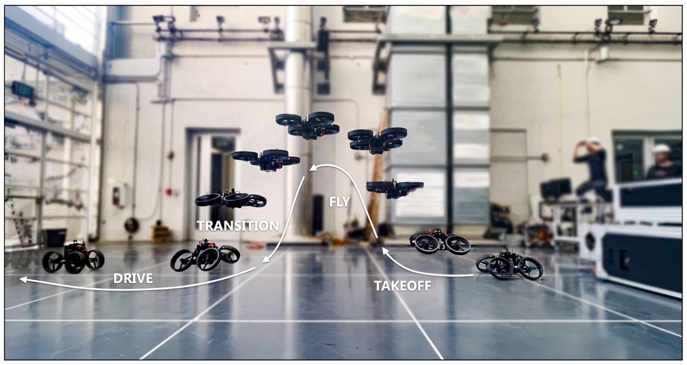


Figure 4.2: Experimental validation of dynamic wheel landing, or morpho-transition maneuver, in the CAST flight arena using bi-modal ground aerial locomotion controller. (left) Snapshots from the performed trajectory. (right) Recorded data during an autonomous wheel landing. In blue $z(t), \varphi(t), \alpha(t), \bar{u}(t)$, i.e. the altitude, tilt angle, control blending factor, and the mean normalized thrust applied by the four rotors are plotted. The three vertical lines denote the time instances at which the robot begins to morph (height z_φ), the point at which transition begins (height z^*) and finally the point at which impact occurs (height z_g). The morphing, transition, and ground regions are highlighted in blue, red, and green respectively. The morphing height, transition height, and ground contact height (z_φ, z^*, z_g) are indicated along with corresponding angles, and the maximum normalized thrust is labeled.

morphing again to land on wheels and continue driving. The resulting maneuver and the evolution of key states are depicted in Figure 4.3. A video of the maneuver can be found in Supplementary Materials, Video 3.

The robot started in drive mode with $\varphi_d = 56°$ tilt angle in driving configuration. As the robot drove forward, the throttle was increased, triggering the robot to decrease the tilt angle. At $\varphi_0 = 50°$, the thrusters were activated and the throttle increased, thus propelling the robot into the air. After some moments in the air, the robot descended and smoothly transitioned into driving configuration. Once in the transition phase $z < z^*$, the wheels were activated and the robot smoothly continued its forward driving motion. As can be seen in Figure 4.3, the maximum



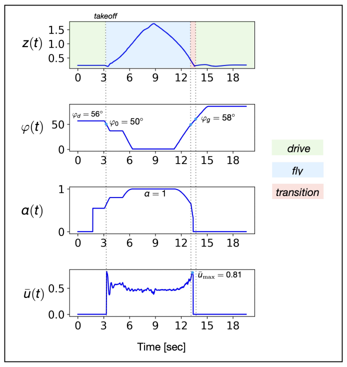


Figure 4.3: Experimental validation of ground aerial transition maneuvers: flight-initiating driving takeoff. (left) Robot moving from right to left begins in driving configuration, drives and jumps up by switching on its thrusters. It transforms to quadrotor configuration until it reaches the apex of the trajectory, and finally it begins its descent with forward velocity, transforming into drive mode, smoothly landing, and continuing its driving motion (Video 3 in Supplementary Materials). (right) In blue $z(t), \varphi(t), \alpha(t), \bar{u}(t)$, i.e. the altitude, tilt angle, time varying mixing controller coefficient, and the average normalized thrust. The driving, flying, and transitioning modes are highlighted in green, blue, and red respectively.

tilt angle while in flight was 50°, and the final tilt angle at landing was $\varphi_g = 58°$. The maximum mean normalized thrust in the landing phase of the maneuver was $\bar{u} = 0.81$. This is significantly higher than in the previous maneuver potentially due to the more aggressive forward motion, requiring higher propeller rotational rates to produce a pitch moment and counteract gravity simultaneously. Additional experiments verified that landings at forward velocity with uninterrupted driving are repeatable; see Figure 4.4 and Video 6.

Finally, we demonstrate a practical benefit of mid-air transformation: a dynamic wheel-first landing on an inclined plane. In the maneuver of Fig. 4.4 (see Video 5), ATMO transitions in flight, aligns its wheels to a slope of known pose, touches down smoothly, and continues driving. By establishing wheel contact before touchdown, the approach mitigates the tip-over risk that conventional quadrotors face when landing on sloped terrain.

4.8 Summary

To capture all phases of the maneuver, we develop a model-predictive controller (MPC) that stabilizes the Aerially Transforming Morphobot (ATMO) during mid-air morphing and in ground proximity, with a contact-triggered switching logic

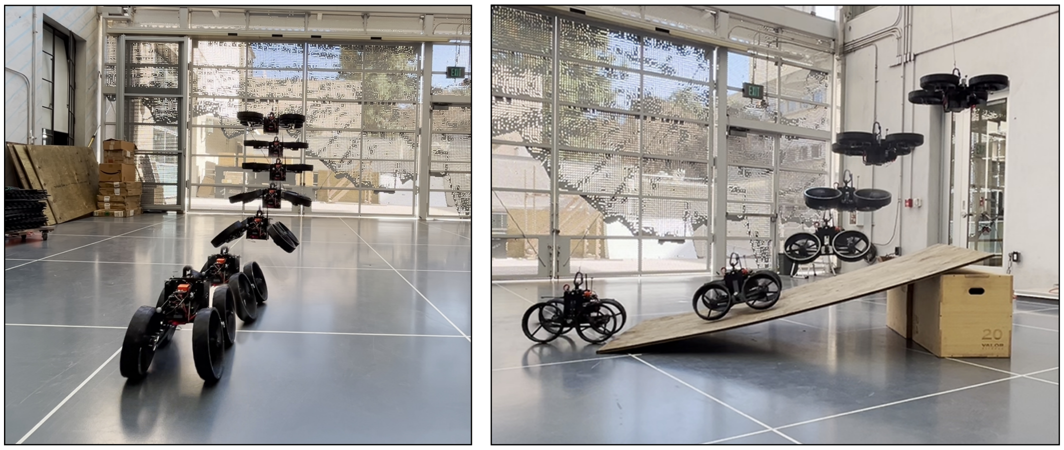


Figure 4.4: Experimental validation of ground aerial transition maneuvers: landing with forward velocity, and landing on inclined slope.

that hands off to ground locomotion upon touchdown. The MPC exploits the morphing quadcopter dynamics model, actuator limits, and ground-effect induced thrust variations to maintain stability across the full range of body postures. The resulting time and shape varying controller adapts continuously to altitude and morphing state, coherently linking the flight, transition, and wheel-landing phases.

To achieve this we used specialized objective functions for the flight, morphing, and morpho-transition phases of the maneuver. These are changed online in the optimization-based controller using a blending factor $\alpha(z, \varphi)$ which depends on the current robot altitude $z(t)$ and body tilt angle $\varphi(t)$. The blending factor smoothly switches the cost function for flight to a transition cost function as soon as ATMO enters the transition phase of the maneuver.

The controller exploits near-ground aerodynamics by intentionally relaxing regulation of vertical position and velocity upon entering the transition region. By harnessing the additional thrust available in ground effect, it prioritizes attitude stabilization and achieves the desired impact velocity, despite the inherent trade-off between attitude stabilization and impact speed at post-critical tilt angles. This use of ground effect makes gradual transformation advantageous over rapid reconfiguration in morpho-transition landings. Overall, the results of this chapter provide a model-based baseline for bi-modal ground-aerial locomotion control. The next chapter is an exploration of whether this performance can be improved using deep reinforcement learning, and how learning the morpho-transition maneuver through direct experience with a simulation environment can improve robustness.

Chapter 5

AIR-TO-GROUND MORPHO-TRANSITION VIA DEEP RL

This chapter incorporates material from the following publication:

Ioannis Mandralis, Richard M. Murray, and Morteza Gharib. "Quadrotor Morpho-Transition: Learning vs Model-Based Control Strategies". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2025. arXiv: `2506.14039 [cs.RO]`. (accepted).

In the previous chapter, tools from the field of optimization-based control were used to achieve bi-modal ground aerial locomotion for ATMO. Part of our effort was invested in solving the *morpho-transition* task, defined as the act of transitioning from air to ground locomotion by near ground aerial transformation. This locomotion phase was found to involve complex aerodynamic interactions and a need to operate near actuator saturation, complicating first-principles controller design. This chapter is an exploration of whether Reinforcement Learning, a growing paradigm for control based on experiential learning through direct interaction with an environment, can be applied to improve control performance.

First, Reinforcement Learning (RL) is briefly reviewed and the work on applying RL for control of aerial robots is surveyed. The development of a massively parallelized simulation environment for the Aerially Transforming Morphobot is then described, enabling fast evaluation of the robot dynamics. This environment is applied to train a reinforcement learning policy—entirely in simulation—using a specialized task definition, domain randomization, reward function, and algorithm that learns the morpho-transition maneuver from direct experience with the simulator. Techniques are discussed to achieve successful transfer of the learned control policy from simulation to hardware, and flight experiments are presented. Finally, the RL control policy is compared to the MPC controller and their robustness against unknown actuator failures is investigated.

5.1 A Brief Sketch of Reinforcement Learning

Loosely defined, RL refers to a set of methods for learning behavior in sequential decision making problems. In the RL framework, an intelligent actor, termed agent, learns to earn rewards through trial-and-error interaction with its environment. At each step of the decision making process, the agent uses its current state $s_j \in \mathcal{S}$ to select an action $a_j \in \mathcal{A}$. In part as a consequence of the action, the agent transitions to a new state s_{j+1} and receives a reward r_{j+1}. The long term accumulation of reward is used as a measure of how successful the agent is in the decision process.

The successive actions, or controls, are generated by a policy distribution $\pi(a_j|s_j)$ i.e. $a_j \sim \pi(\cdot|s_j) \in \mathcal{A}(s_j)$. To measure the quality of a particular policy π, the system dynamics are queried, when acting under the policy π and starting from state s, and the expected long-term cumulative reward, denoted by $V_\pi(s)$, is computed. This quantity, central to RL, is termed the *state-value function*. The objective of reinforcement learning is to find the *optimal policy* $\pi^*(a|s)$, defined as the policy under which the state-value function $V_\pi(s)$ is maximized for all starting states s.

In the case where the state and action space are discrete and finite, the central RL functions, including the policy, can be written as tables and learned incrementally using tabular learning methods. However, with large or continuous, state-action spaces, these methods suffer from the curse of dimensionality—computational requirements grow exponentially with the number of state variables making tabular learning methods intractable [84]. A solution to this issue has been to use deep neural-networks to approximate the central functions of RL. In this case, the method is then classified as a *deep reinforcement learning* method. In the following the terms RL and deep RL are used interchangeably.

5.2 Learning-Based Flight Control

Research in learning-based flight control has mainly focused on using RL to push the envelope of performance and agility for quadcopter flight. This effort is in part motivated by the need to develop flying robotic systems that withstand adverse wind conditions, perform acrobatic maneuvers, or achieve maximum speed and efficiency. Recently, this effort culminated to an outstanding achievement—an RL trained control policy outperformed the world champion in drone racing using a policy that outputs collective thrust and body rates that are then tracked by a lower-level body rate controller [85].

In this milestone, the decision to use an RL policy in tandem with a lower-level controller highlights one of the key issues in RL based flight control: deciding which level of the control hierarchy to inject the RL commands. As can be seen in a typical cascaded control architecture for a quadcopter (Figure 3.3), various

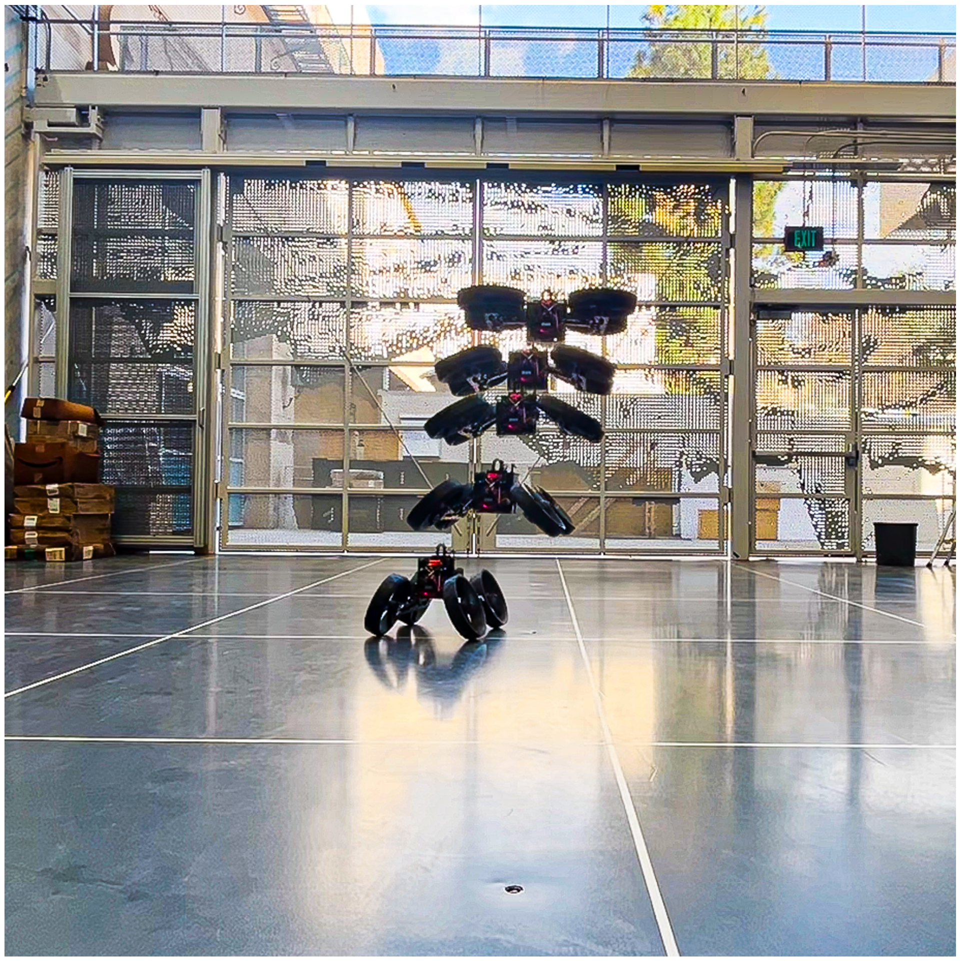


Figure 5.1: Snapshots of ATMO performing the Morpho-Transition maneuver using an end to end Reinforcement Learning control policy developed in this work. In this maneuver the robot starts in quadrotor flight mode and lands on its wheels by mid-air transformation.

controllers work in tandem to achieve stabilization and tracking. This gives the algorithm designer freedom to choose at which level of the control hierarchy the RL policy should be trained.

In a benchmark comparison between RL agents that were trained to supply control commands at different levels of the control hierarchy, [86] showed that control policies that commanded collective thrust and body rates struck an optimal balance between agility and robustness when compared to policies that specified higher level trajectories or single rotor thrusts. This study also showed that transferring policies trained to output single rotor thrust actions to the real world was not possible. Sensitivity to delays in the real system, on the order of 60ms, was cited as the primary reason for this failure.

However, other studies have shown that using direct thrust actions to control quadcopter flight is possible and can even be beneficial. Indeed, one of the earliest works on RL-based aerial control used an RL policy, combined with a low gain proportional-derivative attitude controller used during training only, to demonstrate stable quadcopter flight and recover from random initial states while outputting rotor thrusts directly [87]. The advantage of this approach is that directly outputting RPM commands or rotor thrusts removes the need for manually engineered lower level control loops. Moreover, giving the RL policy direct access to the actuators enables it to learn and push the true actuation limits of the platform. Other notable examples of using the end-to-end approach for quadcopter control can be found in [88, 86, 89, 90].

This end-to-end formulation is particularly appealing for ATMO during morphing flight and morpho-transition. As discussed in Chapter 3, the vehicle's geometry, mass properties, and thrust-vector directions vary during operation, requiring gain-scheduled inner loops and a non-trivial control structure. At the same time, during transition maneuvers the robot has to operate near the actuation limits of the platform—due to the reduced thrust authority induced by tilt—so an accurate treatment of saturation is essential. Granting the policy direct access to motor commands lets it reason over these dynamic tradeoffs without hand-crafted allocation logic. To our knowledge, no RL method has yet been applied to this class of mid-air shape-changing robotic systems.

To bridge this gap, we develop an RL method to achieve the morpho-transition maneuver on ATMO. The RL agent autonomously learns to stabilize ATMO, reject disturbances, and finds a feasible trajectory to solve the task from a single high level reward objective. Unlike the MPC method developed in the previous chapter, where special care has to be taken when approaching the ground due to the switching from flight dynamics to ground dynamics, the RL method does not need to explicitly account for ground contact, but rather learns it directly from the simulation, enabling smooth control across the whole flight-drive operational envelope. Our RL method operates at the RPM command level making this an end-to-end approach which bypasses lower level body rate or attitude control loops whose implementation requires extensive manual tuning, domain knowledge, and engineering effort.

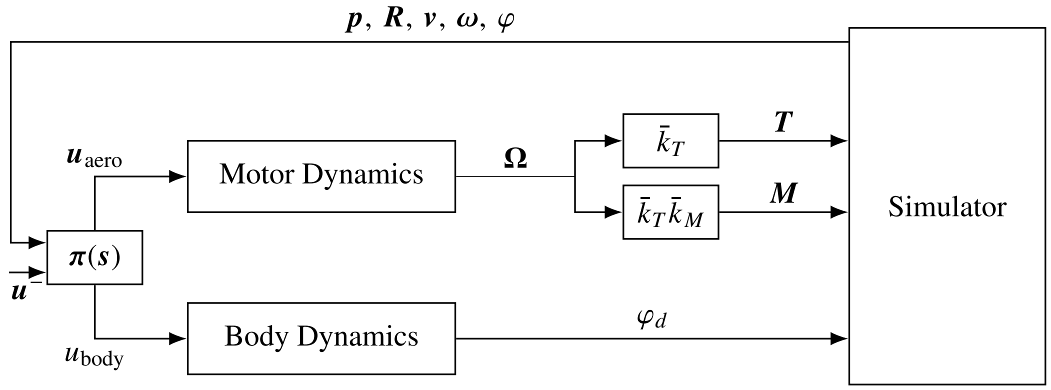


Figure 5.2: Overall schematic of signal flow from RL policy $\pi(s)$ to motor dynamics, body dynamics, aerodynamics, and to the Isaac Lab simulator.

5.3 Simulator Development

To train a control policy for morpho-transition maneuvers, we first modeled and simulated the Aerially Transforming Morphobot as an extension of the state-of-the-art Isaac Lab simulation framework [91, 92]. ATMO's inertial and kinematic parameters, reported in Chapter 2 and Appendix D, were used to write the robot in the *Unified Robot Description Format* (URDF). This robot description contains the kinematic relations and inertial parameters of the seven main components: the robot base, the left arm and right arm, and the four spinning propellers. The wheel joints were fixed and did not enter the URDF description since they were not used for stabilization during aerial maneuvers. The contact of ATMO with the ground was handled internally in Isaac Lab using the CAD model of ATMO to resolve collisions.

The motion of the morphing mechanism, depicted in Figure 2.4, was not simulated directly. We found that simulating the closed loop kinematic chain caused numerical instabilities, as has also been observed in [93, 94]. Instead, the desired tilt angle φ_d was tracked by two proportional-derivative position controllers (one for each arm joint), with stiffness $k_p = 1 \times 10^{15}$ and damping $k_d = 1 \times 10^5$. Thus, the closed loop morphing mechanism was replaced by the direct motion of the arm link CDE around the pivot point D (see Figure 2.4) using a single velocity-level input. The large gains chosen for the proportional-derivative controller made the actuator a pure velocity controller ensuring the mechanism is stiff—approximating its real world, self-locking characteristics.

The thrust force and drag torque caused by the rotating propellers were applied to the propeller links in the URDF model through the interface provided by Isaac Lab. As detailed in Section 3.1, the thrust and drag torque are linear functions of the normalized propeller RPMs and act parallel to the rotor axes, defined in the positive body z axis:

$$\boldsymbol{T}_j = \bar{k}_T u^j_{\text{aero}} \hat{\boldsymbol{z}}_j \tag{5.1}$$

$$\boldsymbol{M}_j = (-1)^j \bar{k}_M \bar{k}_T u^j_{\text{aero}} \hat{\boldsymbol{z}}_j, \tag{5.2}$$

where j denotes the propeller number $i = \{1, \dots, 4\}$ and u_{aero} is the normalized propeller RPM input command. The thrust and moment coefficients $\bar{k}_T, \bar{k}_M$ were identified experimentally through thrust stand system identification experiments in Section 2.3 and the values $\bar{k}_T = 28$ N and $\bar{k}_M = 0.018$ m, were used. Here, $\hat{z}_j$ denotes the rotation axis of propeller j. Second order aerodynamic effects like proximity effects, propeller flapping, or vehicle rotor dynamic couplings [95] were ignored.

The propeller motor dynamics, presented in Section 3.2, as well as the morphing mechanism motion, were implemented in a custom plugin with dynamics:

$$\begin{cases} \dot{\boldsymbol{\Omega}} = T_m^{-1}(\boldsymbol{u}_{\text{aero}} - \boldsymbol{\Omega}) & (5.3) \\ \dot{\varphi} = \dot{\varphi}_{\max} u_{\text{body}}, & (5.4) \end{cases}$$

where $\boldsymbol{u}_{\text{aero}} \in [0, 1]^4$ denote the normalized propeller RPMs and $u_{\text{body}} \in [-1, 1]$ is the commanded tilting velocity. The motor dynamics of the thrusters are first order linear systems with input $\boldsymbol{u}_{\text{aero}}$ and output the normalized propeller RPMs $\boldsymbol{\Omega}$. The normalized propeller RPMs are the ratio of the squared propeller rotational speed to the square of the maximum propeller rotational speed. The thruster motor constant is T_m and the maximum speed of the closed loop kinematic linkage is $\dot{\varphi}_{\max}$.

The simplified dynamics models for the closed loop kinematic linkage, aerodynamics models, and the motor dynamics were implemented in discrete time in the pre-physics step of the simulation, by performing the following operations:

$$\boldsymbol{\Omega}[k+1] = \beta \boldsymbol{u}_{\text{aero}}[k] + (1-\beta)\boldsymbol{\Omega}[k] \tag{5.5}$$

$$\boldsymbol{T}_j[k+1] = \bar{k}_T \boldsymbol{\Omega}_j[k+1] \hat{\boldsymbol{z}}_j \tag{5.6}$$

$$\boldsymbol{M}_j[k+1] = (-1)^j \bar{k}_M \bar{k}_T \boldsymbol{\Omega}_j[k+1] \hat{\boldsymbol{z}}_j \tag{5.7}$$

$$\varphi_d[k+1] = \varphi[k] + \dot{\varphi}_{\max} u_{\text{body}}[k] T_s. \tag{5.8}$$

Equation (5.5), is an exact discretization of motor dynamics in Equation (5.3), under the assumption of zero-order hold between simulation time-steps, leading to a first-order filter with constant,

$$\beta = 1 - e^{-T_s/T_m}. \tag{5.9}$$

A nominal motor time constant of $T_m = 0.15$ was used. This was determined experimentally by supplying the motor with step throttle inputs and calculating the time required to reach 63% of the steady state rotational rate. Since the simulator time step was $T_s = 0.02$ seconds, the resulting filter constant was: $\beta = 0.12$. The filtered normalized motor RPMs are then used to determine the thrust and moment acting on the four propeller axes, $\boldsymbol{T}_j, \boldsymbol{M}_j$ at each time step in Equations (5.6)–(5.7). Equation (5.8) is a forward Euler step of the tilt angle dynamics in Equation (5.4).

The Isaac Lab simulator solved the robot equations of motion, including collisions, using a time step of $T_s = 0.02$ seconds. At each time step, the simulator updated ATMO's generalized coordinates and velocities, defined as: $\boldsymbol{q} = (\boldsymbol{p}, \boldsymbol{\xi}, \varphi)$ and $\boldsymbol{w} = (\boldsymbol{v}, \boldsymbol{\omega})$ where $\boldsymbol{p}$ is the position of the robot center of mass, $\boldsymbol{\xi}$ is the quaternion of the robot base frame relative to the inertial world frame, φ is the body tilt angle, $\boldsymbol{v}$ is the velocity of the center of mass, and $\boldsymbol{\omega}$ is the angular velocity of the robot relative to the world frame expressed in the body frame. The overall simulation environment is summarized in Figure 5.2.

5.4 Training Approach

The developed simulation was used to train an RL policy to solve the task of morpho-transition. In this section we describe the essential components to train a control policy in simulation that achieves zero-shot transfer to hardware.

Task Definition & Domain Randomization

At the beginning of each episode ATMO was spawned uniformly at a random (x_0, y_0) position in a square of 2 meters side length around the origin. The initial height was sampled from a uniform distribution: $z_0 \sim \mathcal{U}(1, 2)$ meters from the ground. The goal state, i.e. the desired final position, was set as the origin. For more information about how the robot was incentivized to reach the goal state, as well as the desired final configuration, the reward function is discussed at the end of this section.

The initial orientation relative to the inertial frame was randomized by sampling a roll and pitch from $\mathcal{U}(-\frac{\pi}{6}, \frac{\pi}{6})$ and an initial yaw angle ψ_0 from $\mathcal{U}(0, 2\pi)$. Initial velocities, $\boldsymbol{v}_0$, and angular velocities, $\boldsymbol{\omega}_0$ were sampled uniformly from the distribution $\mathcal{U}(-0.1, 0.1)$. The initial tilt angle was sampled randomly as $\varphi_0 \sim \mathcal{U}(0, \pi/6)$. Each episode ended if one of the following conditions were met:

1. The elapsed simulation time exceeded five seconds.
2. The robot speed exceeded a maximum threshold of 2.0 ms^{-1}
3. The (x, y) position of the robot deviated from the goal position by more than 1.5 meters.

The policy was made robust to disturbances by pushing the robot in random directions once per episode. To achieve this, at the beginning of each episode a push time, t^*, and push duration, T^*, were sampled from uniform distributions $t^* \sim \mathcal{U}(0, 0.8t_f), T^* \sim \mathcal{U}(0, 0.2)$, where $t_f = 5$ seconds is the episode length. The force and moment direction and intensity were sampled at the beginning of each episode and applied in the pre-physics step of the simulator for the given duration at the push time. The disturbance force and torque $\boldsymbol{f}, \boldsymbol{\tau}$ were generated as follows:

$$\boldsymbol{f} = f\hat{\boldsymbol{n}}_f, \boldsymbol{\tau} = \tau\hat{\boldsymbol{n}}_\tau, \tag{5.10}$$

$$\boldsymbol{n}_f \sim \mathcal{N}(\boldsymbol{0}, \mathbb{I}_3), \boldsymbol{n}_\tau \sim \mathcal{N}(\boldsymbol{0}, \mathbb{I}_3), \tag{5.11}$$

$$f \sim \mathcal{N}(0, f^*), \tau \sim \mathcal{N}(0, \tau^*), \tag{5.12}$$

$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{n}}{||\boldsymbol{n}||}, \tag{5.13}$$

with the force and moment scales set to $f^* = 0.20k_T$ and $\tau^* = 0.20k_Tk_M$. The maximum impulse and angular impulse imparted on the robot during training was thus $J^{\text{max}}_{\text{disturbance}} = 0.04k_T$ and $\Delta L^{\text{max}}_{\text{disturbance}} = 0.04k_Tk_M$.

The motor time constant was randomized around the nominal value of $T_m = 0.15$ seconds at the beginning of each episode $T_m \sim \mathcal{U}(0.10, 0.20)$, and kept fixed throughout the episode. To take into account potential mismatch between the motor dynamics in the simulation and the hardware, the thrust and moment constants were assumed to have a ±20% uncertainty. Thus, they were sampled at the beginning of each episode as: $\bar{k}_T \sim \mathcal{U}(0.8\bar{k}_T, 1.2\bar{k}_T)$ and $\bar{k}_M \sim \mathcal{U}(0.8\bar{k}_M, 1.2\bar{k}_M)$. Finally, the uncertainty of the tilt actuator dynamics was incorporated by randomizing the maximum tilt velocity parameter $\dot{\varphi}_{\text{max}} \sim \mathcal{U}(0.8\frac{\pi}{8}, 1.2\frac{\pi}{8})$. We found that training without randomizing these parameters results in neither sim-to-sim nor sim-to-real transfer.

Action Space

The actions are the outputs of the trained RL policy, $\boldsymbol{\pi}(\boldsymbol{s})$. For the policy network we used three hidden layers of 128 units each with Exponential Linear Unit (ELU) activation functions between the layers. The network outputs were passed through a sigmoid layer to ensure the control actions are bounded between 0 and 1, i.e. $\boldsymbol{u} = \boldsymbol{\pi}(\boldsymbol{s}) \in [0, 1]^5$, and can be effectively used to represent the normalized propeller RPMs and the tilt velocity. The outputs of the policy network are interpreted as the four thruster control inputs $\boldsymbol{u}_{\text{aero}}$ and the tilt angle velocity input u_{body}.

The inputs to the policy network are the policy observations,

$$\boldsymbol{s}_\pi = (\boldsymbol{p}, \boldsymbol{R}, \boldsymbol{v}, \omega, \varphi, \boldsymbol{u}^-) \in \mathbb{R}^{19+5n}, \tag{5.14}$$

where $\boldsymbol{u}^-$ is an observation history of $n = 10$ previous time steps. For the rotation representation we found that it was necessary to use the full rotation matrix $\boldsymbol{R} \in SO(3)$. Using a Quaternion representation of rotation did not result in sim-to-real transfer since quaternions double cover the space of rotations, meaning that the policy network must learn that $\boldsymbol{\xi}$ and $-\boldsymbol{\xi}$ represent the same rotation.

Observation Space

We used an asymmetric actor critic [96] scheme and gave fourteen additional privileged observations to the critic to improve value estimation. The critic observation space was chosen as

$$\boldsymbol{s}_Q = (\boldsymbol{s}_\pi, \boldsymbol{f}, \boldsymbol{\tau}, t^*, T^*, t, J_c, \boldsymbol{\Omega}^-) \in \mathbb{R}^{33+5n}, \tag{5.15}$$

where $\boldsymbol{f}, \boldsymbol{\tau}$ are the disturbance force and moment, t^*, T^*, t are the push time, push duration, and current time, and J_c is the impulse acting on the robot due to ground contact forces. This was approximated between simulation steps as

$$J_c = \int_t^{t+T_s} \boldsymbol{f}_c dt \simeq [\boldsymbol{f}_c(k+1) - \boldsymbol{f}_c(t)]\, T_s, \tag{5.16}$$

where $\boldsymbol{f}_c$ is the contact force between the ground and the robot, obtained from the Isaac Lab simulation environment. We gave the critic network access to the observed RPM values $\boldsymbol{\Omega}^-$ which was shown in [89] to stabilize training. A small amount of uniform noise, was added to the policy observations. We added uniform noise of magnitude 0.005 to the position $\boldsymbol{p}$ and rotation matrix $\boldsymbol{R}$, 0.035 to the velocity $\boldsymbol{v}$ and angular velocity $\boldsymbol{\omega}$, and 0.018 to the tilt angle φ. No noise was added to the action history.

Delays & Motor Dynamics

Finally, we found that for successful sim-to-real transfer, it was essential to add an observation delay of one simulation time step, or 20 ms, to the observations fed into the policy network during train time. This approximates the delays present due to inter-computer communication on the hardware. Without taking into account observation delays, as well as motor dynamics, the RL control policy could not be transferred to hardware.

Reward Function

Coefficient	**Value**	**Description**
a_0	$-0.10\,T_s$	Velocity Penalty
a_1	$-0.30\,T_s$	Angular Velocity Penalty
a_2	$-0.10\,T_s$	Orientation penalty
a_3	$-0.07\,T_s$	Control Rate Penalty
a_4	$-0.13\,T_s$	Ground Thrust Penalty
a_5	-1.0	Contact Impulse Penalty
a_6	0.40	Contact in Acceptance Reward
a_7	$0.30\,T_s$	Distance to Goal Reward
a_8	$0.40\,T_s$	Morphing reward
a_9	$0.30\,T_s$	Descending reward

Table 5.1: Coefficients for the reward function defined in Equation (5.18).

The reward function was designed to incentivize landing on the wheels in the vicinity of the goal state. The main term that produced this behavior was

$$r_c^w = (c_w \wedge a). \tag{5.17}$$

Here, c_w is a Boolean variable active when any of the wheels make contact with ground and a is a Boolean variable denoting that the robot is in the acceptance state, defined as the state where the (x, y) position of the robot is within 40 cm from the goal position.

To ensure that the robot had sufficiently rich reward information, and performed the maneuver safely, for example by keeping angular and linear velocities, as well as ground impact speed low, various reward shaping terms were added. First, we penalized the linear and angular velocities of the robot base frame, the action rate, and deviations from flat orientation. We rewarded the distance to the (x, y) coordinates of the goal position as well as descending at a constant rate of 0.5ms^{-1}. We also rewarded high tilt angles as the robot approaches the ground. Finally, we took advantage of the ability to accurately measure contact times and forces in the simulator to penalize large impulse forces with the ground as well as to penalize undesirable thrust actions that occur while the robot is in contact with the ground. We imposed a penalty of -1 unit for early termination due to conditions 1–3 described above. The final reward function expression is

$$\begin{aligned} r(\boldsymbol{s},\boldsymbol{u}) = a_0||\boldsymbol{v}||^2 &+ a_1||\boldsymbol{\omega}||^2 + a_2|1 - q_a| \\ &+ a_3||\boldsymbol{u} - \boldsymbol{u}^-||^2 + a_4 c_0||\boldsymbol{u}_{\text{aero}}||^2 \\ &+ a_5 J_c^2 + a_6(c_w \wedge a) \\ &+ a_7\phi(d) + a_8\phi(z^2 + e_\varphi^2) \\ &+ a_9\phi(e_{v_z}^2). \end{aligned} \tag{5.18}$$

Here, $\phi(\cdot)$ is the function $\phi(x) = e^{-4x}$. q_a denotes the axis of the quaternion, c_0 is a boolean variable that is equal to 1 when the first contact with the ground has occurred, d is the distance to the (x, y) position of the goal state, and $e_\varphi = \varphi - \frac{\pi}{2}$. The coefficients $a_1 \dots a_9$ were chosen by manual tuning and are reported in Table 5.1.

Algorithm Selection and Training Parameters

We used the Proximal Policy Optimization algorithm (PPO) [97] and exploited massively parallel training on GPU [93]. We used the RL-games implementation [91] with an initial learning rate of $\lambda = 1 \times 10^{-5}$. Our model was trained for 1,000 policy updates after which the reward had converged to a stable value. This took about 20 minutes on a GPU-enabled desktop computer. An NVIDIA GeForce RTX 4070 graphics card with 16GB memory was used.

5.5 Model Predictive Controller Implementation

The model predictive controller for the morpho-transition maneuver was detailed in Chapter 4 and summarized here for completeness. The controller is a trajectory tracking MPC that receives a desired reference $x_{\text{ref}}, u_{\text{ref}}$ and solves the following optimal control problem iteratively in a receding horizon control loop:

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{u}}{\mathit{minimize}} \quad & \int_0^{t_f} L(\boldsymbol{x},\boldsymbol{u})dt \\ \text{subject to} \quad & \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}), \qquad \forall t \in [0, t_f], \\ & \boldsymbol{u}_{\text{aero}} \in [0,1]^4. \end{aligned}$$

The dynamics are discretized and enforced using multiple shooting [77] with N collocation points and a look-ahead time t_f. A nonlinear program is solved using sequential quadratic programming in a real-time iteration scheme onboard the

NVIDIA Jetson Orin Nano using the software packages CasADi and ACADOS [78, 79]. The controller runs at 150Hz. A look-ahead of $t_f = 2.0$ seconds and $N = 20$ collocation points are used. The cost function is given by

$$L(\boldsymbol{x}, \boldsymbol{u}) = \alpha(\boldsymbol{x}) \underbrace{L_1(\boldsymbol{x}, \boldsymbol{u})}_{\text{flying}} + (1 - \alpha(\boldsymbol{x})) \underbrace{L_2(\boldsymbol{x}, \boldsymbol{u})}_{\text{transitioning}}, \tag{5.19}$$

and is a convex combination of quadratic costs specialized for flying and transitioning. The cost is varied online according to the height and body shape through the scalar function $\alpha(\boldsymbol{x})$. The MPC controller assumes that the RPM commands $\boldsymbol{\Omega}$ are equal to the control commands i.e. $\boldsymbol{u}_{\text{aero}} = \boldsymbol{\Omega}$, thus operating in the reduced state space $\boldsymbol{x} = (\boldsymbol{p}, \boldsymbol{\xi}, \boldsymbol{v}, \boldsymbol{\omega})$. The tilt angle φ is supplied from an external reference generator and tracked by a low level controller. For details on the implementation of the MPC algorithm, the reader is referred to [98].

5.6 Experimental Validation

We performed different Morpho-Transition maneuvers in the CAST flight arena at Caltech, using the MPC and the RL controllers. Optitrack motion capture information was used to supply position and orientation estimates to an Extended Kalman Filter running onboard ATMO removing the need for GPS or vision based localization. ATMO was initialized on the ground and flown up to 1.25 meters using the PX4 quadrotor position controller [53]. At that point, the RL or MPC controllers were engaged and the morpho-transition landing maneuver was performed. Snapshots from a representative experiment are depicted in Figure 5.1. Here ATMO can be seen starting from its initial quadrotor configuration at 1.25 meters from the ground and transforming mid air to land on wheels. This experiment was performed without a tether and repeated multiple times to ensure reproducibility.

The evolution of key states for the morpho-transition maneuver performed using the RL policy and the MPC controller are shown in Figures 5.3 and 5.4, respectively. Videos of the corresponding maneuvers are also available as supplementary materials.

Both controllers were successfully able to complete the task. The final tilt angle achieved by the MPC controller was $\varphi = 60°$ and the RL controller achieved $\varphi = 65°$. The RL controller achieved a preferable impact velocity of 0.5ms^{-1}, as opposed to the 1.0ms^{-1} impact achieved by the MPC controller. Remarkably, this was achieved with a landing tilt angle beyond the actuator saturation tilt angle.

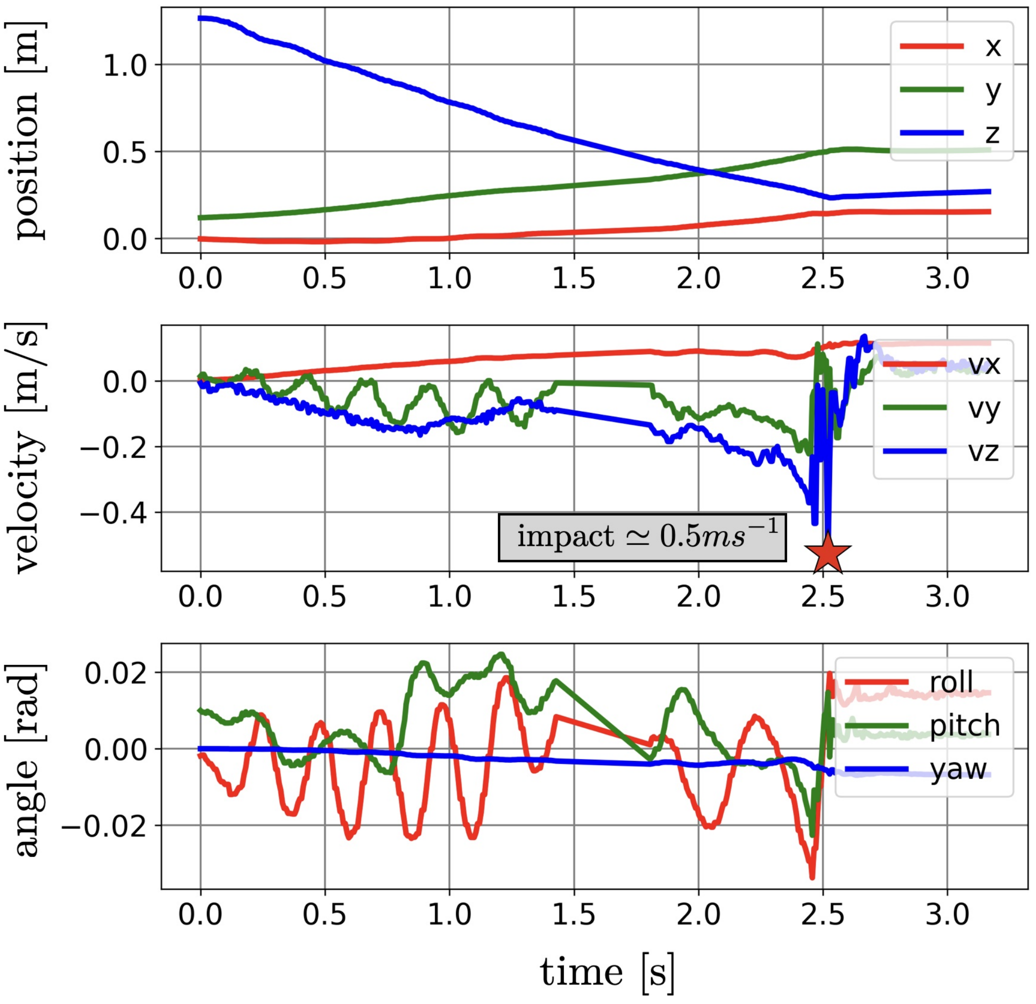


Figure 5.3: Key state evolution for morpho-transition maneuver performed on the hardware using the end-to-end learned RL policy.

On the other hand, the RL controller exhibited larger oscillations in the roll angle evidencing slight instability during the descent. We believe that this was due to inexact estimation of the observation delay of 20 ms as well as in the motor dynamics time constant. Indeed, in our initial experiments, before adding observation delays to the Isaac Lab simulation environment, the transfer of the RL policy to hardware was unsuccessful due to very large roll and pitch oscillations. As we improved our estimate of the delay the oscillations were reduced. The final result in Figure 5.3 is satisfactory. The effect of delays can be further mitigated by inferencing the RL control policy directly on the flight controller micro-controller hardware rather than on the companion computer, reducing the latency caused by streaming the control

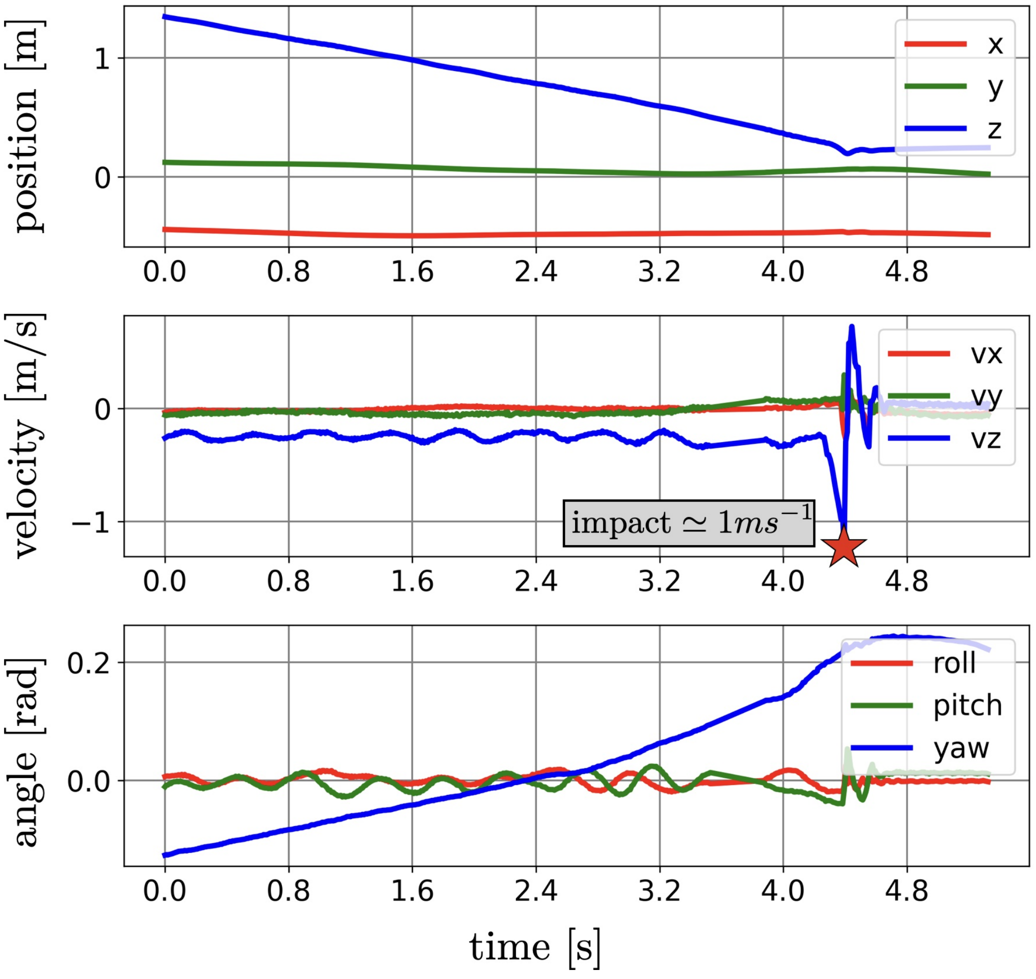


Figure 5.4: Key state evolution for morpho-transition maneuver performed on the hardware using the end-to-end learned MPC policy.

commands through the ROS2 network. Overall, the MPC method exhibits more stable angular dynamics but shows significant drift in yaw.

5.7 Recovery from Disturbances

To obtain a deeper understanding of the performance of the two methods, we exploited the parallel simulation environment of Isaac Lab to test their ability to recover from disturbances. We examined the effect of the direction and magnitude of a push disturbance on both controllers. To achieve this, a push force was applied systematically at different angles and magnitudes in the first quadrant of the body (x, y) plane at the robot's center-of-gravity, as shown in Figure 5.5. The angle was varied in increments of $\frac{\pi}{16}$ in the first quadrant of the body (x, y) plane. ATMO's

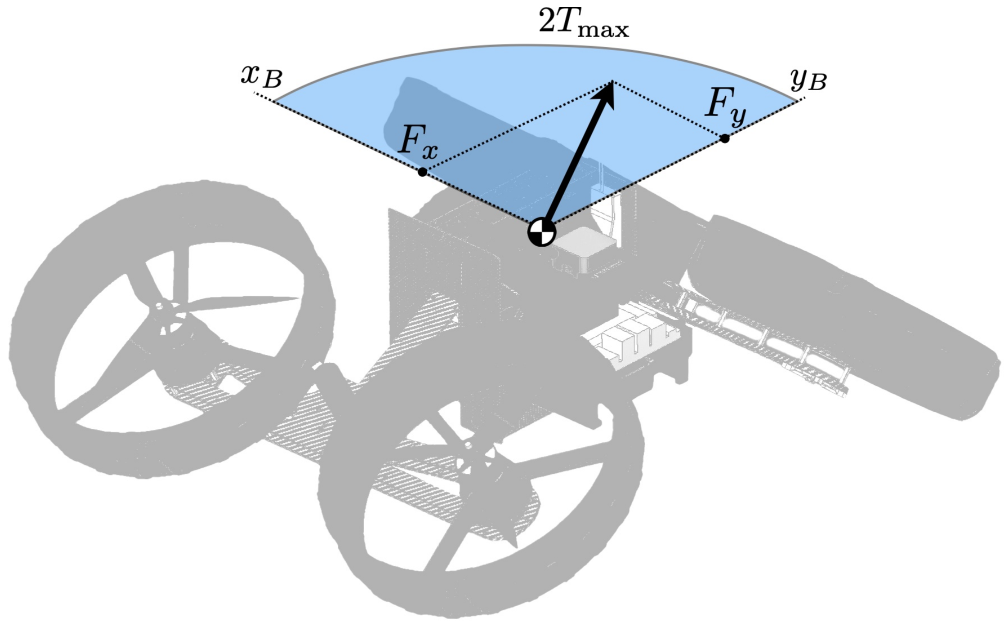


Figure 5.5: Schematic of the disturbance forces that ATMO is subject to in order to evaluate the effectiveness of the RL approach compared to the MPC controller. The disturbance is generated in the first quadrant of a circular disk which intersects ATMO's center of mass. The forces are sampled with changing direction and magnitude within this disk. The maximum disturbance force is equal to $2T_{\text{max}}$, where $T_{\text{max}} = \bar{k}_T$ is the maximum thrust force that can be generated by a single propeller.

symmetry did not require us to test push forces in other quadrants since they would be equivalent. The magnitude of the force was varied in increments of $0.925\bar{k}_T$ from $0.6\bar{k}_T$ to $8.0\bar{k}_T$. The maximum disturbance force that the robot underwent was of magnitude 2 times greater than the overall thrust capability of the robot (four propellers generating $T_{\text{max}} = \bar{k}_T$, each, generates a total thrust of $4\bar{k}_T$).

The MPC method was tasked with following a downward descending trajectory at 0.5ms^{-1} and the RL agent performed actions according to the policy learned during the training process. The simulation was configured with with $T_m = 0.15$ and 20 ms observation delay.

To characterize the ability of the controllers to recover from push disturbances, two metrics were employed: 1) the impact velocity, and 2) the final distance from the goal. For each applied disturbance force, the control policies were rolled out in simulation for 5 seconds and the impact velocity and final distance to the goal were recorded. Lower impact velocity and lower final distance to goal indicate that the

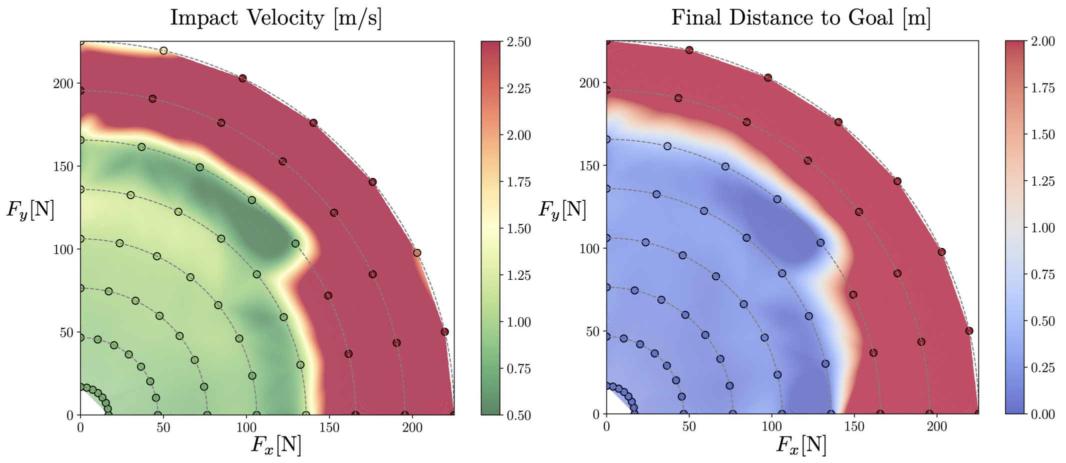


Figure 5.6: Disturbance recovery performance of RL policy. Impact velocity and final distance maps are depicted for different disturbance directions and magnitudes. The impact velocity and final distance to goal were recorded for 64 different push forces for each controller. The direction of the push force was varied in the (x, y) plane and the magnitude of the push force was varied between $0.60\bar{k}_T$ and $8.0\bar{k}_T$. The data points are shown in a scatter plot with an interpolated heat map overlayed.

controller was able to successfully recover from the push disturbance.

Heat maps of the performance of the results according to the two metrics were generated and shown in Figures 5.6 and 5.7. In the case of the RL method, the control policy performed very well up to between $5 - 6\bar{k}_T$ disturbance force, as shown in Figure 5.6. In this region, a low impact velocity, close to 0.5ms^{-1}, was observed, as rewarded by the reward function. The final distance to the goal was also close to zero. In this region of disturbances, the RL agent was able to fully recover from the pushes. This is notable since this was far beyond what the RL policy had been trained on—evidencing that the RL policy can generalize outside the training data points. However, once the threshold of $5 - 6\bar{k}_T$ was surpassed, the RL agent failed to land with an acceptable impact velocity or final distance to goal.

On the other hand, the MPC controller, with results shown in Figure 5.7, exhibited a more uniform performance. It was able to recover reasonably well from large disturbances but did not exhibit a uniform region of excellent disturbance recovery performance like the RL controller. Indeed, although the MPC controller is able to arrive at the goal position with reasonable success rate, as evidenced, it also often impacts the ground with unacceptable impact velocity.

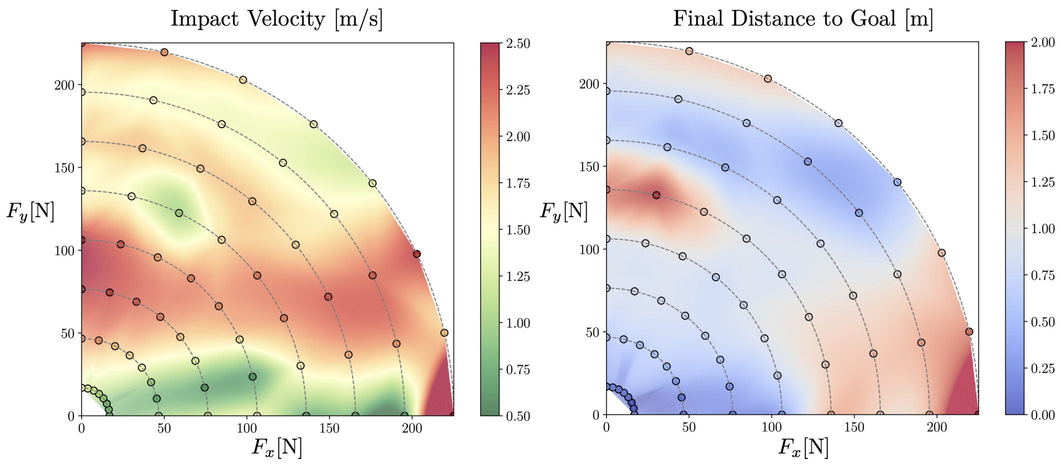


Figure 5.7: Disturbance recovery performance of MPC controller. Impact velocity and final distance maps are depicted for different disturbance directions and magnitudes. The impact velocity and final distance to goal were recorded for 64 different push forces for each controller. The direction of the push force was varied in the (x, y) plane and the magnitude of the push force was varied between $0.60\bar{k}_T$ and $8.0\bar{k}_T$. The data points are shown in a scatter plot with an interpolated heat map overlayed.

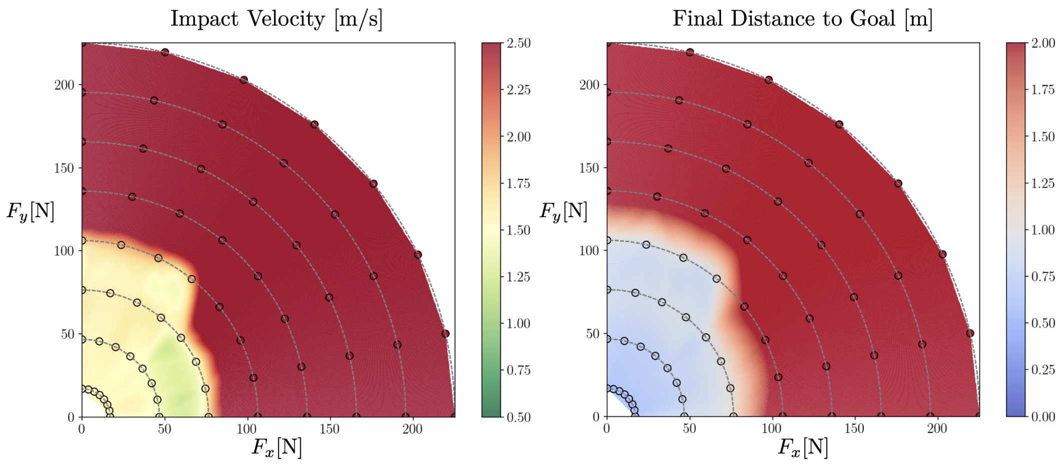


Figure 5.8: RL recovery characteristics under partial actuator failure. The thrust and moment coefficients of the thrusters are multiplied by $[0.8, 0.9, 0.85, 1.1]$ with order front right, back left, front left, back right.

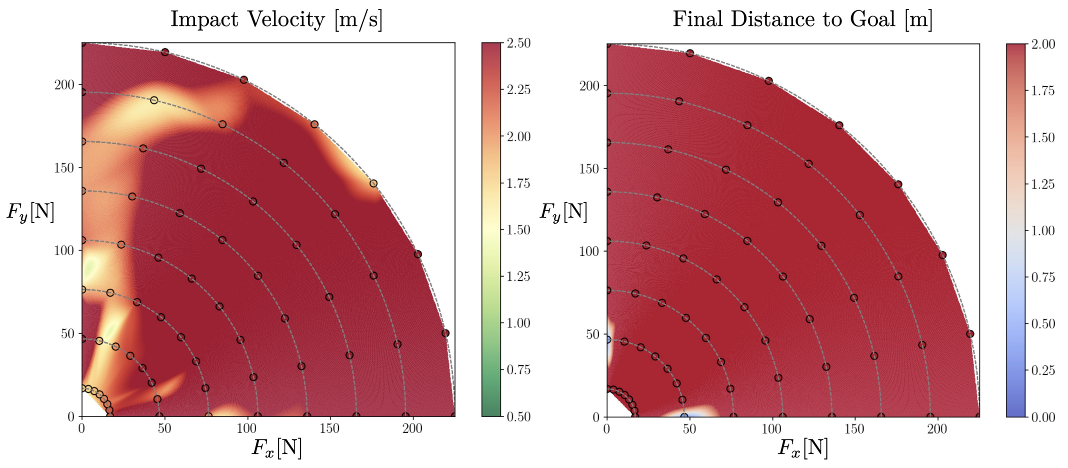


Figure 5.9: MPC recovery characteristics under partial actuator failure. The thrust and moment coefficients of the thrusters are multiplied by $[0.8, 0.9, 0.85, 1.1]$ with order front right, back left, front left, back right.

5.8 Recovery from Partial Actuator Failure

Finally, we tested the performance of the two controllers under partial thruster failure. This can happen in real-world scenarios due to motor wear or low energy state. To simulate this scenario, we multiplied the nominal value of the thrust and moment coefficients by $[0.8, 0.9, 0.85, 1.1]$. This is a significant perturbation that the RL has not been trained on since it was trained only on uniform changes in the thrust and moment coefficients. Likewise, the MPC has no knowledge of this perturbation.

The results are shown in Figures 5.8 and 5.9. The RL controller is able to recover from considerable disturbance forces, even under the partial actuator loss. Naturally, the recovery region is much smaller than in the case where the thrusters are operating at nominal efficiency (Figure 5.6) but the RL agent is still able to recover from considerable applied impulses. In contrast, the MPC controller cannot recover at all, resulting in failure for almost all push tests considered. Note that it is possible to extend the MPC algorithm to take into account actuator failure as done in [72], which is expected to significantly improve performance. However, this would require explicit estimation of the actuator failure to trigger online changes in the optimal control problem being solved by the MPC.

5.9 Summary

In this chapter, we trained an end-to-end Reinforcement Learning (RL) controller to learn a morpho-transition policy and demonstrated successful transfer to hardware. The RL policy achieved controlled landings with tilt angles greater than 65° in experimental flight tests—on par and slightly better than the performance of the model-predictive controller. However, the RL policy successfully transferred to hardware only if motor dynamics and observation delays were taken into account. In contrast, the MPC controller developed in Chapter 4 transferred out-of-the-box without knowledge of the actuator dynamics and delays, at the cost of reduced recovery from disturbances in the event of unknown actuator failures. The main contributions of this chapter are:

1. We developed a reward function that accurately encodes the problem of morpho-transition by taking advantage of the perfect information available in the simulator.

2. We proposed a performance benchmark for the morpho-transition maneuver based on impact velocity and final distance to the goal after push disturbances of different magnitude and direction. The controllers were thoroughly characterized by exploiting massively parallel GPU simulation.

3. We extended the Isaac Lab simulation framework to include common quadrotor aerodynamics such as the thrust and moment coefficients, as well as motor dynamics and observation delays for the specialized tasks requiring aerial transformation. This code, along with the MPC method, has been made publicly available here.

Chapter 6

WAKE VECTORING FOR EFFICIENT MORPHING FLIGHT

This chapter incorporates material from the following publication:

Ioannis Mandralis, Severin Schumacher, and Morteza Gharib. Wake Vectoring for Efficient Morphing Flight. 2025. (under review).

Chapters 4 & 5 were dedicated to the development of controllers that enabled agile behaviors, such as morpho-transitions and morphing flight. In this chapter, we extend these capabilities by drawing on aerodynamic principles from aircraft and rocket design to develop a wake-vectoring mechanism that recovers thrust lost during morphing, thereby enhancing our robot's agility and maneuverability.

We begin by introducing the broader concept of flow manipulation as a means of improving the agility and efficiency of aerial vehicles. Building on this foundation, we present our wake-vectoring solution and validate it through benchtop experimental testing. Our design is then refined through a computational fluid dynamics study, and is integrated onto our robotic platform where it is characterized under realistic operating conditions. Finally, we demonstrate ATMO's ability to hover in extreme aerial configurations beyond the critical tilt angle, and conclude with a discussion of the broader implications of our results.

6.1 Introduction

Principles of airflow manipulation have long been used to enhance aircraft performance in the field of aircraft and rocket design. In rocketry, for example, gimbaled or vectoring nozzles redirect the engine exhaust, which is crucial for steering the vehicle when aerodynamic surfaces are ineffective, e.g. in vacuum [100]. In general, directing engine thrust off-axis provides key benefits including vertical/short takeoff and landing (VTOL/STOL) ability, and significantly higher agility in flight through thrust reversal or vectoring [101, 102]. A notable example is the Harrier Jump Jet, which used a rotating nozzle system to vector its engine exhaust downward—giving it the ability to take off and land vertically on very short runways [103]. Other fighter jets likewise use vectored thrust to perform maneuvers that exceed the normal aerodynamic envelope, illustrating how airflow manipulation translates into enhanced maneuverability [104].

This insight suggests that morphing aerial robots could reap substantial benefits by incorporating flow re-direction mechanisms into their designs. By reconfiguring their structure or deploying internal deflectors to divert the rotor wake, a morphing drone can recover otherwise-wasted airflow momentum and boost useful thrust.

The potential of this approach has been demonstrated in prior work on unmanned aerial vehicles (UAVs). Examples include shape changing platforms that employ movable flaps to enable both multi-rotor and efficient fixed-wing flight [105, 106, 107, 108], single-rotor systems equipped with adjustable vanes to redirect thrust without additional actuators [109], and quad-rotor vehicles that use deflectors in the propulsive slipstreams to apply lateral forces during hover without altering propeller speeds [110]. Collectively, these studies highlight the promise of integrating flow manipulation principles into robotic fliers to boost thrust efficiency, enhance control authority, and enable multi-modal flight through mode-switching capability.

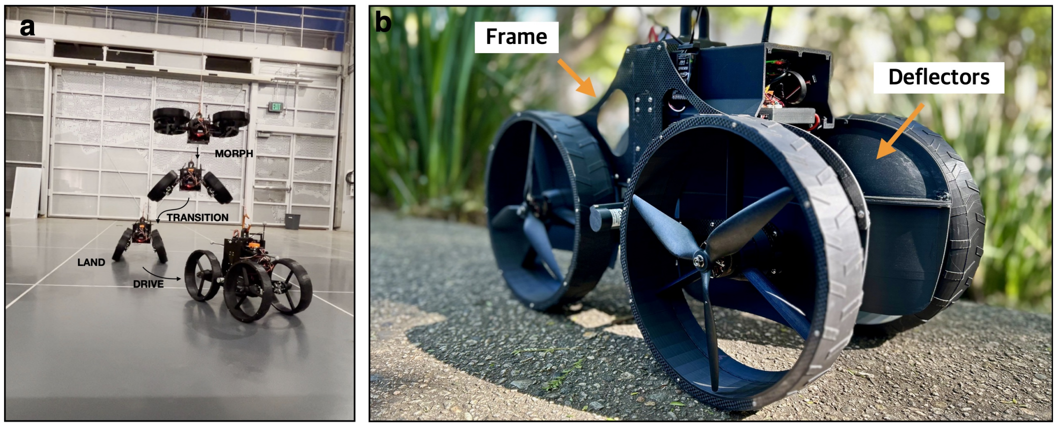


Figure 6.1: Proposed wake vectoring solution. (a) Example of a Morphing Aerial Robot, the Aerially Transforming Morphobot (ATMO). This robot is capable of the Morpho-Transition maneuver where the robot transitions from flying to driving in one smooth aerial transformation maneuver. During this maneuver, the rotors are tilted away from the vertical axis resulting in lower thrust available in the vertical direction. (b) Our proposed solution is inspired by the flow manipulation techniques used in aircraft design and rocketry. A passive flow deflector is incorporated behind each wheel-thruster combination. This intercepts and redirects the rotor wake, resulting in extended momentum recovery during transition or in transformed configurations.

In this chapter, we translate the aforementioned principles into a novel flow deflec-

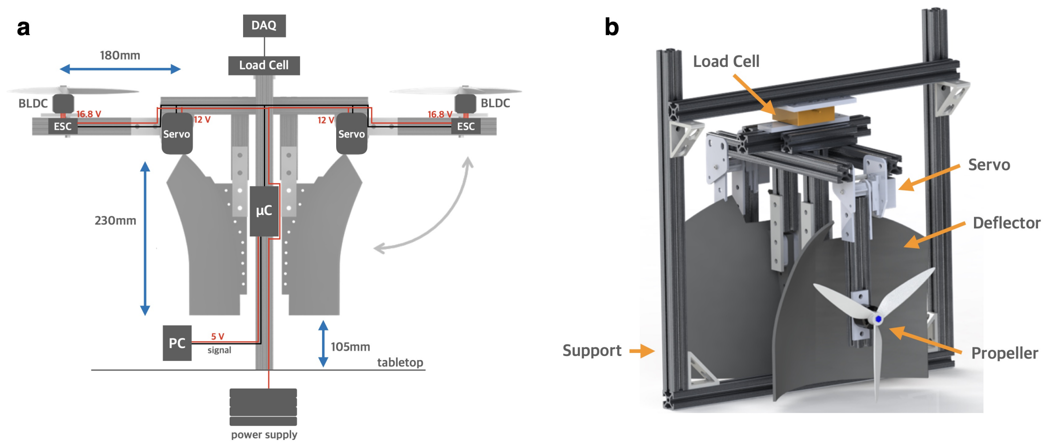


Figure 6.2: Benchtop Experimental Setup. (a) Schematic of benchtop test rig with electronic placement, wiring, and dimensions depicted. The thrusters can rotate by servo motors which are controlled from a central micro-controller labeled (μC). The power supply consists of a 16.8V DC supply that is stepped down to 12V to power the servo motors, and feeds 16.8V to the ESCs which control the rotational speed of two Brushless DC motors (BLDC). A load cell is attached to the top of the test rig and the thrust data is read into a PC using a digital acquisition (DAQ) module. The deflectors are incorporated rigidly onto the main frame. The setup is symmetric to eliminate moment crosstalk interference. (b) 3D model of the experimental setup. The load cell is depicted, and the propellers are attached to rigid bars that rotate due to a servo motor attached at the pivot point. Everything is mounted onto a rigid frame to eliminate vibrations.

tion mechanism that requires no moving parts or electronics and that is embedded directly in the Aerially Transforming Morphobot's (ATMO) chassis, shown in Figure 6.1 (b). As the robot transforms, the deflectors intercept and redirect the rotor wake downward. This mechanism extends momentum recovery in transformed configurations, yielding increased thrust during transitional phases. Our experiments demonstrate a significant vertical thrust augmentation, highlighting the potential of passive fluidic systems in morphing UAV architectures. Remarkably, we found that in extreme aerial configurations where no thrust was available without our deflector design, up to 40% of maximum thrust was recovered, significantly expanding the robot hover capabilities.

6.2 Benchtop Experiments

Benchtop load cell experiments were conducted to evaluate the feasibility of the proposed thrust recovery concept. Inspired by the design of ATMO, a custom test rig was constructed featuring two propellers that direct airflow onto 3D-printed deflector surfaces. The propeller thrust axes can be rotated using two servo motors, replicating the configuration changes that occur during ATMO's flight-to-drive transition. A load cell is integrated in series with the propeller-deflector assembly to measure the vertical thrust force. To minimize sensor cross-talk and balance the moment acting on the load cell, two identical propeller-deflector assemblies were operated at the same rotational speed. The experimental setup is shown in Figure 6.2(a) and (b).

The deflector was designed as a concave surface with two principal curvatures and sufficient depth to turn the flow effectively without inducing separation. Its area was made larger than that of the propeller disk to ensure full capture of the propeller wake. Particle image velocimetry (PIV) studies have shown that the highest momentum in a propeller flow is concentrated in a ring of fluid near the propeller tips [54]. Since the force generated on the deflector is directly proportional to the rate at which momentum is redirected vertically, capturing and turning this high-momentum fluid is key to maximizing thrust recovery.

We first oriented the deflector exit angle parallel to gravity ($\theta = 0°$) and measured the total force at the load cell for ten different propeller tilt angles (φ), ranging from $\varphi = 0°$ (thrust axis vertical) to $\varphi = 90°$ (thrust axis horizontal). The experimental setup is shown in Figure 6.3(a). The resulting thrust, normalized by the thrust at $\varphi = 0°$ (T_0), is plotted as T/T_0 versus φ in Figure 6.3(b).

As expected, the thrust generally decreased with increasing tilt angle as the propeller was oriented away from vertical. When compared to tests without deflectors (also shown in Figure 6.3(b)), the deflector provided markedly better thrust recovery at tilt angles greater than 40°. For reference, we compared our measurements to the theoretical decay $T/T_0 = \cos\varphi$, which represents the loss of vertical thrust component due purely to geometric projection, absent aerodynamic effects.

With $\theta = 0°$, the deflector had negligible influence on thrust for $\varphi < 40°$. However, beyond this angle, the thrust no longer followed the cosine trend, and a substantial recovery was observed. At $\varphi = 90°$, where thrust would otherwise fall to zero, approximately 25% of T_0 was retained due to the deflector. This excess thrust is highlighted in Figure 6.3(c), which plots $\Delta T/T_0$, the recovered thrust beyond the cosine prediction.

To explore the potential for enhanced recovery, we repeated the experiment with the deflector exit angle set to $\theta = 10°$. The angled deflector yielded significantly better performance at high tilt angles. Thrust levels were similar to the $\theta = 0°$ case below $\varphi = 40°$, but consistently higher at larger angles. In particular, at $\varphi = 90°$, the thrust recovery exceeded 40% of T_0. Due to mechanical constraints of the test bench, we were unable to test exit angles greater than $\theta = 10°$.

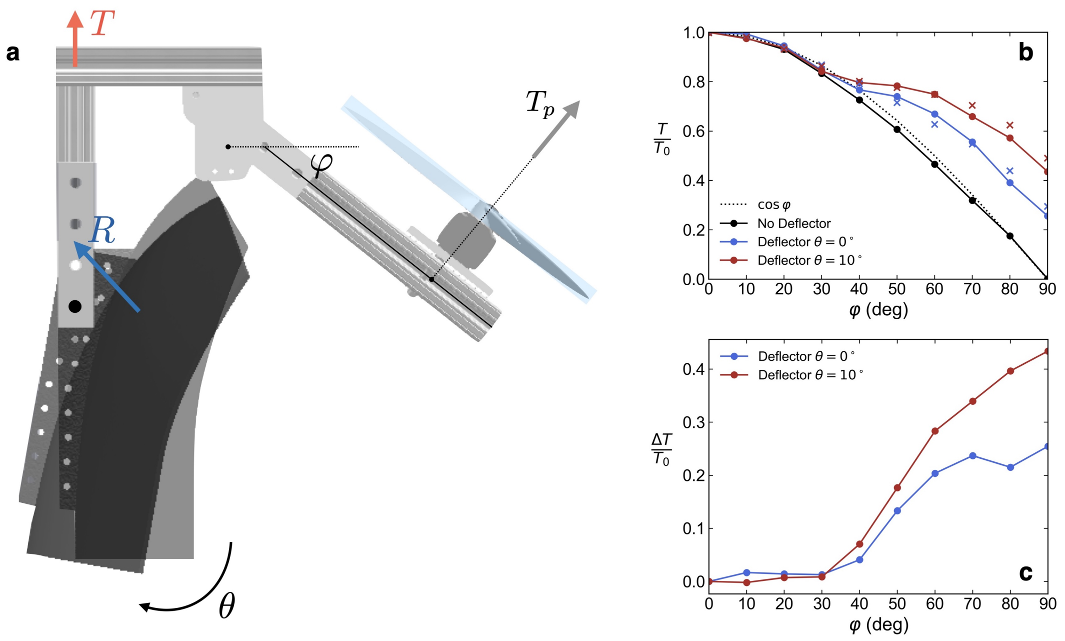


Figure 6.3: Thrust recovery results for benchtop experiment. (a) One half of the benchtop experiment is depicted with key quantities labeled. $\boldsymbol{R}$ is the force acting on the deflector due to the propeller flow, T_p is the thrust produced by the spinning propeller, and T is the overall level of vertical thrust acting on the propeller-deflector assembly. θ is the rotation of the deflector from the baseline configuration where the deflector is pointing vertically down, and φ is the tilt angle of the propeller thrust axis. (b) Thrust values from the experimental setup as a function of φ for two different deflector angles $\theta \in \{0°, 10°\}$ as a ratio of the thrust at $\varphi = 0°$. The thrust with no deflectors in place is plotted in black showing close agreement with the theoretical $\cos\varphi$ decay curve. The result of the numerical flow simulations are overlayed with x markers over the $\theta = 10°$ and $\theta = 20°$ cases, showing reasonable agreement. (c) Thrust recovery as a ratio of the thrust level at $\varphi = 0°$ for two different deflector exit angles.

6.3 Effect of Deflector Angle on Thrust Recovery

To further investigate the effect of deflector exit angle on thrust recovery performance, we conducted numerical simulations replicating the benchtop experimental setup. Using a simplified model of the propeller as an actuator disk [111, 112] we

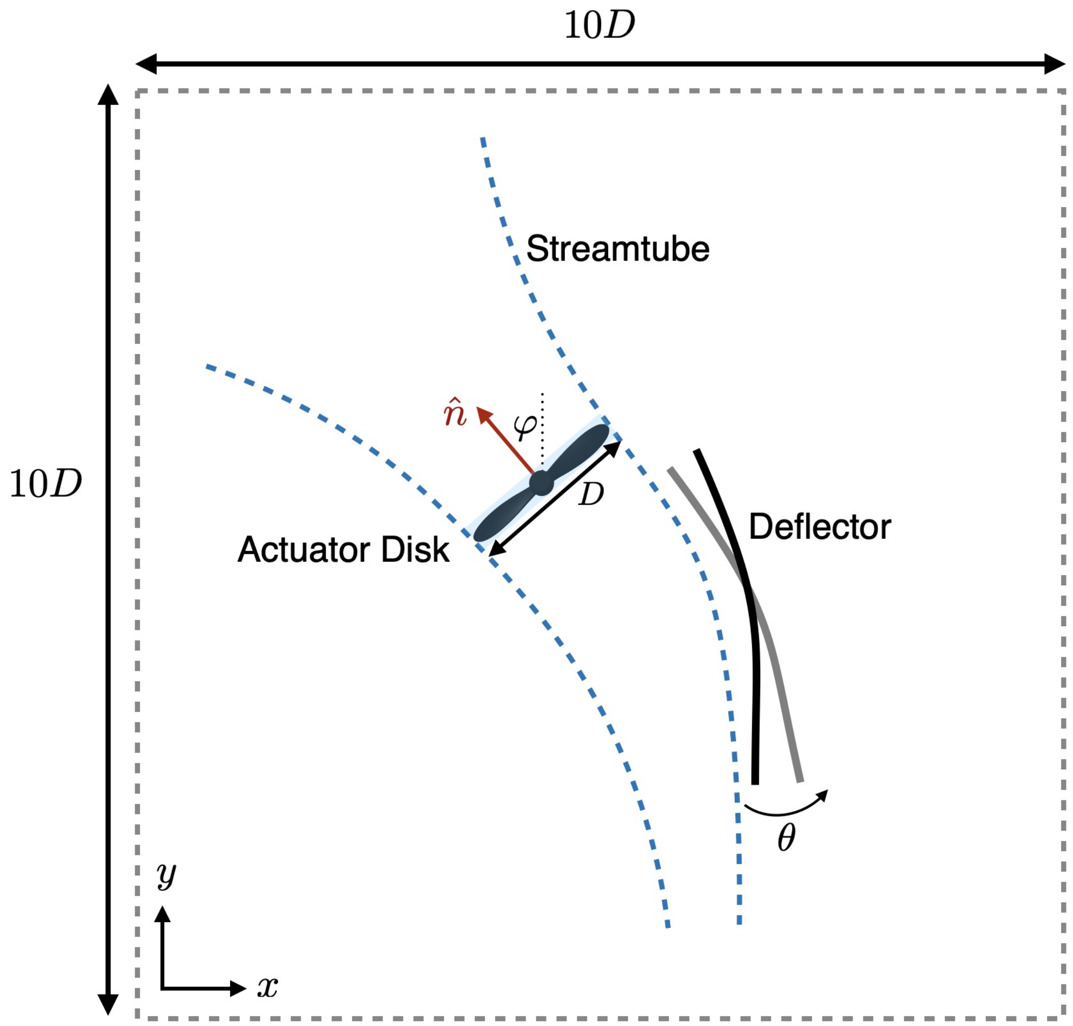


Figure 6.4: Schematic of the computational domain used for numerical flow simulations. A two-dimensional slice of the three-dimensional domain is depicted. The actuator disk is a cell zone of diameter equal to the propeller diameter D that is tilted at an angle φ from the vertical axis. A two-dimensional slice of the deflector and the deflector exit angle θ is depicted.

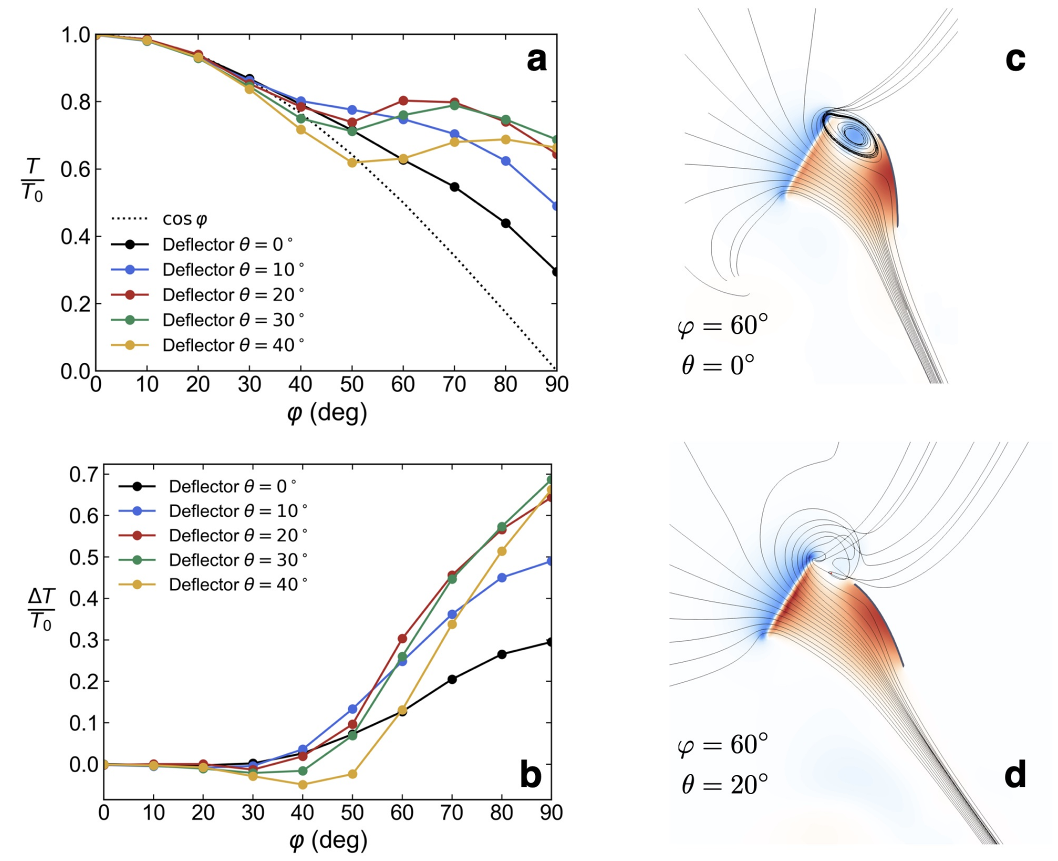


Figure 6.5: (a) Thrust values as a ratio of the thrust at $\varphi = 0^\circ$ are plotted for five different exit angles $\theta \in \{0^\circ, 10^\circ, 20^\circ, 30^\circ, 40^\circ\}$. (b) Thrust recovery values for the same exit angles. (c, d) Representative pressure fields for two cases are shown. The streamlines from the three-dimensional velocity field are projected onto the two-dimensional slice.

performed Reynolds-Averaged Navier-Stokes (RANS) simulations using an open-source finite-volume solver, OpenFOAM [113, 114]. This allowed us to simulate propeller-deflector cases with exit angles past $\theta = 10^\circ$. The computational domain is shown schematically in Figure 6.4 (a). Further details of the numerical solver and setup are provided in the Methods section.

We first validated the numerical simulations against the experimental data from the previous section. Good agreement with the experimental measurements was observed for the $\theta \in \{0^\circ, 10^\circ\}$ cases, as can be seen in Figure 6.3 (b) where the numerical predictions are overlayed on the experimental values. Small discrepancies in thrust levels may be attributed to factors such as discretization errors or the omission of mechanical components of the test rig in the model, which could influence the flow.

We then examined the thrust recovery performance for deflector angles greater than $\theta = 10°$. The results, presented in Figure 6.5 (b) and (c), indicate that thrust recovery generally improved as the deflector angle increased, up to $\theta = 30°$ for high propeller tilt angles ($60° \leq \varphi \leq 90°$). The optimal deflector angle for these high tilt angles lies between $\theta = 20°$ and $\theta = 30°$. At $\theta = 20°$, for example, peak thrust recovery of 0.8 was observed at 60°—a substantial improvement over the baseline case with no deflector tilt. Further increases beyond $\theta = 30°$ did not yield additional benefits; instead, thrust recovery decreased across all tilt angles, suggesting that deflector tilting enhances performance only up to a critical angle, beyond which it becomes detrimental.

To understand the physical mechanisms driving this behavior, we analyzed the pressure fields and representative streamlines of the various deflector configurations. When the deflector was vertical ($\theta = 0°$), part of the flow recirculated near the deflector entrance, creating a low-pressure region that caused a loss of momentum as flow escaped around the deflector. This is illustrated in Figure 6.5(d), which shows the flow field and streamlines for $\varphi = 60°$ and $\theta = 0°$. The low-pressure and recirculation regions are clearly visible in a two-dimensional slice through the center of the deflector and actuator disk. In contrast, at $\theta = 20°$, these recirculation zones disappeared (Figure 6.5(e)), resulting in uniformly high pressure beneath the deflector and more momentum being redirected downward. This produced greater overall thrust and underscores the importance of aerodynamic optimization for maximizing thrust recovery across all propeller tilt angles.

6.4 Implementation of Passive Flow Deflectors on ATMO

We applied the principles from the benchtop experiments and numerical simulations to design deflectors that could be integrated directly into ATMO's chassis. Two design cases were considered. The first was a baseline deflector with no deflector exit angle relative to the vertical axis. This deflector had a width equal to the propeller disk area, aiming to redirect the maximum flow with the smallest possible footprint. Its length was kept smaller than the wheel diameter to maintain sufficient ground clearance, ensuring ATMO could still operate in drive mode when the wheels are fully retracted. The deflector was designed to be as deep as possible without interfering with the onboard robot electronics.

The second design incorporated insights from our earlier experiments and simula-

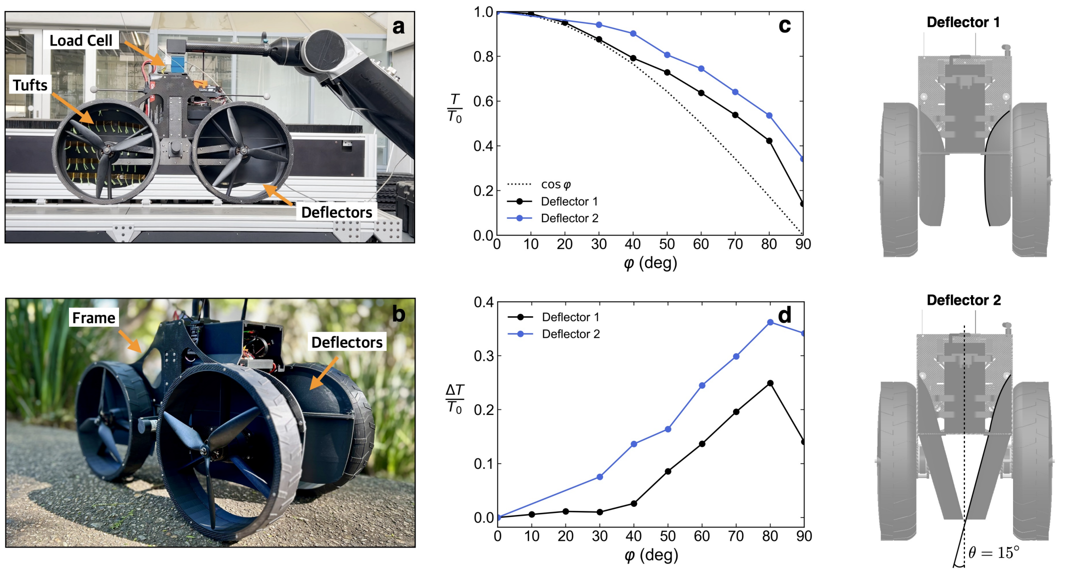


Figure 6.6: Implementation of wake vectoring method on ATMO. (a) Load cell test rig. ATMO is mounted on a robotic arm with a load-cell that measures the vertical force. Tufts have been included on one of the deflectors for some visual feedback of the flow field near the deflector surface. (b) The deflectors implemented on ATMO. (c) Thrust values as a ratio of the thrust at $\varphi = 0°$ for both deflector designs compared to the $\cos\varphi$ decay curve. (d) Thrust recovery values for both deflectors. (right) The two deflector design are shown from the frontal perspective. The Deflector 2 design implements insight from benchtop experiments and numerical flow simulations by maximizing the exit angle of the deflector, achieving superior performance.

tions by adopting the largest feasible exit angle. Due to ATMO's short wheelbase in drive mode, the maximum exit angle achievable was 15°. The deflector was also widened and lengthened beyond the propeller disk area to capture and redirect more of the high-momentum flow near the propeller tips. The two deflector designs are shown in Figure 6.6; Figure 6.6(b) provides a photograph of Deflector 1 integrated onto ATMO in its drive configuration.

Both deflector designs were mounted on ATMO, which was secured to a single-axis load cell as shown in Figure 6.6(a). We measured the total vertical force at ten propeller tilt angles between $\varphi = 0°$ and $\varphi = 90°$. The results, presented in Figure 6.6(c) and (d), show that both designs improved thrust recovery at high tilt angles. As anticipated, Deflector 2 outperformed Deflector 1, achieving a peak thrust recovery just under 40% at $\varphi = 80°$, with only a small decline at $\varphi = 90°$. The use of a 15° exit angle consistently enhanced thrust across all tilt angles, supporting our earlier hypothesis that increasing deflector angle improves thrust recovery efficiency.

Interestingly, Deflector 2 also showed greater thrust recovery at low tilt angles ($\varphi < 40°$) compared to the benchtop results. This may be due to aerodynamic interactions between the propeller flow and the surrounding wheels, or the specific geometry of the deflector on ATMO. These results confirm that passive flow deflectors can be effectively integrated into morphing aerial-ground robots, providing significant aerodynamic benefits without active control mechanisms.

6.5 Hover in Extreme Aerial Configurations

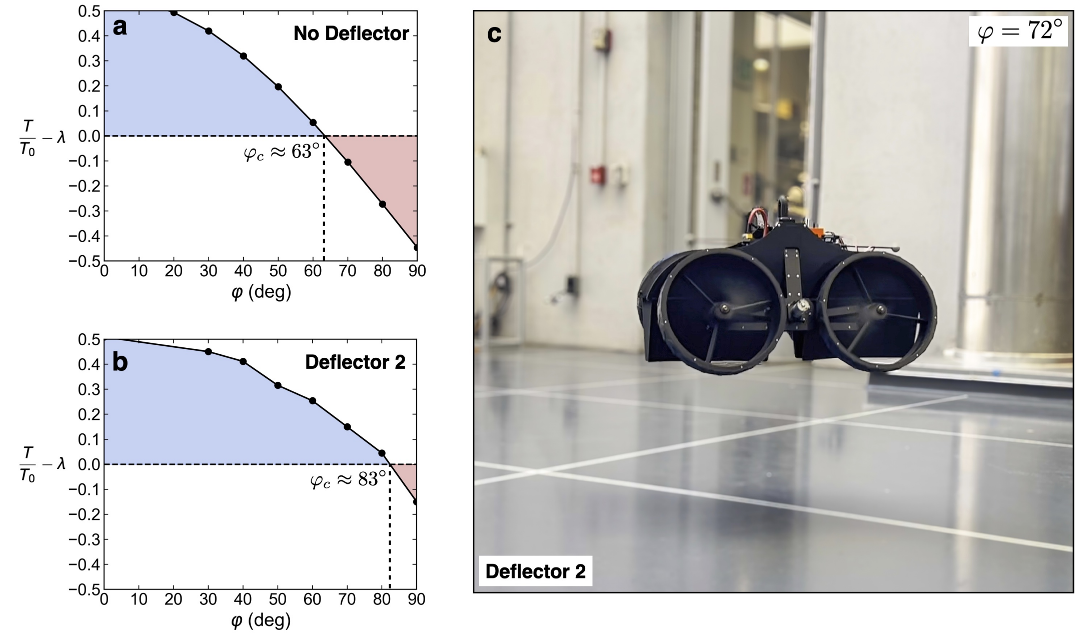


Figure 6.7: ATMO hovering in an extreme aerial configuration. (a) The excess hover thrust, $\frac{T}{T_0} - \lambda$, is plotted as a function of the body tilt angle φ for the case of no deflectors; λ is the reciprocal thrust to weight ratio. Without deflectors the robot weighs 5 kg so $\lambda = 0.45$, where the experimentally measured maximum thrust of 11.2 kg was used. (b) Excess hover thrust with Deflector 2 incorporated on ATMO. Since the deflector adds 0.5 kg of weight, $\lambda = 0.49$. Despite the increased weight, the critical hover angle still increases significantly; $\varphi_c = 83°$. (c) ATMO hovering with $\varphi = 72°$ in an extreme configuration in Caltech's Center for Autonomous Systems and Technology (CAST) flight arena.

The thrust required for hover depends on both the vehicle's weight and its body configuration. At each tilt angle φ the maximum thrust available is given by the thrust recovery curve as $T_{\max}(\varphi) = \frac{T(\varphi)}{T_0} T_{\max}$, where $T(\varphi)/T_0$ is the normalized thrust recovery curve. For a vehicle of mass m, the maximum tilt angle at which hover is possible corresponds to the intersection of this curve with the required hover thrust:

$$\frac{T(\varphi)}{T_0} = \frac{mg}{T_{\max}} \equiv \lambda, \tag{6.1}$$

where λ is the reciprocal of the vehicle's thrust-to-weight ratio. For ATMO without deflectors $\lambda = 0.45$; with thrust deflectors $\lambda = 0.49$.

The solution to Eq. 6.1 is illustrated in Figure 6.6(a) and (b) for the cases without deflectors and with Deflector 2, respectively. Hover is achievable at tilt angles to the left of the intersection between the horizontal line at λ and the thrust recovery curve; the available control authority (excess thrust) is highlighted in blue. Beyond this intersection, thrust is insufficient for hover and control, as indicated in red. For Deflector 2, the critical hover angle is $\varphi_c = 83°$, representing a substantial 20° increase compared to the configuration without deflectors.

We experimentally tested ATMO's ability to hover in extreme configurations equipped with Deflector 2. Hover was successfully achieved at $\varphi = 72°$ while maintaining stability and compensating for disturbances. A snapshot of the experiment is shown in Figure 6.6(c) and in Supplementary Video 1. The robot was able to take off from the drive configuration, sustain hover, and perform controlled hops to new locations. Beyond $\varphi = 72°$, ATMO could initiate takeoff but lacked sufficient control authority to stabilize vertical position against disturbances. This phenomenon, consistent with prior observations for ATMO without deflectors [19], reflects a general limitation of systems operating near actuator saturation.

Remarkably, the same control architecture used for conventional quadrotor flight—with only minor gain tuning—was able to stabilize ATMO even in these extreme aerial configurations. This contrasts sharply with the deflector-free case, where specialized control strategies were required to manage the horizontal thrust components induced by rotor tilting [19]. The deflectors not only augmented vertical thrust but also simplified control by redirecting momentum downward and reducing horizontal force components. Thus, passive flow deflectors provide a dual benefit: enhanced thrust recovery and reduced control complexity.

6.6 Discussion

We have presented an aerodynamics-inspired method for enhancing the thrust efficiency of aerial robots that reconfigure mid-flight. Our findings demonstrate that passive wake vectoring can provide substantial thrust recovery during morphing flight, with up to 40% of vertical thrust regained in configurations where no vertical thrust would be expected without flow manipulation. These results highlight that carefully designed flow-deflection surfaces can mitigate thrust losses associated with mid-flight reconfiguration, reducing reliance on additional actuators or complex control strategies. Compared to prior solutions which used active thrust vectoring through model-predictive control schemes or reinforcement learning [19, 83], passive wake vectoring offers a low-complexity alternative that does not require additional electronics or actuation.

The ability to passively steer rotor wake opens new design opportunities for morphing aerial robots. We successfully implemented the deflector design on ATMO to hover at extreme body angles, indicating that shape-changing aerial robots equipped with passive wake vectoring can achieve more aggressive transformations without sacrificing control authority. This expands the operational envelope, particularly for applications in cluttered environments, where drones must rapidly adapt their morphology to navigate tight spaces or interact with structures. For example, operating ATMO at $\varphi = 72°$ tilt angle reduces its cross-sectional width by 20 cm, enabling entry into narrow gaps where flight or driving can continue.

This study also provides insight into the physical principles governing thrust recovery through passive flow deflection. Key design aspects identified as critical for effective thrust recovery include: (i) ensuring the deflecting surface fully captures the propeller disk area, and (ii) preventing flow separation or recirculation on the deflector surface to maximize momentum redirection. We further developed a numerical simulation framework that can be readily adapted to other case studies. By modeling the propeller as an actuator disk and applying a simple momentum source term to the velocity transport equations, the framework avoids the need for complex rotating meshes while yielding simulation results in reasonable agreement with experimental measurements. The simulation framework is available online (see Code Availability section).

While promising, this approach has limitations. The level of thrust recovery achieved is highly dependent on the geometry of the deflector and the specific morphing configuration. Although the design principles outlined here can inform future designs, the current deflector geometry will not necessarily generalize to other morphing aerial robots without modification.

Future work will explore how passive wake deflection can be integrated at the robot design stage to optimize thrust recovery across a broader range of configurations. Leveraging pre-existing structural elements for wake deflection could minimize the added weight and form drag associated with dedicated deflectors. Finally, augmenting passive wake vectoring with active flow control technologies, such as movable vanes, may further expand the capabilities of morphing aerial robots.

6.7 Methods

Numerical Simulations

For the fluid simulations we used the open-source computational fluid dynamics software OpenFOAM [113, 114] and the algorithm `simpleFOAM` to solve the incompressible, Reynolds Averaged Navier-Stokes equations,

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0$$

$$\rho\, \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = \rho\, \bar{a}_i + \frac{\partial}{\partial x_j}\left[-\bar{p}\,\delta_{ij} + \mu\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) - \rho\,\overline{u_i' u_j'}\right],$$

for $\bar{\boldsymbol{u}}$ and $\bar{p}$ which are the time-averaged velocity and pressure fields respectively. ρ is the density and μ is the viscosity of air at 15° C. The Einstein index notation of summation over repeated indices is implied. $\boldsymbol{a}$ is a body acceleration source that we impose to simulate the propeller flow. The RANS equations are closed by modeling the Reynolds Stresses using Menter's Shear Stress Transport (SST) k-ω model [115, 116].

We used a cubic simulation domain with edge length 2 meters which corresponds to 10 times the propeller diameter ensuring that the boundaries do not affect the flow. The propeller and deflector were placed in the center of the simulation domain with relative distances determined by the benchtop geometry shown in Figure 6.2(a) as well as the angles φ, θ for each case. The mesh consisted of 2.7 million hexahedral cells and was adjusted to the deflector surface using OpenFOAM's built in mesh snapping algorithm `snappyHexMesh`. The mesh was refined around the deflector surface to ensure that the surface was captured accurately and did not contain any unwanted surface imperfections.

To simulate the propeller without needing a rotating mesh or the computational costs of a higher-fidelity simulation, we resorted to a simplified model based on actuator disk theory—one of the simplest and oldest mathematical tools for modeling screw propellers [117]. The key simplification made by this theory is to replace the load on the real propeller by a uniform and normal pressure distribution on an infinitely thin, permeable disc [111, 112]. To implement this in OpenFOAM, we defined a cylindrical zone centered at the propeller of height 10mm and radius equal to the propeller radius. In this cell zone we imposed a source term in the momentum equation of $T_p = 12N$ per cubic meter, simulating the effect of a propeller pushing air through the propeller disk.

The no-slip boundary condition was applied on the deflector surface and a `pressureInletOutletVelocity` boundary condition was applied to the six edges of the cubic domain which adjusts the inflow or outflow depending on the local pressure field, allowing the boundary to switch between inlet and outlet behavior based on the solution during the simulation.

The force on the deflector was computed by integrating the stress tensor $\boldsymbol{\sigma}$ at the deflector surface,

$$\boldsymbol{R} = \int_S \boldsymbol{\sigma} \cdot \boldsymbol{n} \, \mathrm{d}S,$$

where $\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{\tau}$ and $\boldsymbol{\tau}$ is the viscous stress tensor. To simulate the cases where the deflector was tilted, we kept the deflector fixed in the mesh local coordinates and transformed the position of the actuator disk and thrust force applied relative to the deflector using the two-dimensional rotation matrix

$$\boldsymbol{Q}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

For the cases where θ was non zero, the deflector force $\boldsymbol{R}$ was first computed in the transformed coordinates and was transformed back to lab coordinates as follows:

$$T = T_p \cos(\varphi) + \boldsymbol{Q}_z(\theta)\boldsymbol{R} \cdot \boldsymbol{e}_y,$$

where T_p is the thrust (momentum source) imposed on the actuator disk and T is the total vertical foce on the propeller-deflector assembly.

We ran all simulation cases in parallel using 12 CPU cores. Each simulation was terminated when all residuals had dropped below a predefined tolerance and took an average of 15 minutes to run to convergence on a Desktop Computer equipped with an AMD Ryzen 9 7900X processor. The code used is publicly available (See Code Availability section).

Benchtop Experiment

The experimental setup is shown in Figure 6.2(a) and (b). Two counter-rotating propellers were used to minimize the net moment acting on the load cell. The entire assembly was mounted on an adjustable aluminum frame, with the load cell positioned between the frame and the motor–propeller assemblies. Each motor was powered by a dedicated 4S LiPo battery, providing 16.8 V when fully charged. To ensure consistent performance, the batteries were recharged after each test set to compensate for voltage drop over successive runs.

Tests were performed at various rotational speeds and tilt angles, with the angle incremented in 10° steps from 0° (horizontal) to 90° (vertical). The tilt angle was controlled via an Arduino microcontroller, which actuated the servo motors. Motor speed was regulated using APD80F3 ESCs (Advanced Power Drives), with the Cine66 KV1125 brushless motors (T-Motor) driving 9-inch (228 mm) HQ-90503 three-blade propellers.

A 3A60A three-axis load cell (Interface), capable of measuring forces up to 50N per axis, was used to record thrust. Data were acquired at 250Hz using the BlueDAQ software (Interface). Each measurement consisted of setting the propellers to the target RPM and recording the thrust for 15 seconds. The data were analyzed to compute the mean and standard deviation of thrust for each run. To avoid ground effect influences on the measurements, all tests were conducted with the deflectors positioned at least 2m from the nearest surface.

System Integration Load Cell Tests

ATMO, equipped with the deflectors, was mounted onto a robotic arm that positioned the vehicle sufficiently far from the ground to eliminate ground effect. A single-axis load cell was integrated in series between ATMO and the robotic arm to measure vertical thrust. The orientation of ATMO relative to the ground was set in advance using a digital angle gauge and level meter. Power was supplied by a 24V source capable of delivering up to 100A of continuous current. For each deflector design and tilt angle tested, the propellers were operated at 40% of their maximum speed, and thrust was recorded for 15 seconds using the same BlueDAQ software and data acquisition hardware as in the benchtop experiments.

Robot Hardware Description

The Aerially Transforming Morphobot (ATMO) is described in detail in Chapter 2. We provide here a brief summary of key hardware aspects relevant to this work.

The deflectors are 3D-printed using a Bambu 3D printer in PLA, each weighing about 0.1 kg. To integrate them into the chassis, custom carbon fiber plates were designed and CNC-cut, and the deflectors were mounted onto the chassis using screws. The total additional weight due to the deflectors is 0.5 kg. The total thrust produced by each propeller is 2.8 kg. The weight of the robot without deflectors was 5 kg and with deflectors it was 5.5 kg. This results in a thrust to weight ratio of 2.23 without deflectors and 2.04 with deflectors; equivalently $\lambda = 0.45$ without deflectors and $\lambda = 0.49$ with deflectors.

Hover experiments

ATMO, equipped with Deflector 2, was configured at a tilt angle of $\varphi = 72°$. The vehicle was powered by a 2400mAh 6S LiPo battery. A CubeOrange flight controller running PX4 Autopilot software [53] provided onboard control. The body rate controller gains were only slightly adjusted to enhance performance, while the attitude control gains remained at the default manufacturer settings. ATMO was operated in stabilize mode, in which the control inputs correspond to desired roll, pitch, and yaw Euler angles, and the onboard controller stabilizes these through nested attitude and body rate control loops. The control allocation (actuator mixing) was left unmodified.

Code availability

The code used for the numerical simulations is made publicly available here.

Chapter 7

AERIAL POSTURE CONTROL FOR IMPROVED AGILITY

This chapter incorporates material from the following publication:

Ioannis Mandralis et al. “Minimum Time Trajectory Generation for Bounding Flight: Combining Posture Control and Thrust Vectoring”. In: *2023 European Control Conference (ECC)*. 2023, pp. 1–7. DOI: 10.23919/ECC57647.2023.10178360. (published).

This final chapter presents a theoretical investigation into how aerial transformation can enhance speed and agility in trajectory-tracking flight. Taking inspiration from the bounding flight maneuvers of birds—which leverage body posture and aerodynamic changes to achieve superior aerial agility—we explore the benefits of introducing additional degrees of freedom for posture control in aerial robots. Building on prior work that demonstrated successful control of the Aerially Transforming Morphobot (ATMO) during morphing flight and near-ground transformations, we extend the inquiry to ask what might be possible if even greater freedom in aerial posture were available. To frame this exploration, we use a dynamic model of a morphing quadcopter inspired by the Multi-Modal Mobility Morphobot (M4), a robotic platform endowed with multi-degree-of-freedom actuation that enables posture modulation and thrust vectoring beyond what is possible with conventional quadrotors.

The remainder of this chapter is organized around the minimum-time trajectory tracking problem. We first formalize the problem setting and introduce a framework that combines posture control with thrust vectoring. Using this formulation, we solve for time-optimal trajectories that exploit the added degrees of freedom to reduce traversal time and improve agility. We then analyze the resulting trajectories and compare them to those achievable by standard quadrotors, highlighting the gains enabled by posture control. The chapter concludes with a discussion of the broader implications of this theoretical study, emphasizing how aerial transformation may unlock fundamentally new modes of high-performance flight in future robotic systems.

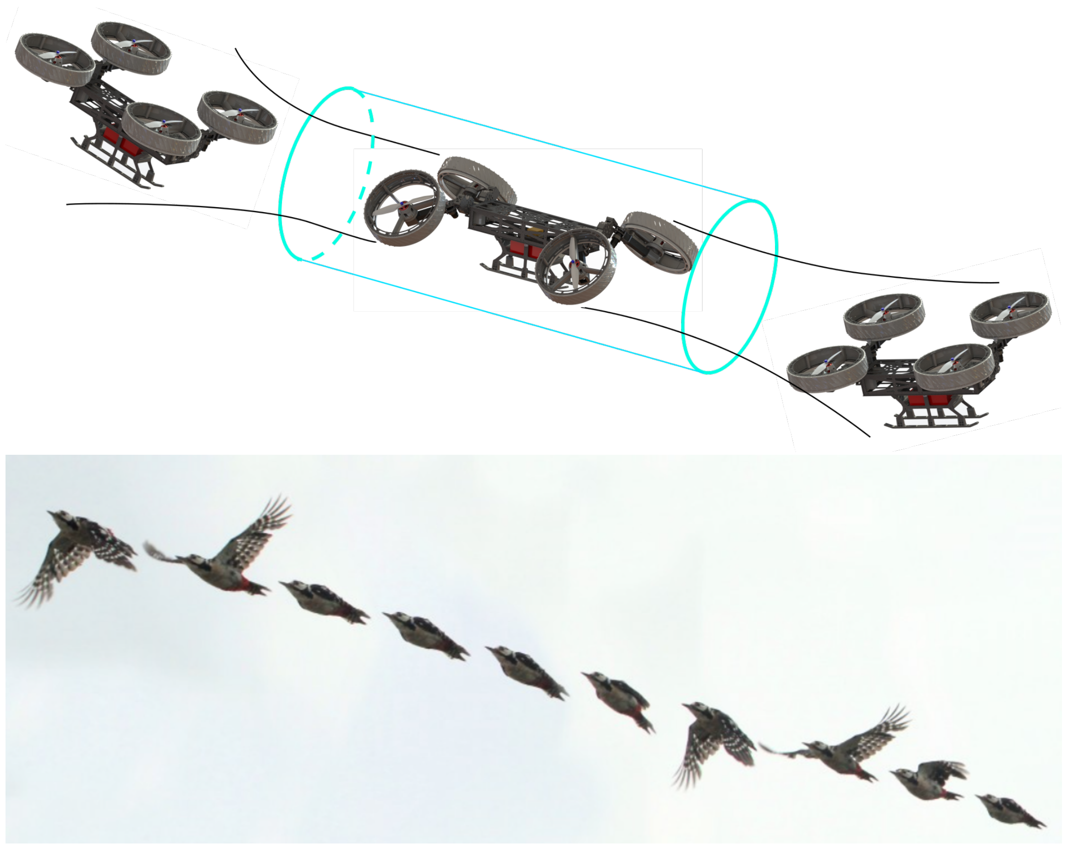


Figure 7.1: Conceptual visualization of the M4 robot performing a bounding flight maneuver compared to the bounding flight maneuver observed in flight by a Great Spotted Woodpecker (image was taken from [122]).

7.1 Introduction

Bounding flight is an intermittent avian flight pattern consisting of bursts of active flapping interlaced with periods of passive flight called "bounds" where the bird tucks its wings under its body to soar [118]. This mode of flight has been hypothesized to increase efficiency by reducing aerodynamic drag [119], increase achievable flight speed at the same level of energetic expenditure [120], and allow birds to transition from stability to instability by dynamically altering roll and yaw inertia [121].

From a dynamical systems point of view, birds performing bounds are modifying their inertial body dynamics and aerodynamic characteristics *simultaneously* to change the global lift and drag (or thrust) vectors. Despite the obvious benefits of coupling aerodynamic and inertial dynamics control (or posture control), this still remains a largely unexplored topic in the aerial robotics community, whose general focus has been on optimizing the performance of fixed geometry quadrotors.

Some notable attempts to produce unmanned aerial vehicles (UAVs) with more general thrust vectoring capabilities include the voliro omnirotational hexacopter [123] which uses variable tilt thrusters that can provide out-of-plane force. However, this is a fixed frame design and, as such, the body inertia cannot be dynamically modified while thrust vectoring. In contrast, examples of systems that are capable of altering their frame geometry include the foldable drone [38], and scissor-like drones which can morph in plane to avoid obstacles [124, 44]. However, these adaptive morphologies are limited to *in-plane* motion of the thrusters. Thus, even though they can morph dynamically while flying (allowing them to pass through narrow gaps), they cannot generate out-of-plane thrust, and their influence on the system inertia tensor is limited.

A notable design that allows for more general thrust vectoring capabilities while simultaneously being able to morph during flight is the DRAGON robot [125], which can adaptively change its shape to achieve complex three-dimensional configurations but is hindered by its complexity, cost, and weight. Similarly, the TiltDrone [32] can move its thrusters out of plane during flight but only has one degree of freedom on the thruster rotations and has a limited range of motion. Finally, there also exist passively morphing quadrotors [36, 37] which use mechanical springs to fold, allowing them to pass through small gaps. However, the passive folding does not allow for general thrust vectoring, thus limiting the applicability of such systems for bounding flight.

In this work we consider a dynamic model of a morphing aerial robot platform, the Multi-Modal Mobility Morphobot (M4) [18], equipped with four thrusters that can be rotated via two joints actuated by servomotors. This system is capable of morphing its body shape both in-plane and out-of-plane—providing it with general thrust vectoring capabilities as well as the ability to modify the inertial body dynamics significantly. The robot was designed with the goal of performing a wide array of tasks ranging from dynamic obstacle avoidance, flight through narrow and spatially varying tunnels, as well as tracking trajectories which would be otherwise infeasible for fixed geometry quadrotors. Inspired by the bounding maneuvers of avian flight, we aim to use this system's capability of morphing while flying to facilitate the optimization of objectives that were previously impossible with fixed quadrotor geometries.

However, generating trajectories that optimize high-level objectives relevant to bounding flight still remains an open challenge. To this end, we formulate bounding flight as an optimal control problem that can be used to solve a wide range of tasks that require morphing while flying. We formulate task-specific cost functions and constraints, and solve the constrained optimization problem using trapezoidal collocation.

To illustrate the proposed method, two bounding flight tasks are considered: the first consists of flying through a spatially varying tunnel without impacting it, and the second consists of tracking a predefined spatial path in minimum time. For the tunnel flying task, we geometrically represent the robot using its bounding sphere and ensure it remains within the variable radius tunnel by writing a state-based constraint. For the minimum time path tracking case we ensure that the robot tracks the desired spatial trajectory within a user-defined tolerance by using a quadratic constraint on the distance to the path. Solving the minimum time trajectory generation problem both with and without posture control indicates that allowing posture control simultaneously with thrust vectoring can lead to significantly lower path tracking times by careful manipulation of the body shape inputs to increase the three-dimensional thrust authority. We also find that combined posture control and thrust vectoring can enable safe flight through narrow and spatially varying tunnel constraints.

Section 7.2 begins with an overview of our robot's hardware as well as its geometric, and dynamic specifications. In Section 7.3 we describe how the dynamical model of the robot is obtained in relation to bounding flight dynamics. Here we use a Lagrangian approach to derive a reduced-order model of the robot which is then used for trajectory generation. In Section 7.4 we formulate the trajectory generation problem as a constrained optimization problem that can be solved numerically using trapezoidal collocation. Finally, in Section 7.6 we present our simulation results and discuss the benefits of using combined posture and thrust control with concluding remarks and future work in Section 7.7.

7.2 Overview of Robotic Platform Considered

The M4 robotic platform [18], which is illustrated in Fig. 7.1, combines joint actuators with thrusters to perform thruster-assisted locomotion. The robot is capable of switching between wheeled, aerial, dynamic 2-contact point (Segway balancing), quadrupedal, and thruster-assisted dynamic locomotion modes of mobility. There are two actuators at the hip joints to articulate the hip movement about the frontal and sagittal axis for the transformation between rover, Segway (mobile inverted pendu-

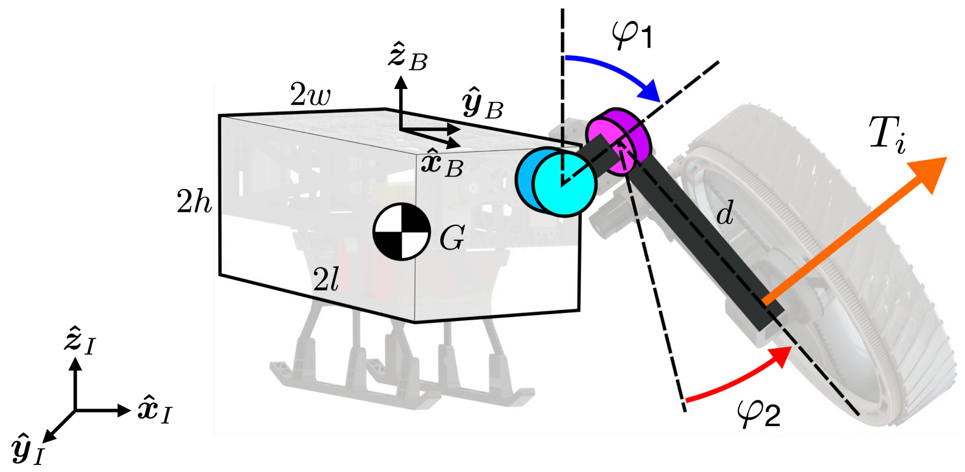


Figure 7.2: Free body diagram of the robot's body articulation. Only one of the arms is depicted for clarity. $\{\hat{\boldsymbol{x}}_I, \hat{\boldsymbol{y}}_I, \hat{\boldsymbol{z}}_I\}$ denotes the inertial frame and $\{\hat{\boldsymbol{x}}_B, \hat{\boldsymbol{y}}_B, \hat{\boldsymbol{z}}_B\}$ denotes the body frame of the robot which is displaced from the inertial frame by position vector $\boldsymbol{p}$ and rotation matrix $\boldsymbol{R}$ (in xyz Euler angles). The physical half-height, half-width, and half-length of the robot body are denoted by h, w, l, respectively, and the arm length is denoted by d. The abductive rotation is denoted by φ_1 and the mediolateral rotation is denoted by φ_2. Finally, the applied thrust at each of the arms is denoted by T_i.

lum, or MIP), and unmanned aerial vehicle (UAV) modes. The ability to transform and perform multi-modal locomotion endows the robot with the resiliency and fault tolerance needed to traverse unstructured environments. In this chapter, we only consider the robot's flight capabilities and leave the transformation and multi-modal locomotion trajectory generation to future work.

The robot weighs 5.6 kg, is approximately 30 cm tall, and is 75 cm wide in UAV mode. The 8-inch propeller and 6S brushless motor combination can generate a maximum thrust force of 2.2 kg, which results in a lift-to-weight ratio of approximately 1.5 and relatively good maneuverability. The hip servo angles can be adjusted mid-flight, which allows the robot to perform the desired bounding flight maneuvers. In this paper we will present the main theoretical framework and demonstrate it in simulation, leaving the implementation on hardware to future work.

7.3 Dynamic Modeling of Bounding Flight

The robot is composed of thirteen rigid bodies (one main body, three linkages per arm) with three joints per arm. In this paper, we derive a simplified model of the robot by neglecting the inertia of the four arms and only allowing abductive rotation (φ_1) and mediolateral rotation (φ_2) of the arms. This leads to a six-degrees-of-freedom model (three translational coordinates and three rotational coordinates) subject to two shape inputs (φ_1 and φ_2) which determine the robot configuration. The robot can control its global thrust vector by setting the force at each of the thrusters independently. The four thruster values are denoted by T_1, T_2, T_3, T_4 and are numbered, in order, from front right, front left, rear right, and rear left. The shape inputs (φ_1 and φ_2), thrust inputs (T_i), and dimensions of the robot (h, w, l, d) can be seen in Fig. 7.2 and are used in our subsequent derivations. Note that we have not allowed each arm to rotate independently, but this can easily be incorporated into future work.

To derive the dynamical equations of motion we consider the robotic system as a set of massed components. For each j-th massed body, m_j denotes the mass, $\hat{I}_j \in \mathbb{R}^{3\times3}$ is the principal inertia matrix, $\boldsymbol{p}_j \in \mathbb{R}^3$, $\boldsymbol{v}_j \in \mathbb{R}^3$ is the inertial position and velocity vectors, respectively, and $\boldsymbol{\omega}_j \in \mathbb{R}^3$ is the angular velocity vector defined in the body frame of the j-th massed component. Furthermore, let $\boldsymbol{g} \in \mathbb{R}^3$ be the gravitational acceleration vector defined in the inertial frame. Then, the Lagrangian $\mathcal{L}$ can be written as the sum of the total kinetic and potential energy in the system:

$$\mathcal{L} = \frac{1}{2}\sum_j \left(m_j \boldsymbol{v}_j^\top \boldsymbol{v}_j + \boldsymbol{\omega}_j^\top \hat{I}_j \boldsymbol{\omega}_j\right) - m_j \boldsymbol{p}_j^\top \boldsymbol{g}, \tag{7.1}$$

where the first two terms are the linear and angular kinetic energy, while the last term is the potential energy.

Let $\boldsymbol{q} \in \mathbb{R}^6$ be the generalized coordinates of the system, which consist of the body's 6 DOF (position and orientation). The orientation of the body frame of the robot relative to the inertial frame is represented by xyz Euler angles: roll (ϕ), pitch (θ), and yaw (ψ). The equation of motion can then be derived using the Euler-Lagrangian formulation as follows:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} = \boldsymbol{u}_g + \sum_k \boldsymbol{u}_{j,k} + \boldsymbol{u}_{w,k} + \boldsymbol{u}_{t,k}, \tag{7.2}$$

where $\boldsymbol{u}_{j,k}$ and $\boldsymbol{u}_{w,k}$ are the generalized joint torques and wheel traction forces acting on the k-th arm, respectively, and $\boldsymbol{u}_{t,k}$ and $\boldsymbol{u}_g$ are the generalized thruster and ground contact forces. The model given by Eq. 7.2 is highly generic. It can be used to model crawling motions, quadrupedal configurations, wheeled motion, as well as aerial locomotion.

Following the above procedure to derive the dynamical model leads to a 12 dimensional state space $x = [\boldsymbol{p}^\top, \boldsymbol{R}^\top, \boldsymbol{v}^\top, \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^{12}$, where $\boldsymbol{p}$ is the position of the robot center of mass in the inertial frame, $\boldsymbol{R}$ is the orientation of the robot relative to the inertial frame (expressed as a vector of xyz Euler angles), $\boldsymbol{v}$ is the absolute velocity of the robot in the inertial frame, and $\boldsymbol{\omega}$ is the angular velocity of the robot expressed in the body frame. Equation 7.2 can hence be written in the following state-space form:

$$\dot{x} = f(x, u), \tag{7.3}$$

where x and u denote the state and input vectors. The input vector $u = [T_1, T_2, T_3, T_4, \varphi_1, \varphi_2]^\top \in \mathbb{R}^6$ embodies thruster inputs and shape inputs (or joint actions) as given in Eq. 7.2. The nonlinear term f embodies all model terms, including gravity, inertial, and Coriolis matrices, shown in Eq. 7.2.

Using this dynamical model, we apply the direct collocation method to resolve the trajectory generation problem for bounding flight through dynamic posture control and thrust vectoring.

7.4 Direct Collocation Nonlinear Dynamic Programming

General Optimization Problem

Trajectory generation for bounding flight is achieved by solving the constrained optimization problem:

$$\begin{array}{lll} \underset{x(\cdot), u(\cdot), t_f}{\mathit{minimize}} & \int_0^{t_f} L(x(t), u(t))dt + V(x(t_f)) & \\ \text{subject to} & \dot{x} = f(x, u), & \forall t \in [0, t_f], \\ & C_j(x, u, t) \leq 0, & j = 1, ..., m, \\ & r_j(x(0), x(t_f), t_f) = 0, & j = 1, ..., k, \end{array} \tag{7.4}$$

where $L(\cdot, \cdot)$ denotes the state and input dependent integral cost, and $V(\cdot)$ denotes the terminal cost. We have also included k equality boundary constraints given by r_j, as well as m inequality path constraints given by C_j which can be used to take into account input saturation, spatial state constraints, etc. Trajectories that solve this optimization problem will be dynamically feasible due to the system dynamics constraint: $\dot{x} = f(x, u)$.

Numerical Solution Using Trapezoidal Collocation

The above optimization problem is solved numerically using the trapezoidal collocation technique. The system dynamics equations are discretized at N equidistant collocation points,

$$0 = t_1 < t_2 < \ldots < t_N = t_f, \tag{7.5}$$

where t_i denotes discrete times. The dynamics are represented as a set of constraints which are derived by integrating $\dot{x} = f(x, u)$ using the trapezoidal quadrature rule:

$$x_{i+1} - x_i = \frac{1}{2}(t_{i+1} - t_i)(f_i + f_{i+1}), \quad i = 1, \ldots, N-1. \tag{7.6}$$

Here, $f_i = f(x_i, u_i)$, and $x_i = x(t_i)$ are the discretized values of the functions on the collocation points.

The collocation constraints (7.6) ensure that the system dynamics are satisfied at the collocation points. The remaining constraints for the nonlinear program constitute the path constraints at t_i, and the boundary constraints at t_1 and $t_f = t_N$. These are given by

$$\begin{aligned} C_j\left(x_i, u_i, t_i\right) &\leq 0 \quad j = 1, \ldots, m \\ r_j\left(x_1, x_N, t_N\right) &= 0 \quad j = 1, \ldots, k, \end{aligned} \tag{7.7}$$

where the path constraint is defined for each collocation point $i = 1, \ldots, N$, and $u_i = u(t_i)$. Combined with the collocation constraints, (7.6) and (7.7) constitute the full constraint set which is denoted by $\mathcal{C}$. Note that the constraints are only enforced at the collocation points. In a practical setting, achieving constraint satisfaction for all times can be achieved via constraint tightening, but this was not explored here.

The optimization problem is resolved using the MATLAB fmincon function combined with the OptimTraj package [80]. To this end, the states and inputs are stacked into the vectors $X = \left[x_1^\top, \ldots, x_N^\top\right]^\top$ and $U = \left[u_1^\top, \ldots, u_N^\top\right]^\top$ which are then concatenated with the final discrete time t_f to form the decision parameter vector, y, for the nonlinear programming problem:

$$y = \left[u_1^\top, \ldots, u_N^\top, x_1^\top, \ldots, x_N^\top, t_f\right]^\top. \tag{7.8}$$

This finally allows us to write the numerical optimization problem, where the cost function has also been approximated using trapezoidal quadrature, as

$$\begin{gathered} \min_y \sum_{i=1}^{N-1} \frac{1}{2}(t_{i+1} - t_i)(L_i + L_{i+1}) + V(x_N) \\ \text{subject to constraints } \mathcal{C}, \end{gathered}$$

where $L_i = L(x_i, u_i)$.

Once the nonlinear program has been solved we approximate the input as a piecewise linear spline. At every sample time, the input is taken to be the linear spline between u_i and u_{i+1} for $t_i \leq t < t_{i+1}$:

$$u_{\text{int}}(t) = u_i + \frac{t - t_i}{t_{i+1} - t_i}\left(u_{i+1} - u_i\right). \tag{7.9}$$

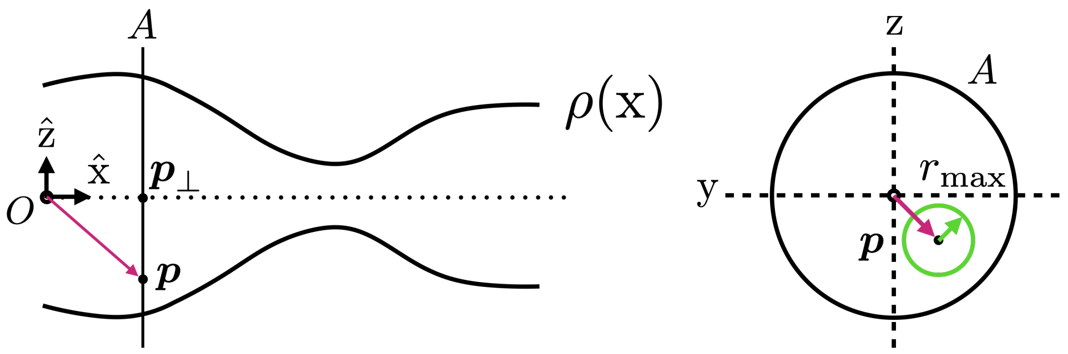


Figure 7.3: Geometry of tunnel constraint. $\boldsymbol{p}$ denotes the absolute position of the geometric center of the robot, $\rho(\mathrm{x})$ denotes the radius of the tunnel constraint as a function of x which starts at the beginning of the tunnel and is oriented along the midline of the tunnel. $r_{\max}$ denotes the maximum dimension of the robot in any given configuration. The goal is that the green circle remains within the larger black circle at all times during the trajectory.

$u_{\text{int}}(t)$ is then fed directly into the dynamical system model, $\dot{x} = f(x(t), u_{\text{int}}(t))$, which is integrated forward in time using an fourth order Runge-Kutta scheme to obtain the final state evolution trajectory $x(t)$ on a finer mesh than originally found via collocation.

7.5 Spatially Varying Tunnel Constraint

In this section, we describe the constraint developed to deal with a spatially varying tunnel geometry that the robot is tasked with flying through.

We can in general describe a tunnel constraint by a spatially varying radius $\rho(s)$ and a midline trajectory, where s is the curvilinear coordinate along the midline. In this chapter, we restrict the problem setup by assuming that the midline of the tunnel constraint is a straight line. Thus $\rho(s) \equiv \rho(\mathrm{x})$, and this radius, along with an initial position, uniquely specifies the tunnel constraint. The geometry considered is depicted in Figure 7.3. Note that x, y, z denote the spatial coordinates, and x which is the spatial x coordinate is not to be confused with x which is the system state.

To ensure that the robot remains within the tunnel we formulate the following path constraint:

$$||\boldsymbol{p} - \boldsymbol{p}_\perp||_2 - \rho(\mathrm{x}) + r_{\max}(\varphi_1, \varphi_2) \leq 0, \tag{7.10}$$

where $\boldsymbol{p}$ denotes the position of the center of the robot in the inertial frame, and $\boldsymbol{p}_\perp$ denotes the position of the midline of the tunnel constraint which is closest to the robot. The largest dimension of the robot is denoted by $r_{\max}(\varphi_1, \varphi_2)$ and computed as follows:

$$r_{\max}(\varphi_1, \varphi_2) = \max(r_{\mathrm{x}}, r_{\mathrm{y}}, r_{\mathrm{z}}), \tag{7.11}$$

where

$$\begin{aligned} r_{\mathrm{x}} &= l + d\sin(\varphi_2)\cos(\varphi_1) \\ r_{\mathrm{y}} &= w + d\cos(\varphi_2)\cos(\varphi_1) \\ r_{\mathrm{z}} &= -h + d\sin(\varphi_1). \end{aligned}$$

Here, the robot has been bounded by a sphere with a radius of the largest dimension of the robot in its current configuration. For a definition of the dimensions and the angles see Figure 7.2. We further assumed that $\frac{\partial \rho}{\partial \mathrm{x}}$ does not vary fast enough for the out-of-plane dimensions to play a role in the constraint. Note that a constraint of similar complexity can be formulated by finding an ellipsoidal approximation of the robot, but this is left for future work.

7.6 Results

To demonstrate the proposed method we consider two scenarios: minimum time flying through a spatially varying tunnel, and minimum time path tracking. In both scenarios the cost function was chosen to minimize the total time of the maneuver, i.e. $L(x, u) = 1$, $V(x) = 0$, and we used input saturation constraints of the form

$$0 \leq T_i \leq 22 \text{ N}, \quad i = 1, \ldots, 4,$$

as well as constraints on the shape variables of the form

$$0 \leq \varphi_i \leq \frac{\pi}{2}, \quad i = 1, 2.$$

where φ_i is in radians. For both the tunnel flying and path tracking case the start and end positions were fixed and enforced via boundary constraints:

$$\begin{aligned} \boldsymbol{p}_0 &= (0, 0, 0.5)^\top \text{ m} \\ \boldsymbol{p}_f &= (5, 0, 0.5)^\top \text{ m}. \end{aligned}$$

The final time t_f is an optimization variable and was hence left unspecified.

Flying Through a Tunnel in Minimum Time

As a first example, we attempt a maneuver that consists of flying 5 meters forwards through a spatially varying tunnel while maintaining a height of 0.5 meters above

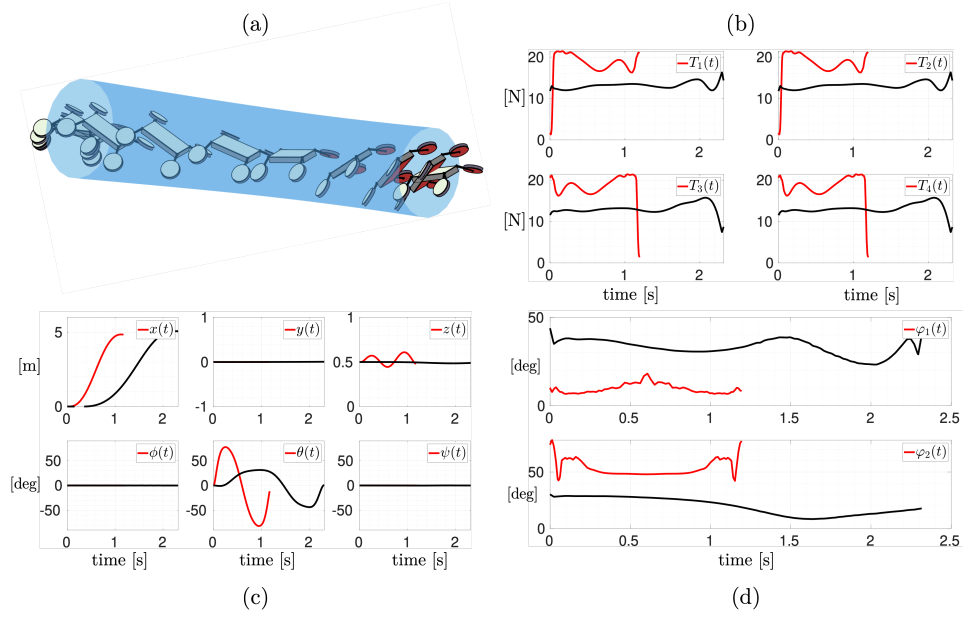


Figure 7.4: Comparison of the optimal minimum time trajectories with and without the tunnel path constraint. (a) Visualization of snapshots of the simulated trajectory in 3D with the path constraint. Snapshots are taken at evenly spaced time intervals. (b) Time evolution of the thrust inputs for a trajectory with and without path constraints. (c) Time evolution of states for a trajectory with and without path constraints. (d) Time evolution of shape inputs for a trajectory with and without path constraints. For figures (b), (c), and (d) the trajectory without path constraint is in red, and the trajectory with path constraint is in black.

the ground. Since the tunnel has a contracting geometry, the robot has to use posture control combined with thrust vectoring to fold its body as it flies. The functional form of the tunnel chosen for this example was a shifted Gaussian bump, i.e.,

$$\rho(\mathrm{x}) = \rho_0 - \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(\mathrm{x}-\mathrm{x}_c)^2}{2\sigma^2}\right], \tag{7.12}$$

where σ is a length scale and x_c is the center of the Gaussian bump. We set $\sigma = 2$ and $\mathrm{x}_c = 3.5$ meters. At its minimum radius, this tunnel is smaller than the robot in its nominal configuration. This ensures that flying through the tunnel is impossible without using posture control.

We solved the numerical optimization problem twice, using $N = 100$ collocation points, and optimizing the nonlinear program until convergence: once without the path constraint and once with the path constraint active. The evolution of the optimal thrust inputs, states, as well as shape inputs for both trajectories are given in Figure 7.4 (b), (c), and (d), respectively. Snapshots of the robot performing the trajectory with the path constraint are shown in Figure 7.4 (a).

As seen in Figure 7.4(a), the robot folds its thrusters inwards to satisfy the imposed tunnel constraint for all time, thus demonstrating the benefit of combining posture control with thrust vectoring. From Figures 7.4(c) and (d) we can see that there is a tradeoff between satisfying the tunnel constraint and performing the maneuver in minimum time. In particular, the trajectory that satisfies the tunnel constraint takes approximately $t_f \approx 2.3$ seconds whilst the trajectory that does not satisfy the constraint takes approximately $t_f \approx 1.2$ seconds. The main difference in strategy between the two trajectories is that, when satisfying the path constraint, there is a much smaller variation in the vertical coordinate $\mathrm{z}(t)$ as well as the pitch angle $\theta(t)$. Combined with a much larger abductive rotation $\varphi_1(t)$ and smaller mediolateral rotation $\varphi_2(t)$, which minimizes the body dimensions, this leads to the satisfaction of the tunnel constraint at the expense of a longer maneuver time.

Minimum Time Path Tracking via Posture Control

Combining posture control and thrust vectoring can also be beneficial for the problem of tracking a spatial trajectory in minimum time. We focus on tracking the spatial trajectory,

$$\boldsymbol{p}_d(\mathrm{x},\mathrm{y},\mathrm{z}) = \begin{bmatrix} \mathrm{x} \\ -\frac{\mathrm{x}(\mathrm{x}-L)}{H} \\ \mathrm{z}_0 - \frac{\mathrm{x}(\mathrm{x}-L)}{H} \end{bmatrix}, \tag{7.13}$$

which is a quadratic curve in y and z as a function of x. Note that the desired spatial trajectory $\boldsymbol{p}_d(\mathrm{x},\mathrm{y},\mathrm{z})$ is specified as a function of the full position $\boldsymbol{p} = (\mathrm{x},\mathrm{y},\mathrm{z})$ to allow for general paths to be specified in space. Here, L is the total length of the trajectory in the x-direction, H is a scale that controls how far the robot should move in both the y and z directions and z_0 is the height that the robot starts above the ground. With $L = 5$ m, $H = 6.25$ m, and $\mathrm{z}_0 = 0.5$ m which corresponds to an initial position $\boldsymbol{p}_0 = (0, 0, 0.5)^\top$, this trajectory consists of moving 5 meters forward in x, 1 meter up in z and 1 meter left in y before reaching the final position $\boldsymbol{p}_f = (5, 0, 0.5)$ meters.

To ensure that the geometric center of the robot tracks the desired path $\boldsymbol{p}_d$ we introduced the following path constraint:

$$[\boldsymbol{p} - \boldsymbol{p}_d(\boldsymbol{p})]^\top [\boldsymbol{p} - \boldsymbol{p}_d(\boldsymbol{p})] \leq \tau^2, \tag{7.14}$$

where τ is a user-specified tolerance. Under this optimization scheme, the robot

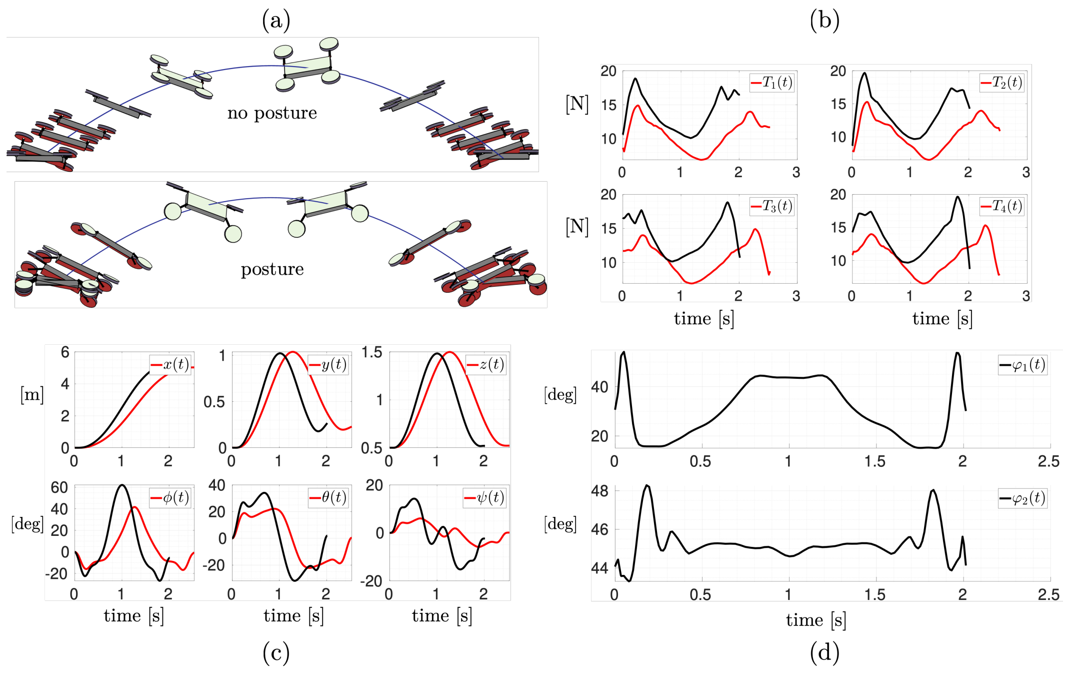


Figure 7.5: Minimum time path tracking with and without posture control. (a) Visualization of snapshots of the simulated trajectory in 3D. Snapshots are taken at evenly spaced time intervals. (top) Trajectory without posture control - shape inputs are fixed at $\varphi_1 = 0, \varphi_2 = 0$. (bottom) Trajectory with posture control - shape inputs are controlled. This is a side view, there is an out-of-plane component of the trajectory. (b) Time evolution of thrust inputs for a trajectory with and without posture control. (c) Time evolution of states for a trajectory with and without posture control. (d) Time evolution of shape inputs for a trajectory with posture control. For figures (b), (c) and (d) the trajectory without posture control is in red, and the trajectory with posture control is in black.

is guaranteed to track the desired trajectory within the tolerance at each of the collocation points. For this example, the tolerance is set to $\tau = 10^{-2}$ m.

The collocation problem was solved twice using $N = 120$ collocation points - once allowing full control over the shape variables φ_1, φ_2 and once by fixing $\varphi_1 = 0$ and $\varphi_2 = 0$ (corresponding to the nominal configuration of the robot). We found that the time required to track the desired trajectory was 20% lower when using posture control ($t_f = 2.0$ s), compared to without posture control ($t_f = 2.5$ s).

The thrust inputs, state evolution, as well as shape variables both for the active posture control and no posture control case, are displayed in Figure 7.5(b), (c), and (d), respectively. The converged trajectories are significantly different. In particular, the posture control trajectory has a significantly higher peak roll angle $\phi(t)$ and a high angle of abduction $\varphi_1(t)$ at around 1 second. This strategy possibly serves to increase the thrust authority at the apex of the trajectory. Indeed, in the case of no posture control, the peak roll angle that can occur during the trajectory is limited since, if the robot were to roll too far, there would not be enough thrust authority in the vertical z-direction to sustain the robot in the air. However, by morphing mid-flight ($\varphi > 0$) thrust authority in the vertical direction is maintained while still having thrust to track the out-of-plane y-trajectory. Striking this balance is possible only through combined posture control and thrust vectoring, and seems to be a determining factor that reduces overall maneuver time.

This maneuver is also shown via snapshots of the simulation in 7.5(a). Here it can be seen that, compared to the no posture control case, the robot tucks its arms mid-flight, increasing thrust authority and achieving a lower time of flight.

7.7 Summary

We have described and implemented an approach for generating minimum time trajectories for bounding flight under spatially varying tunnel constraints. Our results indicate that the interplay between posture control and thrust vectoring can be exploited to decrease maneuver time by simultaneous manipulation of shape and thrust inputs. The main takeaway is that the out-of-plane motion of the thrusters is important for increasing the three-dimensional thrust vectoring capabilities of the robotic system—enabling a wider array of challenging trajectories to be tracked. Furthermore, the ability to morph the robot body during flight allows flight through narrow and spatially varying geometries.

In future work, we plan to incorporate the inertial dynamics of the arms as well as an aerodynamics model into the system dynamics. This will open up further possibilities for the manipulation of inertial body dynamics and aerodynamic control such as drag reduction, soaring, etc. Future work should also aim to implement this research direction on hardware to deepen our understanding of how posture-induced inertial and aerodynamic couplings translate into measurable gains in agility and energy efficiency under realistic constraints.

Chapter 8

DISCUSSION AND OUTLOOK

In this thesis, we developed methods to leverage aerial transformation for enhanced air–ground robotic mobility. Motivated by the energetic advantages of wheeled locomotion over single-mode flight and by the limitations of Morphobots that require on-ground reconfiguration to switch modes, we developed the Aerially Transforming Morphobot (ATMO) and demonstrated that mid-air transformation improves both agility and mode-switching reliability. Our contributions span design, aerodynamics, and control. This chapter synthesizes our principal findings, situates them in a broader context, and concludes with limitations and avenues for future work.

8.1 Summary of findings

We first developed a specialized robot, ATMO, that contends with mid-air thrust forces to transform while flying using a unified structural and actuation system. ATMO distinguishes itself from other flying–driving robots by virtue of a self-locking tilt actuator mechanism that enables mid-air transformation with minimal actuation requirements, lower cost, and a simpler overall design. Although other morphing quadrotor designs have been employed for fitting through narrow gaps [38, 124, 44, 36, 37], or for achieving full actuation or manipulation capabilities [32, 126, 127], few works have used mid-air transformation as a means to enhance ground–aerial locomotion performance [128].

We tested the aerodynamics of the near-ground morpho-transition phase and found that the flow regime of four tilted, interacting rotors approaching the ground differs significantly from that of non-transforming, vertically descending quadrotors. We showed that the ground-effect relation persisted for tilted rotors up to and including the $\varphi = 60^\circ$ case, and then reversed due to fluid-dynamic instability. In Chapter 4, we exploited the ground effect to achieve landings past the critical actuator-saturation angle; formally understanding and utilizing the complex aerodynamic interactions of morphing flight is an effective way to increase the agility of transforming ground–aerial robotic systems.

Secondly, we developed a model-based control scheme that accounts for the full operational envelope of bi-modal ground-aerial locomotion. To tackle the actuator-saturation problem that occurs as the robot tilts its thrusters to land on its wheels, we decomposed the control objective into a convex combination of specialized objectives for each locomotion mode, offering a flexible framework for controlling mid-air transforming robotic systems during ground–aerial transition. Since actuator constraints play a major role for ATMO, we opted to use a model predictive controller which allows us to seamlessly incorporate limits such as thrust bounds. While it is possible to use other methods such as gain scheduling combined with linear–quadratic control, these methods require significant engineering effort and offer less interpretability. We showed that the developed controller enables landings with tilt angles beyond actuator-saturation limits—enabling ATMO to clear large debris and negotiate rough terrain at the landing site.

In parallel, we demonstrated successful transfer of an end-to-end deep RL controller to hardware for the morpho-transition maneuver. To achieve this, we developed an RL method that we compared to the MPC algorithm developed previously. We found that end-to-end RL trained on a well-informed distribution of disturbance dynamics can reject small to moderately sized disturbances more reliably than an equivalent MPC approach. The MPC method performed worse for small disturbances but was able to recover from large disturbances where the RL controller fails. Moreover, the RL method can recover from partial actuator failure without explicit knowledge of the failure. The MPC method can, in principle, be extended to achieve this but would require explicit estimation of the actuator failure to trigger online changes in the optimization. This highlights the potential of RL to generate control policies that generalize from partial sensor information to produce sophisticated control behaviors.

Equipped with the robotic platform and control methods necessary to perform agile aerial maneuvers, we asked whether it was possible to extend these capabilities using aerodynamic insight. To this end, we revisited ATMO's design and introduced a passive wake-vectoring mechanism that recovers lost thrust during morphing. Using internal deflectors that intercept and redirect the rotor wake vertically downward, we recovered up to 40% of thrust in configurations where no useful thrust would otherwise be produced. This substantially extended hover and maneuvering capabilities during transformation. We also showed that using wake-vectoring surfaces alleviated the fundamental coupling between position and attitude dynamics revealed in Chapter 3. This enabled stabilization of ATMO in a morphed configuration using an off-the-shelf cascaded quadcopter control architecture. Finally, we showed that the additional thrust recovered via wake vectoring enabled ATMO to hover with tilt angles larger than the critical angle, substantially extending the robot's operational envelope. Our findings highlight a new direction for morphing aerial-robot design, where passive aerodynamic structures—inspired by thrust vectoring in rockets and aircraft—enable efficient, agile flight without added mechanical complexity.

8.2 Directions for Future Work

Simultaneous Thrust-Vectoring and Posture Control

In Chapter 7 we showed that simultaneously regulating posture and thrust vectoring can increase the agility of aerial robots. Exploiting body-shape change to influence the control-effectiveness matrix as well as the aerodynamic and inertial properties of the system has the potential to markedly improve performance in time-critical maneuvers. A promising direction is to extend the model predictive controller of Chapter 4 with a more general model of dynamic morphing flight that admits additional posture-control inputs. Validating the magnitude of the benefit on suitable hardware, such as the M4 robot [18], would provide experimental evidence for the gains achievable when combining posture and thrust control. The same idea naturally extends to fault tolerance (e.g., recovering from a failed actuator), where mid-air posture adaptation can compensate for loss of authority on a subset of actuators [43].

Maximizing Thrust Recapture with Passive Deflecting Surfaces

In Chapter 6 we showed that varying the deflector exit angle substantially influences the amount of thrust recaptured across tilt angles, underscoring the central role of deflector geometry in thrust recovery. Two questions naturally follow: (i) what is the theoretical upper bound on thrust that can be recaptured using a passive flow deflector; and (ii) how closely can practical designs approach this bound through shape optimization, for example via adjoint methods [129]. A complementary direction is to integrate deflecting surfaces into the chassis itself, yielding weight reductions and, in turn, improved morphing-flight performance.

Sim-to-Real Transfer: MPC versus RL

In our comparison, the MPC controller transferred from simulation to hardware without explicitly modeling sensing–actuation delays or motor dynamics, whereas the RL policy required these effects to be represented during training to achieve successful transfer. A plausible explanation is that MPC's inherent *feedback structure* can tolerate moderate unmodeled latencies and actuator lags better than a policy trained on a—perhaps incomplete—representation of the dynamics. The gap between the simulation and real environment may result in a brittle mapping from observed state to action that is sensitive to timing and actuation model mismatch, unless those are sufficiently well represented during training. Another possibility is that the RL controller "over-fits" and exploits the lack of delay in the model, which works fine in simulation but does not transfer to the hardware. Knowing the true underlying cause of this discrepancy is challenging and requires further investigation. There are several avenues for study: (i) a systematic study of when and how observation and actuator delay "break" each method; or (ii) hybrid approaches that combine MPC with learning, such as the Actor-Critic Model Predictive Control method which uses RL to change cost parameters of an MPC controller running online [75].

Posture-Dependent Ground-Effect Aerodynamics

While the literature on quadrotor ground effect is extensive and analytical models have matured [57], models for platforms with tilted thrusters approaching the ground remain nascent [130, 131], particularly when rotor–rotor interactions are present. Our findings suggest opportunities for new behaviors such as energetically efficient perching or driving up steep inclines using aerodynamic proximity effects [41, 40], now reconsidered through the lens of intentional tilt [132]. Developing predictive, posture-aware aerodynamic models could unlock reliable planning and control strategies that actively leverage ground-effect phenomena during transformation.

Aerial Transformation in Realistic Environments

Although dynamic transition maneuvers were demonstrated experimentally, the conditions were intentionally controlled to accelerate development. A key simplification was reliance on a motion-capture system for high-rate, high-accuracy state estimation that exceeds what is currently achievable with onboard sensors such as GNSS and inertial measurement units. Future work should assess how the presented maneuvers translate to real-world settings featuring unstructured terrain and decision-making under partial, noisy observations. Immediate next steps include (i) integrating a vision-based module that autonomously selects feasible landing sites and body configurations, and (ii) augmenting ATMO's onboard sensing to support outdoor transitions without external infrastructure. With these additions, mid-air transforming platforms like ATMO could offer practical assistance in scenarios where transforming before touch-down or take-off both protects mission-critical hardware from hostile terrain and enhances overall agility.

BIBLIOGRAPHY

[1] Samuel P. Langley and Charles M. Manly. *Langley Memoir on Mechanical Flight*. Smithsonian Institution, 1911. URL: https://www.gutenberg.org/ebooks/47981.

[2] Eric Feron and Eric N. Johnson. "Aerial Robotics". In: *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008, pp. 1009–1029. ISBN: 9783540303015. DOI: 10.1007/978-3-540-30301-5_45. URL: http://dx.doi.org/10.1007/978-3-540-30301-5_45.

[3] J. Gordon Leishman. *The Bréguet–Richet Quad-Rotor Helicopter of 1907*. AHS technical historical note. Historical overview of the No. 1 Gyroplane (1907). 2001. URL: https://www.academia.edu/815361/The_breguet_richet_quad_rotor_helicopter_of_1907.

[4] *Kettering Aerial Torpedo "Bug"*. National Museum of the U.S. Air Force Fact Sheet. 2015. URL: https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/198095/kettering-aerial-torpedo-bug/.

[5] Akira Sato. *The RMAX Helicopter UAV*. Tech. rep. Yamaha Motor Co., Ltd., 2003. URL: https://www.semanticscholar.org/paper/The-RMAX-Helicopter-UAV-Sato/5d80faae7d1ffd27422df3ad6e3d08dc6bdb1920.

[6] Stephen Cass. *The Consumer Electronics Hall of Fame: DJI Phantom Drone*. IEEE Spectrum. On the Phantom's 2013 impact as a ready-to-fly consumer quadcopter. 2018. URL: https://spectrum.ieee.org/the-consumer-electronics-hall-of-fame-dji-phantom-drone.

[7] Ismael Colomina and Pere Molina. "Unmanned Aerial Systems for Photogrammetry and Remote Sensing: A Review". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 92 (2014), pp. 79–97. DOI: 10.1016/j.isprsjprs.2014.02.013.

[8] J. (Bob) Balaram, MiMi Aung, Matthew P. Golombek, et al. "The Ingenuity Helicopter on the Perseverance Rover". In: *Space Science Reviews* 217.56 (2021). DOI: 10.1007/s11214-021-00815-w.

[9] Dario Floreano and Robert J. Wood. "Science, technology and the future of small autonomous drones". In: *Nature* 521.7553 (May 2015), pp. 460–466. ISSN: 1476-4687. DOI: 10.1038/nature14542. URL: http://dx.doi.org/10.1038/nature14542.

[10] Jane Pauline Ramirez and Salua Hamaza. "Multimodal Locomotion: Next Generation Aerial–Terrestrial Mobile Robotics". In: *Advanced Intelligent*

Systems (Dec. 2023). ISSN: 2640-4567. DOI: 10.1002/aisy.202300327. URL: http://dx.doi.org/10.1002/aisy.202300327.

[11] Tony H. Grubesic, Jake R. Nelson, and Ran Wei. “Drones and Their Future Applications”. In: *UAVs for Spatial Modelling and Urban Informatics*. Springer International Publishing, 2024, pp. 149–167. ISBN: 9783031541148. DOI: 10.1007/978-3-031-54114-8_9. URL: http://dx.doi.org/10.1007/978-3-031-54114-8_9.

[12] Clément Lemardelé, Miquel Estrada, Laia Pagès, and Mónika Bachofner. “Potentialities of drones and ground autonomous delivery devices for last-mile logistics”. In: *Transportation Research Part E: Logistics and Transportation Review* 149 (May 2021), p. 102325. ISSN: 1366-5545. DOI: 10.1016/j.tre.2021.102325. URL: http://dx.doi.org/10.1016/j.tre.2021.102325.

[13] Wolduamlak Ayele, Caleb McEntire, Lorne Greene, Bhanu Prakash, Erik Farley-Talamantes, and Victor Maldonado. “Conceptual Design of a Robotic Ground-Aerial Vehicle for Mars Planetary Exploration”. In: *AIAA AVIATION 2022 Forum*. American Institute of Aeronautics and Astronautics, June 2022. DOI: 10.2514/6.2022-3285. URL: http://dx.doi.org/10.2514/6.2022-3285.

[14] Larry Young, Greg Pisanich, and Corey Ippolito. “Aerial Explorers”. In: *43rd AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, Jan. 2005. DOI: 10.2514/6.2005-912. URL: http://dx.doi.org/10.2514/6.2005-912.

[15] G. Gabrielli and Theodore von Kármán. “What price speed ? Specific power requirede for propulsion of vehicles”. In: *Mech Eng.* 72 (1950), pp. 775–781.

[16] Arash Kalantari and Matthew Spenko. “Modeling and Performance Assessment of the HyTAQ, a Hybrid Terrestrial/Aerial Quadrotor”. In: *IEEE Transactions on Robotics* 30.5 (2014), pp. 1278–1285. DOI: 10.1109/TRO.2014.2337555.

[17] Arash Kalantari, Thomas Touma, Leon Kim, Rianna Jitosho, Kyle Strickland, Brett T. Lopez, and Ali-Akbar Agha-Mohammadi. “Drivocopter: A concept Hybrid Aerial/Ground vehicle for long-endurance mobility”. In: *2020 IEEE Aerospace Conference*. 2020, pp. 1–10. DOI: 10.1109/AERO47225.2020.9172782.

[18] Eric Sihite, Arash Kalantari, Reza Nemovi, Alireza Ramezani, and Morteza Gharib. “Multi-Modal Mobility Morphobot (M4) with appendage repurposing for locomotion plasticity enhancement”. In: *Nature Communications* 14.1 (June 2023). ISSN: 2041-1723. DOI: 10.1038/s41467-023-39018-y. URL: http://dx.doi.org/10.1038/s41467-023-39018-y.

[19] Ioannis Mandralis, Reza Nemovi, Alireza Ramezani, Richard M. Murray, and Morteza Gharib. "ATMO: an aerially transforming morphobot for dynamic ground-aerial transition". In: *Communications Engineering* 4.1 (Apr. 2025). ISSN: 2731-3395. DOI: 10.1038/s44172-025-00413-6. URL: http://dx.doi.org/10.1038/s44172-025-00413-6. (published).

[20] Masahiro Ootsuka, Chinthaka Premachandra, and Kiyotaka Kato. "Development of an air-ground operational robot and its fundamental controlling approach". In: *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*. 2014, pp. 1470–1474. DOI: 10.1109/SCIS-ISIS.2014.7044710.

[21] Shatadal Mishra, Karishma Patnaik, YiZhuang Garrard, Zachary Chase, Michael Ploughe, and Wenlong Zhang. "Ground Trajectory Control of an Unmanned Aerial-Ground Vehicle using Thrust Vectoring for Precise Grasping". In: *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2020, pp. 1270–1275. DOI: 10.1109/AIM43001.2020.9158943.

[22] Miguel Pimentel and Meysam Basiri. "A Bimodal Rolling-Flying Robot for Micro Level Inspection of Flat and Inclined Surfaces". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5135–5142. DOI: 10.1109/LRA.2022.3154027.

[23] David D. Fan, Rohan Thakker, Tara Bartlett, Meriem Ben Miled, Leon Kim, Evangelos Theodorou, and Ali-akbar Agha-mohammadi. "Autonomous Hybrid Ground/Aerial Mobility in Unknown Environments". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 3070–3077. DOI: 10.1109/IROS40897.2019.8968276.

[24] Daoxun Zhang, Ce Guo, Haoran Ren, Pengming Zhu, Ming Xu, and Huimin Lu. "The Design of an Aerial/Ground Dual-modal Mobile Robot for Exploring Complex Environments". In: *2021 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. 2021, pp. 393–398. DOI: 10.1109/RCAR52367.2021.9517607.

[25] Yang Wang, Naisi Zhang, Bo Pan, Bo Su, and Shengfei Li. "Multi-mode Motion Control System Design and Implementation for an Air-ground Amphibious Robot". In: *2021 China Automation Congress (CAC)*. 2021, pp. 7751–7756. DOI: 10.1109/CAC53003.2021.9727502.

[26] Katsuaki Tanaka, Di Zhang, Sho Inoue, Ritaro Kasai, Hiroya Yokoyama, Koki Shindo, Ko Matsuhiro, Shigeaki Marumoto, Hiroyuki Ishii, and Atsuo Takanishi. "A design of a small mobile robot with a hybrid locomotion mechanism of wheels and multi-rotors". In: *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*. 2017, pp. 1503–1508. DOI: 10.1109/ICMA.2017.8016039.

[27] Hyungho Chris Choi, Inhwan Wee, Micah Corah, Sahand Sabet, Taeyeon Kim, Thomas Touma, David Hyunchul Shim, and Ali-akbar Agha-mohammadi. "BAXTER: Bi-Modal Aerial-Terrestrial Hybrid Vehicle for Long-Endurance Versatile Mobility". In: *Springer Proceedings in Advanced Robotics*. Springer International Publishing, 2021, pp. 60–72. ISBN: 9783030711511. DOI: `10.1007/978-3-030-71151-1_6`. URL: `http://dx.doi.org/10.1007/978-3-030-71151-1_6`.

[28] Nir Meiri and David Zarrouk. "Flying STAR, a Hybrid Crawling and Flying Sprawl Tuned Robot". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5302–5308. DOI: `10.1109/ICRA.2019.8794260`.

[29] Robert Baines, Sree Kalyan Patiballa, Joran Booth, Luis Ramirez, Thomas Sipple, Andonny Garcia, Frank Fish, and Rebecca Kramer-Bottiglio. "Multi-environment robotic transitions through adaptive morphogenesis". In: *Nature* 610.7931 (2022), pp. 283–289.

[30] Jiefeng Sun, Elisha Lerner, Brandon Tighe, Clint Middlemist, and Jianguo Zhao. "Embedded shape morphing for morphologically adaptive robots". In: *Nature Communications* 14.1 (2023), p. 6023.

[31] Robert Baines, Frank Fish, Josh Bongard, and Rebecca Kramer-Bottiglio. "Robots that evolve on demand". In: *Nature Reviews Materials* (2024), pp. 1–14.

[32] Peter Zheng, Xinkai Tan, Basaran Bahadir Kocer, Erdeng Yang, and Mirko Kovac. "TiltDrone: A Fully-Actuated Tilting Quadrotor Platform". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6845–6852. DOI: `10.1109/LRA.2020.3010460`.

[33] Fernando Ruiz, Begoña C. Arrue, and Aníbal Ollero. "Aeroelastics-aware compensation system for soft aerial vehicle stabilization". In: *Frontiers in Robotics and AI* 9 (Nov. 2022). ISSN: 2296-9144. DOI: `10.3389/frobt.2022.1005620`. URL: `http://dx.doi.org/10.3389/frobt.2022.1005620`.

[34] Ioannis Mandralis, Eric Sihite, Alireza Ramezani, and Morteza Gharib. "Minimum Time Trajectory Generation for Bounding Flight: Combining Posture Control and Thrust Vectoring". In: *2023 European Control Conference (ECC)*. 2023, pp. 1–7. DOI: `10.23919/ECC57647.2023.10178360`. (published).

[35] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. "Alphapilot: Autonomous drone racing". In: *Autonomous Robots* 46.1 (2022), pp. 307–320.

[36] Nathan Bucki and Mark W. Mueller. "Design and Control of a Passively Morphing Quadcopter". In: *2019 International Conference on Robotics and*

Automation (ICRA). 2019, pp. 9116–9122. DOI: 10.1109/ICRA.2019.8794373.

[37] Nathan Bucki, Jerry Tang, and Mark W. Mueller. *Design and Control of a Midair-Reconfigurable Quadcopter using Unactuated Hinges*. 2021. DOI: 10.48550/ARXIV.2103.16632. URL: https://arxiv.org/abs/2103.16632.

[38] Davide Falanga, Kevin Kleber, Stefano Mintchev, Dario Floreano, and Davide Scaramuzza. "The Foldable Drone: A Morphing Quadrotor That Can Squeeze and Fly". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 209–216. DOI: 10.1109/LRA.2018.2885575.

[39] Peter Zheng, Feng Xiao, Pham Huy Nguyen, Andre Farinha, and Mirko Kovac. "Metamorphic aerial robot capable of mid-air shape morphing for rapid perching". In: *Scientific Reports* 13.1 (2023), p. 1297.

[40] Yi-Hsuan Hsiao, Songnan Bai, Yongsen Zhou, Huaiyuan Jia, Runze Ding, Yufeng Chen, Zuankai Wang, and Pakpong Chirarattananon. "Energy efficient perching and takeoff of a miniature rotorcraft". In: *Communications Engineering* 2.1 (June 2023). ISSN: 2731-3395. DOI: 10.1038/s44172-023-00087-y. URL: http://dx.doi.org/10.1038/s44172-023-00087-y.

[41] Yi Hsuan Hsiao and Pakpong Chirarattananon. "Ceiling Effects for Surface Locomotion of Small Rotorcraft". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 6214–6219. DOI: 10.1109/IROS.2018.8593726.

[42] Mark W. Mueller and Raffaello D'Andrea. "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 45–52. DOI: 10.1109/ICRA.2014.6906588.

[43] Adarsh Salagame, Shashwat Pandya, Ioannis Mandralis, Eric Sihite, Alireza Ramezani, and Morteza Gharib. *NMPC-based Unified Posture Manipulation and Thrust Vectoring for Fault Recovery*. 2025. eprint: arXiv:2503.18307.

[44] A. Desbiez, F. Expert, M. Boyron, J. Diperi, S. Viollet, and F. Ruffier. "X-Morf: A crash-separable quadrotor that morfs its X-geometry in flight". In: *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. 2017, pp. 222–227. DOI: 10.1109/RED-UAS.2017.8101670.

[45] Hikaru Otsuka, Masayoshi Kohno, and Keiji Nagatani. "Fountain Flow Visualization in Quadrotor Wake Decreasing Rotor Thrust In-Ground Effect". In: *Journal of Aircraft* 61.3 (May 2024), pp. 761–773. ISSN: 1533-3868. DOI: 10.2514/1.c037101. URL: http://dx.doi.org/10.2514/1.C037101.

[46] DB Adarkar and GR Hall. "The"fountain effect"and VTOL exhaust ingestion." In: *Journal of Aircraft* 6.2 (1969), pp. 109–115.

[47] JMM Barata. “Fountain flows produced by multijet impingement on a ground plane”. In: *Journal of aircraft* 30.1 (1993), pp. 50–56.

[48] Guanya Shi, Xichen Shi, Michael O’Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. “Neural Lander: Stable Drone Landing Control Using Learned Dynamics”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019. DOI: 10.1109/icra.2019.8794351. URL: http://dx.doi.org/10.1109/ICRA.2019.8794351.

[49] Kenichiro Nonaka and Hirokazu Sugizaki. “Integral sliding mode altitude control for a small model helicopter with ground effect compensation”. In: *Proceedings of the 2011 American Control Conference*. 2011, pp. 202–207. DOI: 10.1109/ACC.2011.5991016.

[50] Li Danjun, Zhou Yan, Shi Zongying, and Lu Geng. “Autonomous landing of quadrotor based on ground effect modelling”. In: *2015 34th Chinese Control Conference (CCC)*. 2015, pp. 5647–5652. DOI: 10.1109/ChiCC.2015.7260521.

[51] M Habibnia and J Pascoa. “ANN assisted flow modeling and analysis for a cyclorotor in ground effect”. In: *Aerospace Science and Technology* 95 (2019), p. 105495.

[52] Richard Gordon Budynas, J Keith Nisbett, et al. *Shigley’s mechanical engineering design*. Vol. 9. McGraw-Hill New York, 2011.

[53] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. “PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074.

[54] Emile Kazuo Oshima. *Experimental Studies of Flow Control Techniques for Future Aircraft*. California Institute of Technology, 2023.

[55] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. “Robot Operating System 2: Design, architecture, and uses in the wild”. In: *Science Robotics* 7.66 (May 2022). ISSN: 2470-9476. DOI: 10.1126/scirobotics.abm6074. URL: http://dx.doi.org/10.1126/scirobotics.abm6074.

[56] Antonio Matus-Vargas, Gustavo Rodriguez-Gomez, and Jose Martinez-Carranza. “Ground effect on rotorcraft unmanned aerial vehicles: a review”. In: *Intelligent Service Robotics* 14.1 (Jan. 2021), pp. 99–118. ISSN: 1861-2784. DOI: 10.1007/s11370-020-00344-5. URL: http://dx.doi.org/10.1007/s11370-020-00344-5.

[57] Pedro Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero. “Characterization of the Aerodynamic Ground Effect and Its Influence in Multirotor Control”. In: *International Journal of Aerospace Engineering* 2017 (2017),

pp. 1–17. ISSN: 1687-5974. DOI: 10.1155/2017/1823056. URL: http://dx.doi.org/10.1155/2017/1823056.

[58] Sanjukta Aich, Chahat Ahuja, Tushar Gupta, and P Arulmozhivarman. "Analysis of ground effect on multi-rotors". In: *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*. IEEE. 2014, pp. 236–241.

[59] Xiang He and Kam K. Leang. "Quasi-Steady In-Ground-Effect Model for Single and Multirotor Aerial Vehicles". In: *AIAA Journal* 58.12 (Dec. 2020), pp. 5318–5331. ISSN: 1533-385X. DOI: 10.2514/1.j059223. URL: http://dx.doi.org/10.2514/1.J059223.

[60] I. C. Cheeseman and W. E. Bennett. *The Effect of the Ground on a Helicopter Rotor in Forward Flight*. Tech. rep. Aeronautical Research Council, R&M 3021, 1955.

[61] Samir Bouabdallah and Roland Siegwart. "Full control of a quadrotor". In: *2007 IEEE/RSJ international conference on intelligent robots and systems*. Ieee. 2007, pp. 153–158.

[62] Robotic Systems Lab, ETH Zurich. *Robot Dynamics Lecture Notes*. HS 2017. ETH Zürich, Robotic Systems Lab. 2017. URL: https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2017/RD_HS2017script.pdf.

[63] Marco Hutter, Christian Gehring, and Roland Siegwart. *proNEu: Derivation of analytical kinematics and dynamics*. Tech. rep. Autonomous Systems Lab, ETHZ, 2011.

[64] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. "The GRASP Multiple Micro-UAV Testbed". In: *IEEE Robotics & Automation Magazine* 17.3 (2010), pp. 56–65. DOI: 10.1109/MRA.2010.937855.

[65] Wilson J Rugh and Jeff S Shamma. "Research on gain scheduling". In: *Automatica* 36.10 (2000), pp. 1401–1425.

[66] Douglas J Leith and William E Leithead. "Survey of gain-scheduling analysis and design". In: *International journal of control* 73.11 (2000), pp. 1001–1025.

[67] Richard M. Murray. *Optimization-Based Control*. https://fbswiki.org/wiki/index.php/Supplement:_Optimization-Based_Control. Version 2.3h, posted 12 March 2023. 2023. URL: https://fbswiki.org/wiki/index.php/Supplement:_Optimization-Based_Control.

[68] L S Pontryagin. *Mathematical theory of optimal processes: Mathematical theory of optimal processes L.s. pontryagin selected works volume 4*. Classics of Soviet Mathematics. Amsterdam, Netherlands: Harwood Academic, Mar. 1987.

[69] Richard Bellman and Stuart Dreyfus. *Dynamic Programming*. Vol. 33. Princeton University Press, 2010. ISBN: 9780691146683. URL: `http://www.jstor.org/stable/j.ctv1nxcw0f` (visited on 08/25/2025).

[70] Anastasios I. Mourikis and Stergios I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 3565–3572. DOI: `10.1109/ROBOT.2007.364024`.

[71] Guillem Torrente, Elia Kaufmann, Philipp Föhn, and Davide Scaramuzza. "Data-driven MPC for quadrotors". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3769–3776.

[72] Fang Nan, Sihao Sun, Philipp Foehn, and Davide Scaramuzza. "Nonlinear MPC for Quadrotor Fault-Tolerant Control". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5047–5054. DOI: `10.1109/LRA.2022.3154033`.

[73] Sihao Sun, Angel Romero, Philipp Foehn, Elia Kaufmann, and Davide Scaramuzza. "A Comparative Study of Nonlinear MPC and Differential-Flatness-Based Control for Quadrotor Agile Flight". In: *IEEE Transactions on Robotics* 38.6 (2022), pp. 3357–3373. DOI: `10.1109/TRO.2022.3177279`.

[74] Fang Nan, Sihao Sun, Philipp Foehn, and Davide Scaramuzza. "Nonlinear MPC for Quadrotor Fault-Tolerant Control". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5047–5054. DOI: `10.1109/LRA.2022.3154033`.

[75] Angel Romero, Yunlong Song, and Davide Scaramuzza. "Actor-Critic Model Predictive Control". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 14777–14784. DOI: `10.1109/ICRA57147.2024.10610381`.

[76] Gianluca Frison and Moritz Diehl. "HPIPM: a high-performance quadratic programming framework for model predictive control$**$This research was supported by the German Federal Ministry for Economic Affairs and Energy (BMWi) via eco4wind (0324125B) and DyConPV (0324166B), and by DFG via Research Unit FOR 2401." In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 6563–6569. ISSN: 2405-8963. DOI: `https://doi.org/10.1016/j.ifacol.2020.12.073`. URL: `https://www.sciencedirect.com/science/article/pii/S2405896320303293`.

[77] H.G. Bock and K.J. Plitt. "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems *". In: *IFAC Proceedings Volumes* 17.2 (July 1984), pp. 1603–1608. ISSN: 1474-6670. DOI: 10.1016/s1474-6670(17)61205-9. URL: http://dx.doi.org/10.1016/S1474-6670(17)61205-9.

[78] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. "acados—a modular open-source framework for fast embedded optimal control". In: *Mathematical Programming Computation* 14 (2019), pp. 147–183. URL: https://api.semanticscholar.org/CorpusID:204960656.

[79] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (July 2018), pp. 1–36. ISSN: 1867-2957. DOI: 10.1007/s12532-018-0139-4. URL: http://dx.doi.org/10.1007/s12532-018-0139-4.

[80] Matthew Kelly. "An Introduction to Trajectory Optimization: How to do your own Direct Collocation". In: *SIAM Review* 59.4 (2017), pp. 849–904.

[81] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: Theory, computation, and design*. en. 2017.

[82] Gianluca Frison, Dimitris Kouzoupis, John Bagterp Jørgensen, and Moritz Diehl. "An efficient implementation of partial condensing for Nonlinear Model Predictive Control". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. 2016, pp. 4457–4462. DOI: 10.1109/CDC.2016.7798946.

[83] Ioannis Mandralis, Richard M. Murray, and Morteza Gharib. "Quadrotor Morpho-Transition: Learning vs Model-Based Control Strategies". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2025. arXiv: 2506.14039 [cs.RO]. (accepted).

[84] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 978-0-262-03924-6.

[85] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. "Champion-level drone racing using deep reinforcement learning". In: *Nature* 620.7976 (Aug. 2023), pp. 982–987. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06419-4. URL: http://dx.doi.org/10.1038/s41586-023-06419-4.

[86] Elia Kaufmann, Leonard Bauersfeld, and Davide Scaramuzza. "A Benchmark Comparison of Learned Control Policies for Agile Quadrotor Flight". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 10504–10510. DOI: 10.1109/ICRA46639.2022.9811564.

[87] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. “Control of a Quadrotor With Reinforcement Learning”. In: *IEEE Robotics and Automation Letters* 2.4 (2017), pp. 2096–2103. DOI: 10.1109/LRA.2017.2720851.

[88] Artem Molchanov, Tao Chen, Wolfgang Hönig, James A. Preiss, Nora Ayanian, and Gaurav S. Sukhatme. *Sim-to-(Multi)-Real: Transfer of Low-Level Robust Control Policies to Multiple Quadrotors*. 2019. DOI: 10.48550/ARXIV.1903.04628. URL: https://arxiv.org/abs/1903.04628.

[89] Jonas Eschmann, Dario Albani, and Giuseppe Loianno. *Learning to Fly in Seconds*. 2023. DOI: 10.48550/ARXIV.2311.13081. URL: https://arxiv.org/abs/2311.13081.

[90] Dingqi Zhang, Antonio Loquercio, Jerry Tang, Ting-Hao Wang, Jitendra Malik, and Mark W. Mueller. *A Learning-based Quadcopter Controller with Extreme Adaptation*. 2024. DOI: 10.48550/ARXIV.2409.12949. URL: https://arxiv.org/abs/2409.12949.

[91] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*. 2021. DOI: 10.48550/ARXIV.2108.10470. URL: https://arxiv.org/abs/2108.10470.

[92] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. “Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments”. In: *IEEE Robotics and Automation Letters* 8.6 (2023), pp. 3740–3747. DOI: 10.1109/LRA.2023.3270034.

[93] Nikita Rudin, Hendrik Kolvenbach, Vassilios Tsounis, and Marco Hutter. “Cat-Like Jumping and Landing of Legged Robots in Low Gravity Using Deep Reinforcement Learning”. In: *IEEE Transactions on Robotics* 38.1 (2022), pp. 317–328. DOI: 10.1109/TRO.2021.3084374.

[94] Tarek El-Agroudi, Finn Gross Maurer, Jørgen Anker Olsen, and Kostas Alexis. “In-Flight Attitude Control of a Quadruped using Deep Reinforcement Learning”. In: *8th Annual Conference on Robot Learning*. 2024. URL: https://openreview.net/forum?id=67tTQeO4HQ.

[95] Pierre-Jean Bristeau, Philippe Martin, Erwan Salaun, and Nicolas Petit. “The role of propeller aerodynamics in the model of a quadrotor UAV”. In: *2009 European Control Conference (ECC)*. IEEE, Aug. 2009. DOI: 10.23919/ecc.2009.7074482. URL: http://dx.doi.org/10.23919/ECC.2009.7074482.

[96] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. *Asymmetric Actor Critic for Image-Based Robot Learning*. 2017. DOI: 10.48550/ARXIV.1710.06542. URL: https://arxiv.org/abs/1710.06542.

[97] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms*. 2017. DOI: 10.48550/ARXIV.1707.06347. URL: https://arxiv.org/abs/1707.06347.

[98] Ioannis Mandralis, Reza Nemovi, Alireza Ramezani, Richard Murray, and Morteza Gharib. *ATMO: An Aerially Transforming Morphobot for Dynamic Ground Aerial Transition*. 2025. DOI: 10.48550/arXiv:2503.00609. URL: https://arxiv.org/abs/2503.00609.

[99] Ioannis Mandralis, Severin Schumacher, and Morteza Gharib. Wake Vectoring for Efficient Morphing Flight. 2025. (under review).

[100] D. Counter and B. Brinton. "Thrust vector control for the Space Shuttle Solid Rocket Motor". In: *11th Propulsion Conference*. American Institute of Aeronautics and Astronautics, Aug. 1975. DOI: 10.2514/6.1975-1172. URL: http://dx.doi.org/10.2514/6.1975-1172.

[101] G. Goetz, J. Young, and J. Palcza. "A two-dimensional Airframe Integrated Nozzle design with inflight thrust vectoring and reversing capabilities for advanced fighter aircraft". In: *12th Propulsion Conference*. American Institute of Aeronautics and Astronautics, July 1976. DOI: 10.2514/6.1976-626. URL: http://dx.doi.org/10.2514/6.1976-626.

[102] E. A. Bare and Jr. Pendergraft O. C. *Effect of thrust reverser operation on the lateral-directional characteristics of a three-surface F-15 model at transonic speeds*. NASA Technical Publication NASA-TP-2234. Document ID: 19840005097; Report Numbers: L-15648, NAS 1.60:2234; Public domain work of the U.S. Government. NASA Langley Research Center, Dec. 1983. URL: https://ntrs.nasa.gov/citations/19840005097.

[103] John Fozard. *The British aerospace harrier case study in aircraft design*. Case Study. Reston, VA: American Institute of Aeronautics & Astronautics, Nov. 2001.

[104] B. Berrier and R. Re. "A review of thrust-vectoring schemes for fighter applications". In: *14th Joint Propulsion Conference*. American Institute of Aeronautics and Astronautics, July 1978. DOI: 10.2514/6.1978-1023. URL: http://dx.doi.org/10.2514/6.1978-1023.

[105] Ruben D'Sa, Devon Jenson, and Nikolaos Papanikolopoulos. "SUAV:Q - a hybrid approach to solar-powered flight". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 3288–3294. DOI: 10.1109/ICRA.2016.7487501.

[106] Ruben D'Sa, Devon Jenson, Travis Henderson, Jack Kilian, Bobby Schulz, Michael Calvert, Thaine Heller, and Nikolaos Papanikolopoulos. "SUAV:Q - An improved design for a transformable solar-powered UAV". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1609–1615. DOI: 10.1109/IROS.2016.7759260.

[107] Ruben D'Sa, Travis Henderson, Devon Jenson, Michael Calvert, Thaine Heller, Bobby Schulz, Jack Kilian, and Nikolaos Papanikolopoulos. "Design and experiments for a transformable solar-UAV". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 3917–3923. DOI: 10.1109/ICRA.2017.7989451.

[108] Ruben D'Sa and Nikolaos Papanikolopoulos. "Design and Experiments for MultI-Section-Transformable (MIST)-UAV". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 1878–1883. DOI: 10.1109/ICRA.2019.8793575.

[109] O. C. Carholt, E. Fresk, G. Andrikopoulos, and G. Nikolakopoulos. "Design, modelling and control of a Single Rotor UAV". In: *2016 24th Mediterranean Conference on Control and Automation (MED)*. 2016, pp. 840–845. DOI: 10.1109/MED.2016.7536015.

[110] Travis Henderson, Ryan Favour, Ben Hamlen, Imraan Mitha, Erika Bowe, and Nikolaos Papanikolopoulos. "Hovering Locomotion for UAVs With Thrust-Vectoring Control Surfaces". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5214–5221. DOI: 10.1109/LRA.2022.3153988.

[111] William John Macquorn Rankine. "On the mechanical principles of the action of propellers". In: *Transactions of the Institution of Naval Architects* 6 (1865).

[112] Robert Edmund Froude. "On the part played in propulsion by differences of fluid pressure". In: *Trans. Inst. Naval Architects* 30 (1889), p. 390.

[113] Hrvoje Jasak, Aleksandar Jemcov, Zeljko Tukovic, et al. "OpenFOAM: A C++ library for complex physics simulations". In: *International workshop on coupled methods in numerical dynamics*. Vol. 1000. Dubrovnik, Croatia). 2007, pp. 1–20.

[114] Hrvoje Jasak. "OpenFOAM: Open source CFD in research and industry". In: *International journal of naval architecture and ocean engineering* 1.2 (2009), pp. 89–94.

[115] F. Menter. "Zonal Two Equation k-w Turbulence Models For Aerodynamic Flows". In: *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference*. American Institute of Aeronautics and Astronautics, July 1993. DOI: 10.2514/6.1993-2906. URL: http://dx.doi.org/10.2514/6.1993-2906.

[116] F. R. Menter. “Two-equation eddy-viscosity turbulence models for engineering applications”. In: *AIAA Journal* 32.8 (Aug. 1994), pp. 1598–1605. ISSN: 1533-385X. DOI: 10.2514/3.12149. URL: http://dx.doi.org/10.2514/3.12149.

[117] G.A.M. van Kuik, J.N. Sørensen, and V.L. Okulov. “Rotor theories by Professor Joukowsky: Momentum theories”. In: *Progress in Aerospace Sciences* 73 (2015). “Special N.E. Joukowsky Volume” “This special volume contains two comprehensive papers on the history of rotor aerodynamics with special emphasis on the original pioneering contributions of Professor N.E. Joukowsky.”, pp. 1–18. ISSN: 0376-0421. DOI: https://doi.org/10.1016/j.paerosci.2014.10.001. URL: https://www.sciencedirect.com/science/article/pii/S0376042114000955.

[118] J.M.V. Rayner. “Bounding and undulating flight in birds”. In: *Journal of Theoretical Biology* 117.1 (1985), pp. 47–77. ISSN: 0022-5193. DOI: https://doi.org/10.1016/S0022-5193(85)80164-8. URL: https://www.sciencedirect.com/science/article/pii/S0022519385801648.

[119] Jeremy M. V. Rayner, Paolo W. Viscardi, Sally Ward, and John R. Speakman. “Aerodynamics and Energetics of Intermittent Flight in Birds1”. In: *American Zoologist* 41.2 (Aug. 2015), pp. 188–204. ISSN: 0003-1569. DOI: 10.1093/icb/41.2.188. eprint: https://academic.oup.com/icb/article-pdf/41/2/188/5881279/i0003-1569-041-02-0188.pdf. URL: https://doi.org/10.1093/icb/41.2.188.

[120] A.J. Ward-Smith. “Analysis of the aerodynamic performance of birds during bounding flight”. In: *Mathematical Biosciences* 68.1 (1984), pp. 137–147. ISSN: 0025-5564. DOI: https://doi.org/10.1016/0025-5564(84)90077-4. URL: https://www.sciencedirect.com/science/article/pii/0025556484900774.

[121] C. Harvey, V. B. Baliga, J. C. M. Wong, D. L. Altshuler, and D. J. Inman. “Birds can transition between stable and unstable states via wing morphing”. In: *Nature* 603.7902 (Mar. 2022), pp. 648–653. DOI: 10.1038/s41586-022-04477-8. URL: https://doi.org/10.1038/s41586-022-04477-8.

[122] URL: http://www.moorhen.me.uk/iodsubject/birds_-_woodpeckers_03.htm.

[123] Mina Kamel, Sebastian Verling, Omar Elkhatib, Christian Sprecher, Paula Wulkop, Zachary Taylor, Roland Siegwart, and Igor Gilitschenski. “The Voliro Omniorientational Hexacopter: An Agile and Maneuverable Tiltable-Rotor Aerial Vehicle”. In: *IEEE Robotics & Automation Magazine* 25.4 (2018), pp. 34–44. DOI: 10.1109/MRA.2018.2866758.

[124] Na Zhao, Yudong Luo, Hongbin Deng, and Yantao Shen. “The deformable quad-rotor: Design, kinematics and dynamics characterization, and flight performance validation”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 2391–2396. DOI: 10.1109/IROS.2017.8206052.

[125] Moju Zhao, Tomoki Anzai, Fan Shi, Xiangyu Chen, Kei Okada, and Masayuki Inaba. “Design, Modeling, and Control of an Aerial Robot DRAGON: A Dual-Rotor-Embedded Multilink Robot With the Ability of Multi-Degree-of-Freedom Aerial Transformation”. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1176–1183. DOI: 10.1109/LRA.2018.2793344.

[126] Jinjie Li, Junichiro Sugihara, and Moju Zhao. “Servo Integrated Nonlinear Model Predictive Control for Overactuated Tiltable-Quadrotors”. In: *IEEE Robotics and Automation Letters* 9.10 (Oct. 2024), pp. 8770–8777. ISSN: 2377-3774. DOI: 10.1109/lra.2024.3451391. URL: http://dx.doi.org/10.1109/LRA.2024.3451391.

[127] Markus Ryll and Robert K. Katzschmann. “SMORS: A soft multirotor UAV for multimodal locomotion and robust interaction”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2022, pp. 2010–2016. DOI: 10.1109/icra46639.2022.9812044. URL: http://dx.doi.org/10.1109/ICRA46639.2022.9812044.

[128] Moju Zhao, Tomoki Anzai, and Takuzumi Nishio. “Design, Modeling, and Control of a Quadruped Robot SPIDAR: Spherically Vectorable and Distributed Rotors Assisted Air-Ground Quadruped Robot”. In: *IEEE Robotics and Automation Letters* 8.7 (July 2023), pp. 3923–3930. ISSN: 2377-3774. DOI: 10.1109/lra.2023.3272285. URL: http://dx.doi.org/10.1109/LRA.2023.3272285.

[129] Kyriakos C. Giannakoglou and Dimitrios I. Papadimitriou. “Adjoint Methods for Shape Optimization”. In: *Optimization and Computational Fluid Dynamics*. Springer Berlin Heidelberg, pp. 79–108. ISBN: 9783540721536. DOI: 10.1007/978-3-540-72153-6_4. URL: http://dx.doi.org/10.1007/978-3-540-72153-6_4.

[130] Ambar Garofano-Soldado, Antonio Gonzalez-Morgado, Guillermo Heredia, and Anibal Ollero. “Assessment and Modeling of the Aerodynamic Ground Effect of a Fully-Actuated Hexarotor With Tilted Propellers”. In: *IEEE Robotics and Automation Letters* 9.2 (2024), pp. 1907–1914. DOI: 10.1109/LRA.2024.3350975.

[131] Ambar Garofano-Soldado, Pedro J. Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero. “Numerical-experimental evaluation and modelling of aerodynamic ground effect for small-scale tilted propellers at low Reynolds numbers”. In: *Aerospace Science and Technology* 126 (July 2022),

p. 107625. ISSN: 1270-9638. DOI: 10.1016/j.ast.2022.107625. URL: http://dx.doi.org/10.1016/j.ast.2022.107625.

[132] Yiliang Liu, Zi Kan, Huadong Li, Yuzhe Gao, Daochun Li, and Shiwei Zhao. "Analysis and modeling of the aerodynamic ceiling effect on small-scale propellers with tilted angles". In: *Aerospace Science and Technology* 147 (Apr. 2024), p. 109038. ISSN: 1270-9638. DOI: 10.1016/j.ast.2024.109038. URL: http://dx.doi.org/10.1016/j.ast.2024.109038.

[133] Wayne Johnson. *Helicopter Theory*. en. Dover Books on Aeronautical Engineering. Mineola, NY: Dover Publications, Jan. 1995.

[134] Robert Deters and Michael Selig. "Static testing of micro propellers". In: *26th AIAA applied aerodynamics conference*. 2008, p. 6246.

[135] Leonard Bauersfeld and Davide Scaramuzza. "Range, Endurance, and Optimal Speed Estimates for Multicopters". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2953–2960. DOI: 10.1109/LRA.2022.3145063.

[136] Daniel Mellinger and Vijay Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 2520–2525.

[137] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories". In: *IEEE Robotics and Automation Letters* 3.2 (2017), pp. 620–626.

Appendix A

RANGE IMPROVEMENT THROUGH BI-MODAL LOCOMOTION

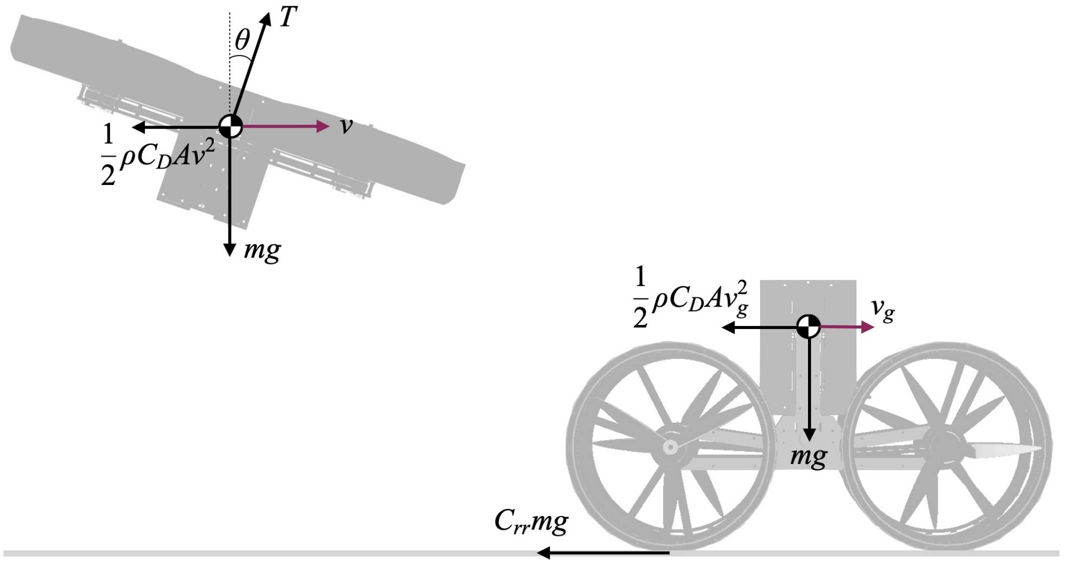


Figure A.1: Force balance in steady flight and in wheeled locomotion. In flight the total thrust force is T, the pitch angle is θ, and the forward velocity is v. In driving locomotion the forward velocity is v_g.

This appendix estimates the energetic benefit of combining flying and wheeled locomotion using simplified models. The problem setup is depicted in Figure A.1. Considering steady forward flight, the projection of the thrust force in the forward axis must balance the drag force. Likewise, the projection of the thrust on the vertical axis must balance the robot weight. This balance of forces yields

$$T\cos\theta = mg, \tag{A.1}$$

$$T\sin\theta = \frac{1}{2}\rho C_D A v^2, \tag{A.2}$$

where T is the total thrust force produced by the propellers, θ is the forward pitching angle, A is the drag area of the flying robot, C_D is the drag coefficient, and ρ is the density of air. The aerodynamic drag equation was used. The drag area is a function of the flight pitch angle:

$$A(\theta) = A_f\cos(\theta) + A_t\sin(\theta), \tag{A.3}$$

where A_f is the frontal area and A_t is the top area. The velocity and thrust are also functions of the pitch angle and can be obtained by combining Equations (A.1) and (A.2) as

$$v(\theta) = \sqrt{\frac{2mg\tan\theta}{\rho C_D A(\theta)}}, \tag{A.4}$$

$$T(\theta) = \frac{mg}{\cos\theta}. \tag{A.5}$$

The hover power can be estimated using Actuator Disk Theory [112, 133]. Applying actuator disk theory for an N_{rotor} vehicle, the expended propulsion power is

$$P_{\text{prop}}(\theta) = \frac{T(\theta)^{3/2}}{f_M\sqrt{2\rho A_{\text{rotor}} N_{\text{rotor}}}}, \tag{A.6}$$

where A_{rotor} is the area of the rotor disk and N_{rotor} is the number of propellers used for flight, and $f_M \in [0, 1]$ is the figure of merit [134] which accounts for the losses of real propellers compared to the ideal actuator-disk case. Thus the flying cost of transport e_f, or energy consumed per unit distance, can be written as

$$e_f(\theta) = \frac{P_{\text{prop}}(\theta)}{v(\theta)} + \frac{1}{2}\rho C_D A(\theta) v(\theta)^2. \tag{A.7}$$

Differentiating $e_f(\theta)$ with respect to θ and setting to zero allows to find the flight pitch angle θ^* that results in the minimum cost of transport e_f^*. When on the ground, and assuming flat terrain and no turning, motion is impeded due to the rolling resistance as well as the drag force. This results in a cost of transport:

$$e_g(v_g) = \frac{C_{rr} mg + \frac{1}{2}\rho C_D A_f v_g^2}{\eta_g}, \tag{A.8}$$

where C_{rr} is the coefficient of rolling resistance, η_g is the drivetrain efficiency, and v_g is the ground speed. Assuming that the robot spends a fraction, α, of time on the ground and $1 - \alpha$ in flight, results in an overall improvement in range $\mathcal{I}$ given by

$$\mathcal{I}(v_g; \alpha) = \frac{e_f^*}{(1-\alpha)e_f^* + \alpha e_g(v_g)}. \tag{A.9}$$

The above quantity is plotted against v_g for different values of α in Figure A.2 for the parameters $N_{\text{rotor}} = 4$, $R_{\text{rotor}} = 0.1143$ m, $m = 5.5$ kg, $g = 9.81$ ms^{-2}, $\rho = 1.225$ kgm^{-3}, $f_M = 0.7$, $C_D = 1$ (drag coefficient of a cube), $a_f = 0.09$ m^2, $a_t = 0.36$ m^2, $C_{rr} = 0.1$ (rolling resistance of turf), and $\eta_g = 0.8$.

The optimal pitch angle for lowest cost-of-transport flight is shown in Figure A.3. Although the numerical value of the optimal pitch angle depends on the particular parameters of the aerial robot, the existence of an optimal pitch angle—and consequently optimal flight velocity—is noteworthy and has also been observed in [135]. Figure A.2 shows the overall improvement in range for three different values of α. The maximum improvement in range is around 15 times and declines as the proportion α increases. The ground speed v_g also significantly affects the overall bi-modal ground-aerial locomotion efficiency. At higher ground velocities, the benefit of wheeled locomotion declines since drag dominates the rolling resistance. At around 35 ms^{-1} flying is more efficient than ground locomotion and bi-modal locomotion would not provide any advantage.

Propulsion Power in Forward Flight

The propulsion model used above relates the induced velocity to the thrust, assuming that the rotorcraft is in hover conditions. In reality this assumption is not satisfied—as the rotorcraft moves forward the induced velocity changes due to the influence of the free-stream air flow. This case is handled by generalized Actuator Disk Theory for forward flight [133]. For an N_{rotor} vehicle with total thrust $T(\theta)$, the induced velocity of a single rotor is the solution of the implicit equation $f(v_i; \theta) = 0$ with f defined as

$$f(v_i; \theta) = v_i - \frac{T(\theta)}{2\rho A_{\text{rotor}} N_{\text{rotor}} \sqrt{v(\theta)^2 + 2v(\theta)v_i \sin\theta + v_i^2}}. \tag{A.10}$$

For each θ, this equation admits a solution $v_i(\theta)$. Using this information, the propulsion power is given by $P_{\text{prop}} = T(\theta)v_i(\theta)/f_M$. The power needed to overcome drag is the same as before. Using the same procedure, and parameters, as above, the range improvement under the generalized model for induced velocity is depicted in Figure A.4. The optimal pitch angle is depicted in Figure A.5. This analysis yields more conservative estimates: at certain forward velocities, flight consumes less power than hover. Accordingly, we use the generalized forward-flight model to substantiate the energetic advantage of bi-modal locomotion in Chapter 6.1.

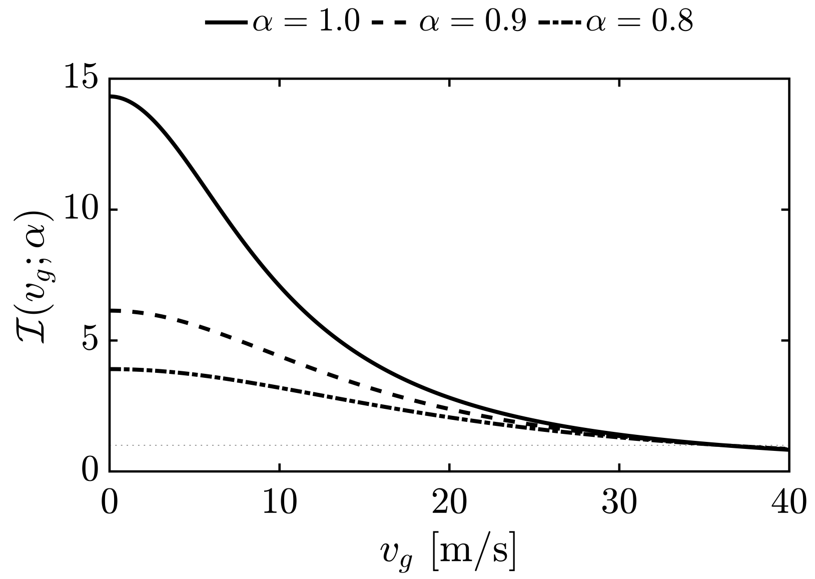


Figure A.2: Range improvement when using bi-modal ground aerial locomotion compared to single-mode flight where propulsion power consumption has been estimated under hover conditions.

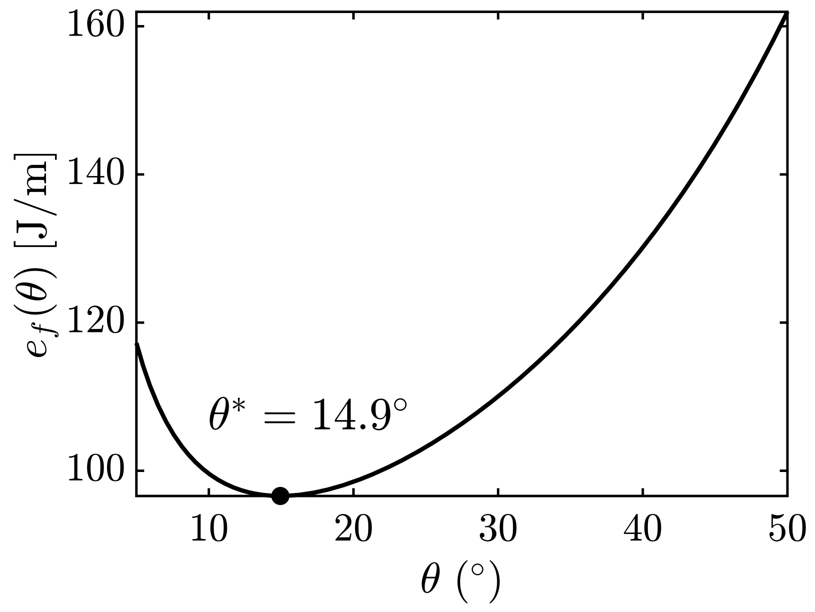


Figure A.3: Flying cost of transport plotted against the pitch angle where propulsion power consumption has been estimated under hover conditions. The optimal pitch in this case is $\theta^* = 14.9°$.

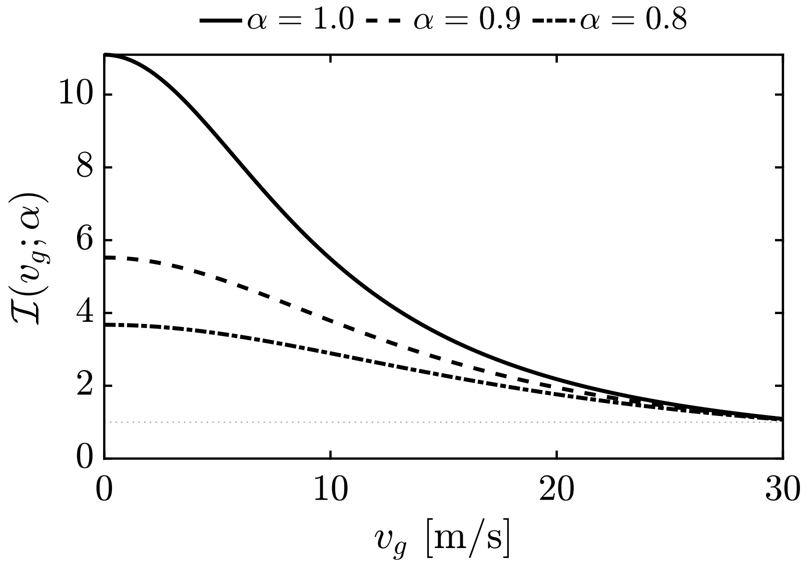


Figure A.4: Range improvement when using bi-modal ground aerial locomotion compared to single-mode flight where propulsion power consumption has been estimated under generalized forward flight conditions.

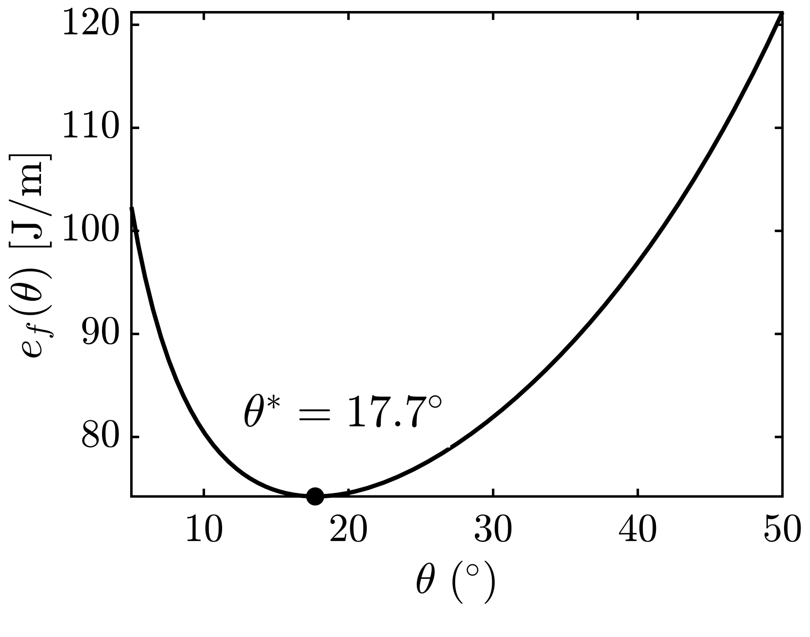


Figure A.5: Flying cost of transport plotted against the pitch angle where propulsion power consumption has been estimated under generalized forward flight conditions. The optimal pitch in this case is $\theta^* = 17.7°$.

Appendix B

DIFFERENTIAL FLATNESS OF MORPHING QUADCOPTER MODEL

Differential flatness is a structural property of certain dynamical systems whereby there exists a set of *flat outputs* such that all system states and inputs can be written as algebraic functions of these outputs and a finite number of their time-derivatives. In practice, this turns motion planning and control into the problem of designing smooth trajectories for the flat outputs, from which dynamically feasible state and input trajectories are obtained algebraically, enabling efficient feedforward generation and trajectory optimization. For robotic systems this property is useful because no nonlinear ODEs need to be integrated in order to compute feasible trajectories. Instead, feasible nominal trajectories and inputs can be computed by solving sets of algebraic equations.

This property has been exploited for quadcopter control to generate smooth, minimum-snap, trajectories that pass through a set of keyframes (Cartesian position and yaw) [136]. To achieve this, the authors of [136] showed that quadcopters are differentially flat with flat outputs (x, y, z, ψ); given a desired position–yaw trajectory and its derivatives, one can compute the required thrust magnitude and body attitudes (roll and pitch) algebraically, which underlies minimum-snap flight planning and simplifies real-time trajectory tracking and replanning.

In this note, we extend the differential flatness property to the morphing quadcopter model, assuming that the arms are inertialess; implying that the inertia matrix is fixed. We show that the system is differentially flat under these assumptions, with flat outputs (x, y, z, ψ, ρ), where ρ is the ratio of vertical body force to horizontal body force. We begin by defining differential flatness formally, showing that the morphing quadcopter model is differentially flat, and verifying our results numerically.

Differential Flatness

Following the definition from [67], a nonlinear system

$$\dot{x} = f(x, u), \qquad x \in \mathbb{R}^n, u \in \mathbb{R}^m \tag{B.1}$$

$$y = h(x, u), \qquad y \in \mathbb{R}^p, \tag{B.2}$$

is *differentially flat* if there exists a function α such that

$$\sigma = \alpha(x, u, \dot{u}, ..., u^{(p)}), \tag{B.3}$$

and we can write the solutions of the nonlinear system as functions of z and a finite number of derivatives:

$$x = \beta(\sigma, \dot{\sigma}, ..., \sigma^{(q)}), \tag{B.4}$$
$$u = \gamma(\sigma, \dot{\sigma}, ..., \sigma^{(q)}). \tag{B.5}$$

The collection of variables $\bar{\sigma} = (\sigma, \dot{\sigma}, ..., \sigma^{(q)})$ is called the *flat flag*.

Problem Setup

The proof of differential flatness for quadrotors presented in [136] and [137] is extended to the morphing quadcopter model. The inertia and center-of-gravity are considered fixed and rotor drag is neglected. We redefine the control inputs to be $u = (u_1, u_2, u_3, u_4, \varphi)$ where

$$u_1 = \Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2, \tag{B.6}$$
$$u_2 = -\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2, \tag{B.7}$$
$$u_3 = -\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2, \tag{B.8}$$
$$u_4 = -\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2, \tag{B.9}$$

and we assume that the tilt angle can be directly assigned. With this selection of inputs, the body forces are

$$T = k_T u_1 \cos\varphi \boldsymbol{z}_B \equiv m\xi_1 \tag{B.10}$$
$$F = k_T u_2 \sin\varphi \boldsymbol{y}_B \equiv m\xi_2 \tag{B.11}$$

where $\xi_1 = \frac{1}{m} k_T u_1 \cos\varphi$ and $\xi_2 = \frac{1}{m} k_T u_2 \sin\varphi$. The torque at the center-of-gravity is given by

$$\boldsymbol{\tau} = \begin{bmatrix} k_T(b + a\cos\varphi)u_2 \\ k_T(c\cos\varphi - k_M \sin\varphi)u_3 \\ k_T(c\sin\varphi + k_M\cos\varphi)u_4 \end{bmatrix}, \tag{B.12}$$

The dynamics are given in Equations (3.42)–(3.45) and reformulated here:

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{B.13}$$
$$\dot{\boldsymbol{v}} = -g\boldsymbol{z}_W + \xi_1 \boldsymbol{z}_B + \xi_2 \boldsymbol{y}_B \tag{B.14}$$
$$\dot{\boldsymbol{R}} = \boldsymbol{R}\hat{\omega} \tag{B.15}$$
$$\dot{\omega} = I^{-1}\left[\boldsymbol{\tau} - \omega \times I\omega\right], \tag{B.16}$$

where $\hat{\omega}$ is the skew symmetric matrix composed of elements of the angular velocity. The state of the robot is $x = (\boldsymbol{p}, \boldsymbol{\theta}, \boldsymbol{v}, \boldsymbol{\omega})$. The orientation is represented by the Euler ZYX representation and the rotation matrix is given by the basis vectors of the body frame expressed in the world frame: $\boldsymbol{R} = [\boldsymbol{x}_B \ \boldsymbol{y}_B \ \boldsymbol{z}_B]$. We will show that the system with the given state and inputs and subject to the above dynamics is differentially flat with flat output $\boldsymbol{\sigma} = (x, y, z, \psi, \rho)$ where (x, y, z) is the Cartesian position $\boldsymbol{p}$, ψ is the yaw angle, and ρ is defined as ratio of force due to the thrusters in the $\boldsymbol{z}_B$ direction and the $\boldsymbol{y}_B$ direction:

$$\rho = \frac{\xi_1}{\xi_2} = \frac{u_1}{u_2} \cot \varphi. \tag{B.17}$$

Proof

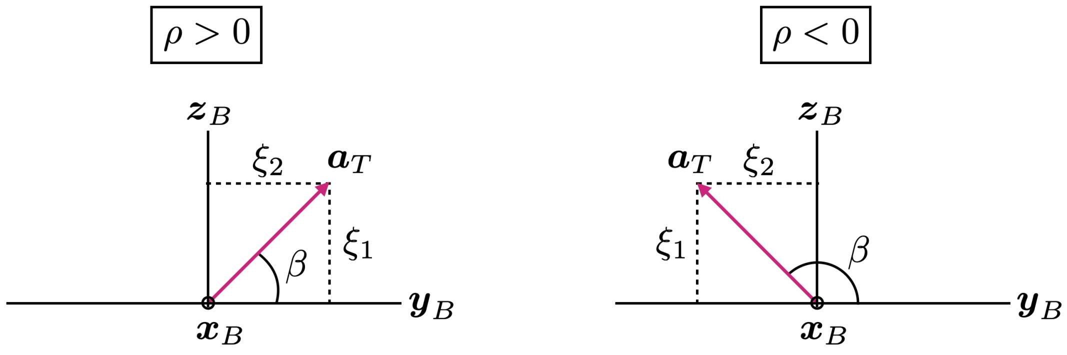


Figure B.1: Definition of the flat output ρ.

Position. The position $\boldsymbol{p}$ is part of the flat output and thus trivially a function of the flat output. This also implies that the velocity $\boldsymbol{v}$, acceleration $\boldsymbol{a}$, jerk $\boldsymbol{j}$, and snap $\boldsymbol{s}$ are also functions of the flat outputs.

Orientation. We show that the orientation $\boldsymbol{R}$ is a function of the flat output and a finite number of its derivatives. Rewrite (B.14) as

$$\xi_1 \boldsymbol{z}_B + \xi_2 \boldsymbol{y}_B = \boldsymbol{a} + g\boldsymbol{z}_W \equiv \boldsymbol{a}_T. \tag{B.18}$$

First notice that $\boldsymbol{x}_B^\top \boldsymbol{a}_T = 0$. Since the projection of $\boldsymbol{x}_B$ on the $(\boldsymbol{x}_W, \boldsymbol{y}_W)$ plane must be parallel to the heading direction then: $\boldsymbol{x}_B^\top \boldsymbol{y}_\psi = 0$, where $\boldsymbol{y}_\psi = [-\sin\psi \ \cos\psi \ 0]^\top$. Combining the two conditions on $\boldsymbol{x}_B$ yields

$$\boldsymbol{x}_B = \frac{\boldsymbol{y}_\psi \times \boldsymbol{a}_T}{||\boldsymbol{y}_\psi \times \boldsymbol{a}_T||}, \tag{B.19}$$

unless $\boldsymbol{y}_\psi$ and $\boldsymbol{a}_T$ are collinear in which case $\boldsymbol{x}_B$ will be momentarily undefined. Furthermore, since $\boldsymbol{a}_T$ is orthogonal to $\boldsymbol{x}_B$ this means that $\boldsymbol{a}_T$ lies in the $(\boldsymbol{y}_B, \boldsymbol{z}_B)$ plane—see Figure B.1. We can thus obtain $\boldsymbol{y}_B$ by rotating $\boldsymbol{a}_T$ about $\boldsymbol{x}_B$ by $-\beta$. β is obtained uniquely as a function of ρ by using the knowledge that $u_1 > 0$ (no negative thrust), which implies that $\xi_1 > 0$ meaning that $\boldsymbol{a}_T$ can lie in the first or second quadrant only:

$$\beta = \begin{cases} \arctan(\rho), & \rho > 0 \\ \pi - \arctan(|\rho|), & \rho < 0 \end{cases}.$$

Given β, $\boldsymbol{y}_B$ can be obtained by rotating $\boldsymbol{a}_T$ around the $\boldsymbol{x}_B$ using the Rodrigues rotation formula. Thus the orientation $\boldsymbol{R} = [\boldsymbol{x}_B\ \boldsymbol{y}_B\ \boldsymbol{z}_B]$ is uniquely defined by

$$\boldsymbol{x}_B = \frac{\boldsymbol{y}_\psi \times \boldsymbol{a}_T}{||\boldsymbol{y}_\psi \times \boldsymbol{a}_T||}, \tag{B.20}$$

$$\boldsymbol{y}_B = \frac{1}{||\boldsymbol{a}_T||}\left[\boldsymbol{a}_T \cos\beta - (\boldsymbol{x}_B \times \boldsymbol{a}_T)\sin\beta\right], \tag{B.21}$$

$$\boldsymbol{z}_B = \boldsymbol{x}_B \times \boldsymbol{y}_B, \tag{B.22}$$

$$\beta = \begin{cases} \arctan(\rho) & \rho > 0 \\ \pi - \arctan(|\rho|) & \rho < 0 \end{cases}, \tag{B.23}$$

where $\boldsymbol{y}_\psi = [-\sin\psi\ \ \cos\psi\ \ 0]^\top$. Finally, we obtain an intermediate result which is necessary for subsequent derivations—namely that the acceleration in the $\boldsymbol{z}_B$ and $\boldsymbol{y}_B$ directions can be obtained by simple projection of $\boldsymbol{a}_T$ onto the now known $\boldsymbol{z}_B$ and $\boldsymbol{y}_B$ vectors:

$$\xi_1 = \boldsymbol{a}_T \cdot \boldsymbol{z}_B \tag{B.24}$$

$$\xi_2 = \boldsymbol{a}_T \cdot \boldsymbol{y}_B \tag{B.25}$$

Angular velocity. We show that the angular velocity $\boldsymbol{\omega}$ is also a function of the flat outputs and its derivatives. First, obtain the derivatives of the body frame vectors. Compute $\dot{\boldsymbol{x}}_B$ by defining $\tilde{\boldsymbol{x}}_B = \boldsymbol{y}_\psi \times \boldsymbol{a}_T$, writing $\boldsymbol{x}_B = \frac{\tilde{\boldsymbol{x}}_B}{||\tilde{\boldsymbol{x}}_B||}$, and taking the derivative of a normalized vector:

$$\dot{\boldsymbol{x}}_B = \frac{\dot{\tilde{\boldsymbol{x}}}_B}{||\tilde{\boldsymbol{x}}_B||} - \tilde{\boldsymbol{x}}_B \frac{\tilde{\boldsymbol{x}}_B^\top \dot{\tilde{\boldsymbol{x}}}_B}{||\tilde{\boldsymbol{x}}_B||^3}. \tag{B.26}$$

Using $\dot{\boldsymbol{y}}_\psi = -\dot{\psi}\boldsymbol{x}_\psi$ and $\dot{\boldsymbol{a}}_T = \boldsymbol{j}$, we obtain

$$\dot{\tilde{\boldsymbol{x}}}_B = -\dot{\psi}(\boldsymbol{x}_\psi \times \boldsymbol{a}_T) + \boldsymbol{y}_\psi \times \boldsymbol{j} \tag{B.27}$$

which closes the expression for $\dot{\boldsymbol{x}}_B$. Next we obtain $\dot{\boldsymbol{y}}_B$:

$$\dot{\boldsymbol{y}}_B = \frac{1}{||\boldsymbol{a}_T||}\left[\boldsymbol{j}\cos\beta - (\dot{\boldsymbol{x}}_B \times \boldsymbol{a}_T + \boldsymbol{x}_B \times \boldsymbol{j})\sin\beta\right] - \frac{\boldsymbol{j}\cdot\boldsymbol{a}_T}{||\boldsymbol{a}_T||^2}\boldsymbol{y}_B. \tag{B.28}$$

Finally we can compute $\dot{\boldsymbol{z}}_B = \dot{\boldsymbol{x}}_B \times \boldsymbol{y}_B + \boldsymbol{x}_B \times \dot{\boldsymbol{y}}_B$ and the angular velocity can be obtained as a function of the flat outputs by differentiating the orientation dynamics and inverting:

$$\hat{\omega} = \boldsymbol{R}^\top \dot{\boldsymbol{R}}. \tag{B.29}$$

Angular accelerations. Now we compute the second derivatives of the body vectors. First compute $\ddot{\boldsymbol{x}}_B$,

$$\begin{aligned}
\ddot{\boldsymbol{x}}_B &= \frac{d}{dt}\left[\frac{\dot{\tilde{\boldsymbol{x}}}_B}{||\tilde{\boldsymbol{x}}_B||} - \tilde{\boldsymbol{x}}_B\frac{\tilde{\boldsymbol{x}}_B^\top\dot{\tilde{\boldsymbol{x}}}_B}{||\tilde{\boldsymbol{x}}_B||^3}\right] && \text{(B.30)}\\
&= \frac{\ddot{\tilde{\boldsymbol{x}}}_B}{||\tilde{\boldsymbol{x}}_B||} - \frac{2\dot{\tilde{\boldsymbol{x}}}_B(\dot{\tilde{\boldsymbol{x}}}_B^\top\tilde{\boldsymbol{x}}_B)}{||\tilde{\boldsymbol{x}}_B||^3} - \tilde{\boldsymbol{x}}_B\frac{\dot{\tilde{\boldsymbol{x}}}_B^\top\dot{\tilde{\boldsymbol{x}}}_B}{||\tilde{\boldsymbol{x}}_B||^3} && \text{(B.31)}\\
&\quad - \tilde{\boldsymbol{x}}_B\frac{\tilde{\boldsymbol{x}}_B^\top\ddot{\tilde{\boldsymbol{x}}}_B}{||\tilde{\boldsymbol{x}}_B||^3} + 3\tilde{\boldsymbol{x}}_B\frac{(\tilde{\boldsymbol{x}}_B^\top\dot{\tilde{\boldsymbol{x}}}_B)^2}{||\tilde{\boldsymbol{x}}_B||^5}, && \text{(B.32)}
\end{aligned}$$

where $\ddot{\tilde{\boldsymbol{x}}}_B$ is given by

$$\ddot{\tilde{\boldsymbol{x}}}_B = -\ddot{\psi}(\boldsymbol{x}_\psi \times \boldsymbol{a}_T) - \dot{\psi}^2(\boldsymbol{y}_\psi \times \boldsymbol{a}_T) - 2\dot{\psi}(\boldsymbol{x}_\psi \times \boldsymbol{j}) + \boldsymbol{y}_\psi \times \boldsymbol{s}. \tag{B.33}$$

Now compute $\ddot{\boldsymbol{y}}_B$:

$$\begin{aligned}
\ddot{\boldsymbol{y}}_B &= \frac{1}{||\boldsymbol{a}_T||}\left[\boldsymbol{s}\cos\beta - (\ddot{\boldsymbol{x}}_B \times \boldsymbol{a}_T + 2\dot{\boldsymbol{x}}_B \times \boldsymbol{j} + \boldsymbol{x}_B \times \boldsymbol{s})\sin\beta\right] && \text{(B.34)}\\
&\quad - \frac{\boldsymbol{j}\cdot\boldsymbol{a}_T}{||\boldsymbol{a}_T||^3}\left[\boldsymbol{j}\cos\beta - (\dot{\boldsymbol{x}}_B \times \boldsymbol{a}_T + \boldsymbol{x}_B \times \boldsymbol{j})\sin\beta\right] && \text{(B.35)}\\
&\quad - \frac{\boldsymbol{s}\cdot\boldsymbol{a}_T + \boldsymbol{j}\cdot\boldsymbol{j}}{||\boldsymbol{a}_T||^2}\boldsymbol{y}_B + 2\frac{(\boldsymbol{j}\cdot\boldsymbol{a}_T)^2}{||\boldsymbol{a}_T||^4}\boldsymbol{y}_B - \frac{\boldsymbol{j}\cdot\boldsymbol{a}_T}{||\boldsymbol{a}_T||^2}\dot{\boldsymbol{y}}_B. && \text{(B.36)}
\end{aligned}$$

Finally $\ddot{\boldsymbol{z}}_B = \ddot{\boldsymbol{x}}_B \times \boldsymbol{y}_B + 2(\dot{\boldsymbol{x}}_B \times \dot{\boldsymbol{y}}_B) + \boldsymbol{x}_B \times \ddot{\boldsymbol{y}}_B$ and the angular acceleration can be obtained in closed form by differentiating the orientation dynamics twice and inverting:

$$\dot{\hat{\omega}} = \boldsymbol{R}^\top(\ddot{\boldsymbol{R}} - \boldsymbol{R}\hat{\omega}^2). \tag{B.37}$$

Inputs. We aim to obtain u_1, u_2, u_3, u_4, ρ as a function of the flat outputs and a finite number of their derivatives. First, use the Euler rotational equation (B.16), $\boldsymbol{\tau} = \boldsymbol{I}\dot{\omega} + \omega \times \boldsymbol{I}\omega$, to obtain an expression for $\boldsymbol{\tau}$ in terms of the flat outputs:

$$\begin{bmatrix} k_T(b + a\cos\varphi)u_2 \\ k_T(c\cos\varphi - k_M\sin\varphi)u_3 \\ k_T(c\sin\varphi + k_M\cos\varphi)u_4 \end{bmatrix} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}. \tag{B.38}$$

First use the τ_x equation combined with $u_2 = \frac{m\xi_2}{k_T \sin\varphi}$ to solve for φ:

$$\frac{\tau_x}{m\xi_2} = \frac{b + a\cos\varphi}{\sin\varphi}. \tag{B.39}$$

Let $\mu = \cos\varphi$. Then $\sin\varphi = \sqrt{1-\mu^2}$ and we can solve for μ, and consequently φ, algebraically:

$$\varphi = \arccos\left[\frac{-ba \pm |\eta|\sqrt{b^2 - a^2 + \eta^2}}{b^2 + \eta^2}\right], \tag{B.40}$$

where $\eta = \frac{\tau_x}{m\xi_2}$. The ambiguity between the two solutions for φ is resolved by using the constraint that $\varphi \in [0, \frac{\pi}{2}]$. If $b > a$, the case $b^2 - a^2 + \eta^2 < 0$ never occurs. Thus, the inputs are obtained as a function of the flat outputs as follows:

$$\varphi = \arccos\left[\frac{-ba \pm |\eta|\sqrt{b^2 - a^2 + \eta^2}}{b^2 + \eta^2}\right], \tag{B.41}$$

$$u_1 = \frac{m\xi_1}{k_T \cos\varphi}, \tag{B.42}$$

$$u_2 = \frac{m\xi_2}{k_T \sin\varphi}, \tag{B.43}$$

$$u_3 = \frac{\tau_y}{k_T(c\cos\varphi - k_M \sin\varphi)}, \tag{B.44}$$

$$u_4 = \frac{\tau_z}{k_T(c\sin\varphi + k_M \cos\varphi)}. \tag{B.45}$$

The only singularity in the above equations is at $\varphi = 0$ where u_2 and u_4 become undefined. It might be possible to create a continuous extension of the map by taking the limit $\varphi \longrightarrow 0$.

Conclusion. We have shown that the morphing quadcopter model is differentially flat with flat output $\boldsymbol{\sigma} = (x, y, z, \psi, \rho)$. $\square$

Validation

We verify the above proof numerically. To do this we implement the *forward map* $\bar{\sigma} = \mathcal{F}(x, u)$ as well as the *reverse map* $(x, u) = \mathcal{R}(\bar{\sigma})$. By verifying that,

$$\mathcal{R}(\mathcal{F}(x, u)) = (x, u), \tag{B.46}$$

the validity of the above computations is verified. The reverse map is given by the expressions in the above proof. The forward map is described below.

Forward map. The highest derivative of the position needed for flatness is the snap $\boldsymbol{s}$ which is the fifth derivative of the flat output $\boldsymbol{p}$. The highest derivative of the yaw angle ψ needed is $\ddot{\psi}$. Let $x \in \mathbb{R}^{12}$ be the state of the system. Then the forward map is given by:

$$\boldsymbol{p} = x(1:3) \tag{B.47}$$

$$\boldsymbol{v} = x(7:9) \tag{B.48}$$

$$\boldsymbol{a} = -g\boldsymbol{z}_W + \xi_1 \boldsymbol{z}_B + \xi_2 \boldsymbol{y}_B \tag{B.49}$$

$$\boldsymbol{j} = \xi_1 \boldsymbol{R}\hat{\omega}\boldsymbol{z}_B + \xi_2 \boldsymbol{R}\hat{\omega}\boldsymbol{y}_B \tag{B.50}$$

$$\boldsymbol{s} = \xi_1 \boldsymbol{R}\hat{\omega}^2\boldsymbol{z}_B + \xi_1 \boldsymbol{R}\dot{\hat{\omega}}\boldsymbol{z}_B \tag{B.51}$$

$$+ \xi_2 \boldsymbol{R}\hat{\omega}^2\boldsymbol{y}_B + \xi_2 \boldsymbol{R}\dot{\hat{\omega}}\boldsymbol{y}_B \tag{B.52}$$

$$\dot{\hat{\omega}} = \left[J^{-1}(\boldsymbol{\tau} - \omega \times J\omega)\right]_\times \tag{B.53}$$

$$\boldsymbol{\tau} = \begin{bmatrix} k_T(b + a\cos\varphi)u_2 \\ k_T(c\cos\varphi - k_M \sin\varphi)u_3 \\ k_T(c\sin\varphi + k_M\cos\varphi)u_4 \end{bmatrix}, \tag{B.54}$$

where we have assumed that the derivatives of all the inputs are zero. The yaw angle ψ requires two derivatives which are obtained by considering the mapping between angular velocity and the time derivative of the rotation parameterization:

$$\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \boldsymbol{E}^{-1}\boldsymbol{R}\omega. \tag{B.55}$$

Furthermore, define $\boldsymbol{\pi} = \boldsymbol{x}_W^\top \boldsymbol{E}^{-1}$ allowing us to obtain $\dot{\boldsymbol{\pi}}$ analytically. The expression of $\boldsymbol{E}^{-1}$ can found in Equation (C.2):

$$\psi = x(6) \tag{B.56}$$

$$\dot{\psi} = \boldsymbol{x}_W^\top \boldsymbol{E}^{-1}\boldsymbol{R}\omega \tag{B.57}$$

$$\ddot{\psi} = \dot{\boldsymbol{\pi}}\boldsymbol{R}\omega + \boldsymbol{\pi}\boldsymbol{R}\hat{\omega}\omega + \boldsymbol{\pi}\boldsymbol{R}\dot{\omega}. \tag{B.58}$$

We set the derivatives of the last flat output ρ to zero; constraining the curvature at the initial and final points of the trajectory:

$$\rho = \frac{u_1}{u_2}\cot\varphi \tag{B.59}$$

$$\dot{\rho} = 0 \tag{B.60}$$

$$\ddot{\rho} = 0. \tag{B.61}$$

Results. We test 10,000 pairs (x, u) sampled from $\mathcal{D}$. The position is sampled $\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{\omega} \sim \boldsymbol{\mathcal{U}(-10, 10)}$. The orientation is sampled as $\phi \sim \mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2})$, $\theta \sim \mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2})$, $\psi \sim \mathcal{U}(-\pi, \pi)$. The inputs are sampled $u_1 \sim \mathcal{U}(0, 10)$, $u_2, u_3, u_4 \sim \mathcal{U}(0, 10)$, and $\varphi \sim \mathcal{U}(\epsilon, \frac{\pi}{2} - \epsilon)$, where $\epsilon = 10^{-2}$. The maximum error is

$$\max_{(x,u)\sim\mathcal{D}} ||\mathcal{R}(\mathcal{F}(x, u)) - (x, u)||_\infty \approx 10^{-3}, \tag{B.62}$$

where the infinity norm is defined by $||x||_\infty = \max(|x_1|, ..., |x_n|)$.

Appendix C

ROTATIONS

This appendix summarizes the essential objects from rotation theory referenced throughout the main text. We parameterize the space of rotations using Euler ZYX angles. The rotation matrix between the base and world frame is given by

$$\boldsymbol{R} = \begin{bmatrix} c_\theta c_\psi & c_\psi s_\phi s_\theta - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\psi s_\theta \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\theta s_\psi & c_\phi s_\theta s_\psi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\phi c_\theta \end{bmatrix}, \tag{C.1}$$

where $c_\psi = \cos(\psi), s_\psi = \sin(\psi)$, $c_\theta = \cos(\theta), s_\theta = \sin(\theta)$, and $c_\phi = \cos(\phi), s_\phi = \sin(\phi)$. The transformation between angular velocity $\boldsymbol{\omega} = (p, q, r)^\top$ in the body frame and body rates $\dot{\boldsymbol{\theta}} = (\dot{\psi}, \dot{\theta}, \dot{\phi})^\top$ is given by $\dot{\boldsymbol{\theta}} = \boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{\omega}$. The transformation matrix is

$$\boldsymbol{E} = \begin{bmatrix} 0 & -\sin(\psi) & \cos(\theta)\cos(\psi) \\ 0 & \cos(\psi) & \cos(\theta)\sin(\psi) \\ 1 & 0 & -\sin(\theta) \end{bmatrix}, \tag{C.2}$$

and its inverse is given by

$$\boldsymbol{E}^{-1} = \begin{bmatrix} \frac{\cos\psi\ \sin\theta}{\cos\theta} & \frac{\sin\theta\ \sin\psi}{\cos\theta} & 1 \\ -\frac{\sin\psi}{\cos\theta} & \frac{\cos\psi}{\cos\theta} & 0 \\ \frac{\cos\psi}{\cos\theta} & \frac{\sin\psi}{\cos\theta} & 0 \end{bmatrix}. \tag{C.3}$$

Appendix D

INERTIAL PARAMETERS

Inertial Parameters

The inertias of each component were derived from the 3D computer-aided design model of the robot:

$$I_{\text{base}} = \begin{bmatrix} 0.67 & -0.003 & 0.05 \\ -0.003 & 1.1 & 0.05 \\ 0.05 & 0.05 & 0.88 \end{bmatrix} \times 10^{-2}, \tag{D.1}$$

$$I_{\text{arm}} = \begin{bmatrix} 0.88 & -0.0007 & -0.001 \\ -0.0007 & 3.7 & 0.06 \\ -0.001 & 0.06 & 0.88 \end{bmatrix} \times 10^{-2}, \tag{D.2}$$

$$I_{\text{rotor}} = \begin{bmatrix} 2.2 & 0 & 0 \\ 0 & 2.2 & 0 \\ 0 & 0 & 4.4 \end{bmatrix} \times 10^{-5}. \tag{D.3}$$

The masses, obtained by independently weighing each part, were

$$m_{\text{base}} = 2.33\text{kg} \tag{D.4}$$

$$m_{\text{arm}} = 1.537\text{kg} \tag{D.5}$$

$$m_{\text{rotor}} = 0.021\text{kg} \tag{D.6}$$

$$m_{\text{total}} = 5.49\text{kg}. \tag{D.7}$$