Leveraging Intentional Collisions to Enhance Aerial Manipulation

by

Mark Nail

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in the University of Michigan
2026

Doctoral Committee:

Professor Ella Atkins, Co-Chair
Professor Brent Gillespie, Co-Chair
Professor Dimitra Panagou
Professor Vasileios Tzoumas

Mark Nail
mrnail@umich.edu
ORCID iD: 0000-0002-5675-0332

© Mark Nail 2026

ACKNOWLEDGEMENTS

I would like to start by acknowledging and thanking Ella Atkins and Brent Gillespie for guiding and mentoring me throughout this journey. You have both been phenomenal—far better advisors than I could have hoped for. I've grown and learned so much from each of you, and you have played a vital role in shaping the engineer I am today. Additionally, I would like to thank Dimitra Panagou and Vasileios Tzoumas for being a part of my committee and helping to guide the final stages of my research. I would also like to thank Gabriel Arellano, who inspired the foundation of this dissertation.

I owe many thanks to my wonderful labmates in the Autonomous Aerospace Systems (A2Sys) Lab and the HaptiX Lab for supporting me in research and for being great friends throughout this Ph.D. From A2Sys, thank you to Brandon Apodaca, Prince Kuevor, Joseph Kim, Matthew Romano, Akshay Mathur, Paul Flanigan, and Prashin Sharma. You were fundamental in helping me dive into the world of unmanned aerial systems and in kickstarting my experimental testing. From HaptiX, thank you to Hannah Kuperus, Hannah Baez, Devon Kelly, Alia Gilbert (my first friend at Michigan), Arjun Chembai Ramachandran, and Zach Brei for your constant help and feedback. Though also labmates, I want to give an enormous thank you to Nicholas Jänne and Olivia Ma. I'm fortunate to say that you've gone from undergraduate students working with me to lifelong friends who helped me tremendously on this journey.

Outside of the lab, I have made more friendships than I could've imagined. While many of my labmates are also dear friends, I want to additionally thank Stanley Lewis, Trevor Kattenberg, Riley Pieper, Tribhi Kathuria, Seth Isaacson, Luis Hernan Cubillos, Sophia Cubillos-Carpentier, and Jeremy Davis for being there for me.

I've also been fortunate to work with amazing faculty and staff in the University of Michigan's Robotics Department, who have been nothing but a joy to work with and always willing to help. Specifically, thank you to Denise Edmund, Peter Gaskell, Damen Provost, Kimberly Johnson, and Samantha Price.

I would like to thank both the Rackham Merit Fellowship and the National Science Foundation Graduate Research Fellowship for funding my research throughout my Ph.D.

I would also like to thank my girlfriend, Alessandra Zieleniewski, and her family for being a second family to me in Michigan when I'm far from home.

Lastly, thank you to my twin brother, Greg Nail, and my parents, Sylvia Perez and Brian Nail, for your constant positive support. You've spent the better part of three decades helping me get to where I am now and be the best I can be.

I could not have done this without all of you, and this acknowledgments page isn't nearly enough to express my gratitude.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

LIST OF FIGURES vii

LIST OF TABLES xi

LIST OF ACRONYMS xii

ABSTRACT xiii

CHAPTER

1 Introduction 1

1.1 Motivation 1
1.2 Problem Statement 2
1.3 Research Approach and Dissertation Outline 3
1.4 Contributions and Innovations 4

2 Simplifying Aerial Manipulation Using Intentional Collisions 6

2.1 Introduction 6
2.2 Model Development 8
2.2.1 Flight Dynamics in 2-D 8
2.2.2 Flight Dynamics in 3-D 9
2.2.3 Collision Response 10
2.3 Hybrid System Controller Design 11
2.3.1 Velocity Matching 11
2.3.2 Hybrid System Integration 12
2.3.3 Thrust Limitation in Velocity Matching Mode 14
2.4 Flight Testing Methods 16
2.5 Results 17
2.5.1 Velocity Matching Performance 17
2.5.2 Flight Envelopes for Collision Approach 18
2.6 Discussion and Conclusions 19

3 Overcoming Underactuation in UAS by Switching Cascaded Controllers 21

3.1 Introduction 21
3.2 Model Development and System Architecture 22

3.2.1 Flight Dynamics in 2-D . . . 22
3.2.2 Cascaded Switching Controller . . . 23
3.2.3 Optimal Trajectory Solver . . . 25
3.3 Hexacopter Simulation Environment . . . 25
3.3.1 Simulation Architecture . . . 25
3.3.2 Flight Testing Methods . . . 26
3.4 Results . . . 26
3.4.1 Flight Sample Test Results . . . 26
3.4.2 Summary Results: Simulation and Experiments . . . 27
3.5 Discussions and Conclusions . . . 27
4 Shifting Underactuated Configuration Variables In Aerial Manipulation By Adding An Actuated Arm . . . 33
4.1 Introduction . . . 33
4.2 Model Development . . . 35
4.2.1 2-D Flight Dynamics for a Rigid Pogostick UAS . . . 35
4.2.2 2-D Flight Dynamics for an Actuated Pogostick UAS . . . 36
4.2.3 Collision Planning with a Pogostick . . . 37
4.2.4 Trajectory Planning . . . 38
4.2.5 Partial Feedback Linearization . . . 40
4.2.6 Zero Dynamics of Each System . . . 42
4.3 Simulation Results . . . 42
4.3.1 Simulation Environment . . . 42
4.3.2 Rigid Pogostick UAS Simulation Results . . . 44
4.3.3 Actuated Pogostick UAS Simulation Results . . . 44
4.4 Discussion and Conclusion . . . 46
5 Aerial Manipulation via Intentional Collisions with a Two Degree of Freedom Pogostick . . . 48
5.1 Introduction . . . 48
5.2 Physical Design . . . 50
5.3 System Dynamics . . . 52
5.3.1 Planar Hexacopter Dynamics . . . 52
5.3.2 Collision Response . . . 53
5.4 Hexacopter Controller Design . . . 55
5.4.1 Optimal Trajectory Planner . . . 56
5.4.2 Cascaded PID Control . . . 56
5.5 Pogostick Controller Design . . . 57
5.6 Combined System Design . . . 58
5.7 Experimental Methods . . . 59
5.8 Results . . . 59
5.9 Discussion and Conclusion . . . 63
6 Conclusion . . . 65

APPENDIX 68

BIBLIOGRAPHY 71

LIST OF FIGURES

FIGURE

1.1 This figure provides an overview of the technical work in this dissertation. Chapters 2 and 3 focus on developing flight controllers that enable interaction with a rigid pogostick payload (as illustrated), while Chapters 4 and 5 address the design of an actuated pogostick and its corresponding controller. These later chapters build on the earlier work, demonstrating improvements in both design and capabilities (as shown in the transitional arrow). 3

2.1 A) 2-D model of UAS in flight experiencing a horizontal disturbance. B) 2-D model of UAS in one-legged perch experiencing a horizontal disturbance. 8
2.2 A) Free Body and Configuration Diagram of 2-D UAS in Flight. B) Dimensions of Collision Contact from UAS Center of Mass and Ground Contact Normal . . 10
2.3 Cascaded control loop for the UAS flight controller managing all flight control tasks other than Velocity Matching. 12
2.4 Inner loop attitude controller that is solely used in the Velocity Matching flight mode. 12
2.5 Hybrid control model for UAS using physical parameters as guard conditions . 14
2.6 Modified DJI F550 Hexacopter (550 mm wheelbase) that is acting as the UAS for boucing interactions. The attached pogo stick was used to facilitate collision. 16
2.7 A) Visualization of UAS collision with $p_{z|initial} = 1.5$m, $\dot{p}_{x|initial} = 1.8$m/s. B) Performance visualization of pitch in Velocity Matching controller over the same time period. Asterisk demarcates where the Velocity Matching controller is toggled off and a zero pitch setpoint is being commanded after a bounce has occured. 17
2.8 Two-dimensional flight envelope mapping initial altitudes $p_{z|initial}$ and horizontal velocities $\dot{p}_{x|initial}$ to successful collisions. The area shaded in yellow and grey represents successful collisions, while the area shaded in orange are failed collisions in simulation. The green dots and red x's represent successful and unsuccessful experimental flights that each included a collision, which form the grey shaded area. The unshaded region represents altitudes that were too low to obtain consistent collision data, due to the length of the pogo stick. 18

3.1 2-D UAS Flight Dynamics 23

3.2 Cascaded switching control. The offline trajectory planner feeds waypoints into either the position+altitude controller pair or the attitude+altitude controller pair. The red toggle shows where the switching controller can bypass position control, sending the planner's commands directly to the attitude loop. Let $q(t_p)_k$ denote the generated trajectory and $q(t)_k$ a single waypoint provided to the controller. The notation $q()_k$ denotes which configuration variable is being traced, horizontal position q_1, altitude q_2, and pitch q_3. 24
3.3 Example of a poor switching time (60%) for initial $q_1 = -0.3$ m, from experimental data. The dotted red line denotes the switch from position–altitude to attitude–altitude cascaded control. Goal states are denoted by a yellow star. . . 28
3.4 Example of a good switching time (90%) for initial $q_1 = -0.3$ m, from experimental data. The dotted red line denotes the switch from position–altitude to attitude–altitude cascaded control. Goal states are denoted by a yellow star. . . 29
3.5 Experimentally determined best switching times overlaid on simulation-predicted best switching times. Switching time is the percentage of the flight at which position control is toggled off. Light shaded blue region shows were switching time was irrelevant and errors for all switch times were low in simulation and experiments. 30
3.6 Experimental errors at the best switching times overlaid on simulated errors at the predicted best switching times. Errors are the sum of lateral errors in the x–y plane and angular error from vertical. 31

4.1 A) 2-D model of a Rigid Pogostick UAS in flight with body force $\mathbf{f_1}$ and torque $\boldsymbol{\tau_1}$ from the propellers. B) 2-D model of an Actuated Pogostick UAS in flight with body force $\mathbf{f_1}$ and torque $\boldsymbol{\tau_1}$ from the propellers. Torque $\boldsymbol{\tau_2}$ comes from a motor between UAS Body D and pogostick PS. C) Collision Response Model of an Actuated Pogostick UAS with rigid body D, point-mass p_c, contact angle $q_3 + q_4$, velocity for the rigid body $\mathbf{v}$, pogostick length l, and contact impulse $\boldsymbol{\Lambda}$. 35
4.2 Top: Planner and run-time controller (based on partial feedback linearization around $\{q_2,\ q_3\}$) for the RP-UAS (UAS without additional actuator). Bottom: Planner and run-time controller (based on partial feedback linearization around $\{q_1,\ q_2,\ q_4\}$) for the AP-UAS (UAS with actuated pogostick). User-defined initial states q_i and final pre-contact states q_f are fed into the planner, $q(t_p)$ is an output trajectory from the planner (available for all states), $q(t)$ is the reference being tracked by the run-time controller, and q_o are the system output states. Only a subset of the planned states are sampled and tracked by the run-time controller, as indicated by the subscripts. The state variables planned but not tracked are indicated with a pigtail arrow. 39
4.3 Frames from simulation animation for AP-UAS. Red Arrow depicts the velocity vector of the UAS. 41
4.4 RP-UAS partial feedback linearizing controller performance. A) No initial state errors, B) Initial error of 0.25 m in q_1, and C) Initial error of 0.25 m in q_1 and q_2 and error of -0.17533 rad in q_3. 43

4.5 AP-UAS partial feedback linearizing controller performance. A) No initial state errors and B) Initial error of 0.25 m in q_1 and q_2 and error of -0.17533 rad in q_3 and q_4. 45
4.6 Plot of final error in q_i at strike based on initial error in q_i ($i = 1, ..., 4$) at the start of the trajectory. Each state was tested with an initial error while the other states started with zero error. 46

5.1 Image of cable-driven pogostick payload, the UAS in flight, and the landing platform used for testing. The pogostick and cable tensioners are labelled. Three cable tensioners provide the two degrees of freedom by allowing the pogostick to slide along the hemispherical base. 51
5.2 A and B) Collision using an actuated pogostick where the center of rotation is not aligned with the CoM of the UAS. The varying distance l from the point of contact p_c to the UAS CoM depends on pitch q_3 and strike angle q_4, which complicates collision planning at a targeted strike location (q_1, q_2). C) Collision using an actuated pogostick where the center of rotation is aligned with the UAS CoM. In this case, l remains constant, which simplifies collision planning. 51
5.3 2-D free body diagram of the proposed hexacopter system. 53
5.4 2-D free body diagram of collision response. 54
5.5 Block diagram outlining the hexacopter portion of planning and control. The user specifies a final strike location and desired strike velocities. The planner then creates a dynamically feasible trajectory from the initial UAS location to the target strike location using optimization. This trajectory is then fed into a cascaded PID controller to command position commands into motor thrusts. . . 55
5.6 Geodesic-based control of the cable-driven pogostick. A) the pogostick is shown in blue, cables in purple. B) the yellow star marks the desired goal orientation, and the green dashed curves show the geodesic cable paths that align the pogostick with the new goal state. C) the cables lie along the geodesic curves intersecting at the goal position. 57
5.7 Block diagram showing how collision response planning influences trajectory generation and how both outputs feed into the pogostick and UAS controllers. Trajectory generation starts with a desired strike velocity vector $\dot{q}_f$, which is then fed into a collision angle solver to determine the appropriate strike angle. After the desired collision angle is found, that angle as well as the desired pre-impact states are fed into the UAS Collision Trajectory Planner, where a dynamically feasible trajectory is created to get to the desired states. 58
5.8 Results from hover testing while the pogostick is moving. The plots show UAS x, y, and z position and pogostick tilt and azimuth angles around the hemisphere. The pink overlayed boxes denote when the pogostick is moving. 60
5.9 Motor commands during hover testing with and without pogostick motion. The pink overlayed boxes denote when the pogostick is moving. 61
5.10 Sample results from 25 trajectory tracking flights with and without pogostick actuation. 63

A.1 Initial design of a bear-trap-inspired collision-based cutting and grasping mechanism. Images show all sides of the mechanism. 69
A.2 Visualization of mechanism triggering and cutting. 70
A.3 Visualization of torsion springs and arm retention linkage. 70

LIST OF TABLES

TABLE

3.1 Switching Controller: Trajectory Generation Costs 25
3.2 Switching Controller: Trajectory Generation Constraints 25

4.1 Trajectory Generation Costs 40
4.2 Trajectory Generation Constraints 40

5.1 Trajectory Generation Costs 56
5.2 Trajectory Generation Constraints 56
5.3 Cable-driven pogostick specifications 59
5.4 Averaged Root Mean Square Error for 25 Flights with and without Pogostick Actuation 60
5.5 Pointing Accuracy at Strike 62

LIST OF ACRONYMS

UAS Uncrewed Aircraft System

RP-UAS Rigid Pogostick Uncrewed Aircraft System

AP-UAS Actuated Pogostick Uncrewed Aircraft System

ABSTRACT

Uncrewed Aircraft Systems (UAS) have become essential in tasks that are too difficult or dangerous for humans. This dissertation was initially inspired by the challenge of making rainforest tree sampling safer for botanists, who often need to climb dangerously tall trees. Building on this motivation, we developed a new approach for aerial manipulation that enables interaction with objects of unknown dynamics through intentional collisions using a pogo-stick payload. The key insight is that intentional collisions consolidate all potential disturbance forces into a single impulse, the outcome of which is hopefully simpler to predict. Given assumptions on the dynamics of the interaction object, the map from pre- to post-impact states (collision response) is known, providing an opportunity to choose pre-impact states that will result in recoverable post-impact states.

The use of intentional collisions with a UAS presents unique challenges due to underactuation. Collision outcomes are set by strike velocity and orientation, yet hitting a desired point requires full position control. Traditional UAS actuator configurations cannot provide both full position and attitude control authority. This dissertation presents several approaches to mitigate underactuation and enable aerial manipulation with intentional collisions.

To begin, this dissertation develops the strategy of using intentional collisions in a flight mode called Velocity Matching. The Velocity Matching flight mode starts in two phases: launch phase and then the Velocity Matching phase. During the launch phase, the UAS is sped up horizontally to "launch" the vehicle, assumed thereafter to behave ballistically, towards the target. After launch, the Velocity Matching phase is entered, and thrust is limited to the bare minimum thrust required for attitude control. During the Velocity Matching phase the minimal thrust attitude control is used to align the center-of-mass velocity vector of the UAS with the pogo-stick's orientation at strike. Using simulation and flight tests, we generate a map from pre-launch modes states into recoverable post-impact states, allowing for planned strikes.

Second, we refine the flight control from Velocity Matching to a cascaded position–attitude switching controller (called Switching Control). In the Switching Control mode, the UAS flies in position cascaded PID control for most of the trajectory, then switches at the last moment to attitude cascaded PID control for pre-strike alignment. We also provide a tool to

predict the best switching time from an initial displacement, and we validate the approach experimentally.

Finally, we develop a new control approach involving an actuated pogo-stick payload controlled via a partial feedback linearizing controller to regulate both UAS position and pogo-stick orientation. After validating the actuated pogo-stick in simulation, we built a physical cable-driven version that slides along a hemispherical base. This mechanism provides two-degree-of-freedom orientation control (pitch and roll compensation) and keeps the pogo-stick perpendicular to the UAS center of mass, enabling more favorable, recoverable collision responses.

CHAPTER 1

Introduction

1.1 Motivation

Autonomous vehicles have taken a foothold in modern technology and can be seen all over the world tackling anything from warehouse management to autonomously classifying the seafloor [13, 33]. One goal of these autonomous systems is to keep humans out of harm's way, or at least mitigate risk when tasks go wrong. Uncrewed aircraft systems (UAS) are a subset of these vehicles and will be the platform of focus for this dissertation.

The specific motivation behind this research is to make rainforest tree sampling a safer task by using a UAS [32]. Current practices, including climbing rainforest trees or launching weighted projectiles to snag branches, are unreliable and often unsafe, especially when access routes are uncertain and branch mechanics are unpredictable. Enabling a UAS to interact with tree branches poses an interesting question: How can a UAS safely interact with an object about which it knows little to nothing regarding the object's dynamics? The default intuition to tree branch interactions seems simple: perch like a bird, which has been done by several UAS researchers [29] but is primarily done with static objects with known geometry like pipes. While perching works when the object the UAS is interacting with has known dynamics, it does not necessarily extend to unknown objects (like light, flexible tree branches). Due to stereotypical multirotor UAS being underactuated (position and attitude control are coupled), disturbance rejection of unknown forces mainly relies on changes in attitude. Because perching and many other prolonged forms of aerial manipulation have horizontal position kinematically tied to attitude, any force applied at the point of contact becomes a moment that must be balanced by actions of the motorized propellers, making disturbance rejection difficult. This becomes more and more true the longer the interaction persists. One way to solve this problem is to make interactions near-instantaneous, meaning that all possible disturbances act as a single impulse whose follow-on consequences can then be corrected for in subsequent flight when the UAS has maximal control authority rather

than while still interacting, when interaction forces not directed through the center of mass carry moments. This dissertation focuses on how UAS can leverage intentional collisions for aerial manipulation that takes place during very short (analyzed as instantenous) periods of time, allowing interaction with unknown objects while minimizing complexity of interaction during the time of contact.

Aerial manipulation, simply put, is the interaction between a flying vehicle and its environment [34]. This domain spans a wide range of tasks, but most aerial manipulation efforts to date fall into three modes: perching [35], grasping [27], and pushing [50]. All of these modes of interaction (aside from grasping freely moving objects) require prolonged contact with the environment. The method introduced here (using intentional collisions to strike, interact, and recover) enables interaction so long as the contribution of the object dynamics to the collision response are known or assumed. Because the strike location, orientation, and speed can be planned, the collision response can be accounted for in the plan to make recovery easier. This is similar to, and partially inspired by, the early work of Mark Raibert and his hopping robots that use planned foot placement for stabilization and navigation [58].

The most challenging part of planning and executing these intentional collisions lies in flight control. Because UAS are underactuated systems, simultaneous full control over position and attitude is not possible [53]. As such, this dissertation presents several control methods and payload designs that work around the constraints of the underactuated UAS, adding intentional collisions to the aerial manipulation toolbox.

1.2 Problem Statement

This dissertation addresses the following challenges:

1. Aerial manipulation with objects of unknown dynamics introduces uncertainty in the interaction between the UAS and the environment, causing an inability to guarantee safety during interaction. How can this be resolved? We propose the use of intentional collisions to enable a new method of near-instantaneous interaction, limiting the risks associated with prolonged contact with unknown objects.

2. Underactuated systems pose challenges to full system control, both attitude and position cannot be controlled simultaneously. Given that the collision response for a rigid object [47] is a function of angular velocity, linear velocity, and the contact angle (angle from the strike surface normal to the vector between the contact point and the center of mass of the UAS), both position and attitude control are needed to plan and complete an intentional collision on a target. This dissertation explores several con-

trol strategies to compensate for a UAS's underactuated nature so that collision-based aerial manipulation can be successful.

3. Interaction via intentional collisions between the UAS and its target require new payloads to allow for quick, predictable collisions with the UAS. With predictable collisions, the UAS post-collision states can be planned such that the UAS can recover and hover after strike. These post-collision, recoverable states form what this dissertation calls a basin of recovery. This dissertation provides two such payloads: one rigid pogo stick and one actuated pogo stick.

1.3 Research Approach and Dissertation Outline

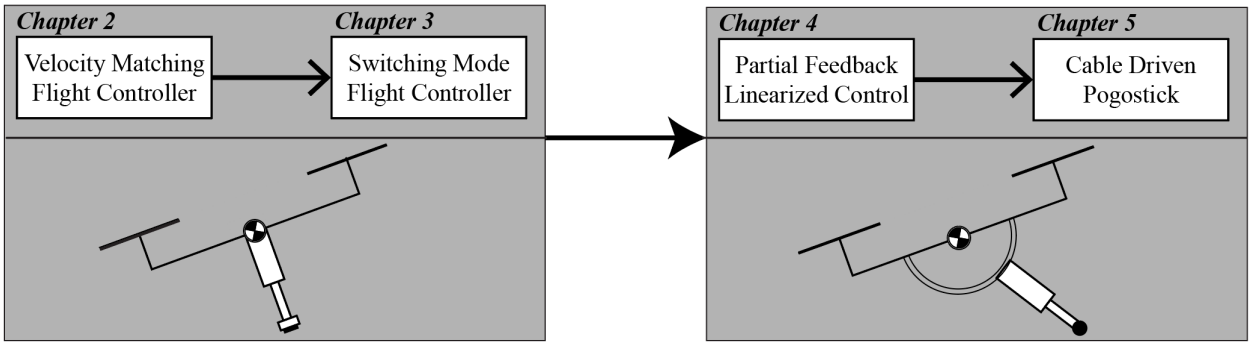

Figure 1.1: This figure provides an overview of the technical work in this dissertation. Chapters 2 and 3 focus on developing flight controllers that enable interaction with a rigid pogo-stick payload (as illustrated), while Chapters 4 and 5 address the design of an actuated pogostick and its corresponding controller. These later chapters build on the earlier work, demonstrating improvements in both design and capabilities (as shown in the transitional arrow).

An outline for the work completed in this dissertation is presented in Figure 1.1. A common theme throughout this dissertation is the tight loop between simulation and experiment: simulation, modeling, and analysis guide design decisions, and experiments validate (and refine) those insights.

Chapter 2 introduces the core idea of intentional collisions. It presents a simplified approach (Velocity Matching) that ensures the rigid pogo-stick payload makes first contact with the environment while deferring analysis and planning around the collision response. Both simulation and experiments are used to validate this approach.

Chapter 3 builds on Chapter 2 by proposing a more capable flight-control strategy that enables more intelligently planned collisions with the rigid pogo-stick. The controller switches

between cascaded position control and cascaded attitude control while tracking an offline-generated optimal trajectory. Having the trajectories generated offline allows for minimal required computational power on the UAS, and simplified control architecture. Simulation and experimental results show that the switching controller reduces combined position error and angular error at the moment of strike.

Chapter 4 replaces the rigid pogo-stick with an actuated pogo-stick to improve control authority so collisions can be planned. These collisions are designed to arrest post-collision angular velocity, ensuring that the UAS lands within the basin of recovery after strike. A partial feedback-linearized controller commands the actuated pogo-stick to continuously correct the UAS attitude required for position control. Simulation studies demonstrate that this control method, paired with the theoretical actuated pogo-stick, achieves the desired strike position and orientation for safer collisions.

Chapter 5 continues the direction of Chapter 4.. It realizes a hardware design for an actuated pogo-stick that enables collisions directed through the UAS center of mass while maintaining full control over pogo-stick orientation. This is achieved with a cable-driven mechanism which allows for independent orientation control.

Chapter 6 explains the major contributions from this dissertation and provides ideas for work to follow this research.

1.4 Contributions and Innovations

The specific contributions of this dissertation are:

- Design and implementation of both rigid and actuated payloads to enable aerial manipulation via intentional collisions.
- Contributions to the open-source autopilot software (`rc_pilot_a2sys`) used by the A2Sys (Autonomous Aerospace Systems) Lab at the University of Michigan/Virginia Tech, enabling several new flight-control modes centered on achieving a state (position and attitude) at an anticipated point in time.
- Development of a UAS simulation environment for interaction planning, controller testing, and performance evaluation of the the physical UAS.
- Experimental validation of each proposed flight-control method to enable collision-based aerial manipulation.

The specific innovations of this dissertation are:

- Use of intentional collisions in UAS outside locomotion and energy-saving contexts.
- A new position–attitude switching control strategy on a UAS that minimizes position error and angular error of the system at strike, even though it is underactuated, while not controlled, velocity and angular velocity errors at strike are modest.
- A partial feedback-linearized controller for control of a (torque-sourced) actuated pogostick, allowing for theoretical orientation control at strike to be deferred to the actuated pogostick.
- A geodesic-controller-based approach for a cable-driven (position-sourced) actuated pogostick, allowing for actual orientation control at strike to be deferred to the actuated pogostick.

CHAPTER 2

Simplifying Aerial Manipulation Using Intentional Collisions

2.1 Introduction

Uncrewed aircraft systems (UAS) are increasingly deployed to complete tasks that are out of reach or dangerous for humans to complete. UAS have been typically deployed for monitoring, surveillance, and sensing tasks [77]. UAS have also been developed to physically interact with their environment under three strategies. The first physical interaction strategy involves deploying payloads in flight, avoiding interaction forces from the environment, e.g., dropping ignition spheres for prescribed burns [11]. The second strategy involves landing or perching prior to interaction to gain a stable base of support for more complicated tasks, e.g., deploying sensors underground [70]. The third strategy involves physical interaction with the environment while in flight, e.g., nailing shingles to a roof using a UAS as presented in [64]. The term aerial manipulation is reserved for in-flight physical interaction, the third strategy named above [65].

We aim to apply aerial manipulation for sampling leaves and small branches from within a dense tree canopy to improve the unreliable and often unsafe methods used to study the health of rainforests, one of the most bio-diverse ecosystems on the planet [49] [10]. UAS have been previously developed for tree sampling by the teams of Charron [18] and Krisanski [36]. The DeLeaves platform [18] performs well in forests without subcanopies, where the majority of trees can be sampled from above (e.g. evergreen forests). The UAS design documented in [36] approaches a tree for sampling from the side, allowing it to collect samples at any altitude. However, challenges arise for both systems in the rainforest, where density and stratification of vegetation make both navigation and manipulation difficult.

The simplest forms of aerial manipulation use a non-actuated, rigidly attached limb for UAS-environment interactions, similar to the "boomcopter" presented in [40]. This type of contact through a rigid limb is similar to leaning against a wall with one's arm. While rigid

limb contact can provide benefits to a UAS by transferring force onto a rigid support, it compromises disturbance rejection and attitude control because of constraints between the UAS and the environment.

More complex aerial manipulation systems use actuated arms to combat disturbances as reviewed in [34]. Robotic arms make increasingly complicated tasks feasible for aerial manipulation, including opening doors [38] or picking up items while on the move [74]. However, increased actuation comes at the cost of overall system weight, energy demand, and changing flight dynamics that arise from changes in arm configuration [34].

Approaches that take inspiration from nature hold significant promise, allowing for perching and grasping to be completed with the same limb [29]. For interacting with stable platforms like industrial piping [60], perching works well and adds stability. On the other hand, the unpredictability of a tree branch's mechanical behavior makes perching in the rainforest a difficult task. Additionally, attitude control and disturbance rejection can be compromised during perching interactions. We hypothesize that minimizing overall interaction time will maximize successful flights of the UAS, thus minimizing risks induced by UAS-environment interactions.

Raibert-style, single-legged hopping robots locomote through short "hopping" interactions between the robot's foot and the ground [58]. The hopping strategy minimizes interaction time and manages each hop's impulse, magnitude, and direction to maintain stability. Control derives from the use of intelligent foot placement. Unlike conventional multi-jointed walking robots, the hopping motion in Raibert's robots allows for simpler mechanical limbs and simpler control [30]. We take inspiration from these hopping robots, equipping our UAS with a pogo stick for short, predictable UAS-environment interactions. A pogo stick-equipped UAS was proposed in [79], although their approach used vertical bouncing as a more energy efficient substitute to hovering, as opposed to bouncing as a means of aerial manipulation.

This paper extends bouncing robots into the domain of aerial manipulation, adopting an attitude-only controller for foot placement, and proposing flight envelopes to manage the approach to collision in a manner that assures the recovery of hover after collision. This paper contributes the two-dimensional flight and perching model that drives our collision strategy (2.2), the attitude-only controller that enables intentional and recoverable collision (2.3), the design of a python-based simulator that evaluates the collision strategy (2.4), and simulation and experimental flight envelopes for velocity mode safety checking (2.5).

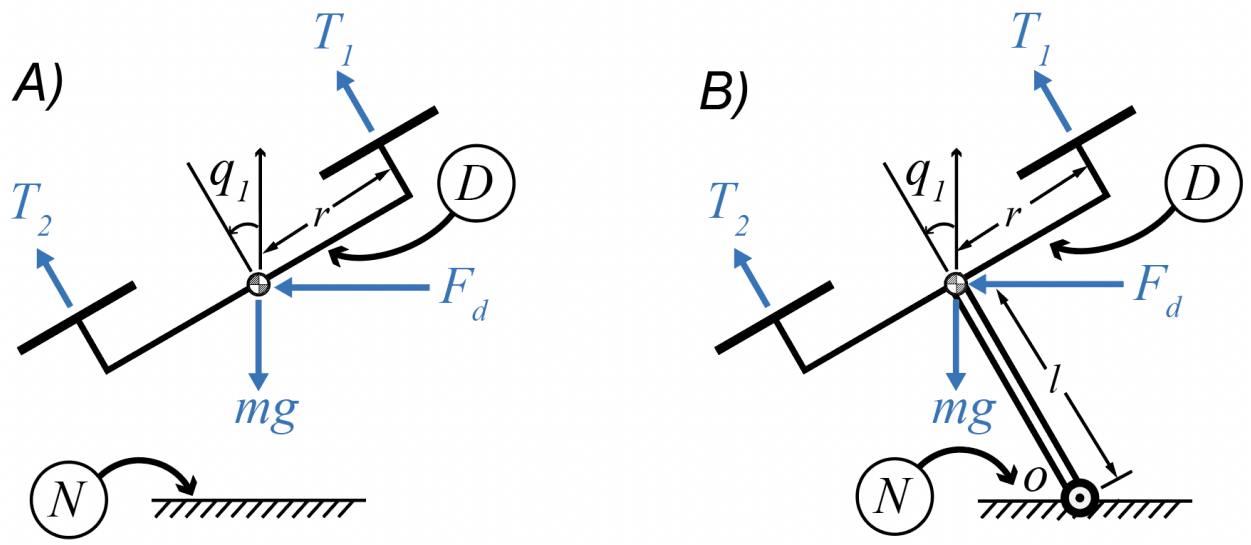

Figure 2.1: A) 2-D model of UAS in flight experiencing a horizontal disturbance. B) 2-D model of UAS in one-legged perch experiencing a horizontal disturbance.

2.2 Model Development

2.2.1 Flight Dynamics in 2-D

To quantify any compromise in the ability of a perched UAS to reject disturbances, we develop simple 2-D models of a UAS either in flight or perched with a single leg on fixed ground. Figure 2.1A shows a planar model of a UAS in flight while Fig. 2.1B shows the same UAS perched with a single central leg on the ground.

Both systems are acted upon by a horizontal disturbance force F_d, a vertical force of gravity mg, and two thrusts T_1 and T_2 fixed in the UAS frame. Authority over the pitch angle q_1 is available from the thrust difference $T_{dif} = T_1 - T_2$, while the total thrust $T_{tot} = T_1 + T_2$ drives changes in positions p_x, p_z, the following comes from a simplification of the 3-D dynamics presented in [6].

A sum of forces and sum of moments acting on the UAS in flight can be used to produce the equations of motion that govern flight

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{p}_z \\ \ddot{p}_x \end{bmatrix} = \begin{bmatrix} r\frac{T_{dif}}{J} \\ g - \frac{T_{tot}}{m}\cos q_1 \\ \frac{T_{tot}}{m}\sin q_1 - \frac{F_d}{m} \end{bmatrix}, \tag{2.1}$$

where J is the moment of inertia, m is the UAS mass, and g is acceleration due to gravity.

In the 2-D flight case, the UAS can pitch toward the disturbance force F_d and maintain

its position. Authority over pitch acceleration $\ddot{q}_1$ is available using T_{dif} while unimpeded by F_d. In turn, the disturbance F_d can be counteracted using a component of T_{tot} even while maintaining null horizontal acceleration $\ddot{p}_x$.

The situation is different while perching. The connection at the point of contact O between the UAS and ground constrains the UAS position to a function of pitch q_1 and leg length l.

$$\begin{bmatrix} p_x \\ p_z \end{bmatrix} = \begin{bmatrix} l \sin q_1 \\ l \cos q_1 \end{bmatrix}. \tag{2.2}$$

A sum of forces and sum of moments acting on the UAS along with a consideration of the constraint Eqs. 2.2 can be used to produce the equation of motion that governs pitch during perch

$$\ddot{q_1} = \frac{F_d l \cos q_1 + mgl \sin q_1 + rT_{dif} + M_o}{J}. \tag{2.3}$$

During perch, all forces acting on the UAS contribute to the moment necessary to maintain a pitch angle. Due to the direct coupling between pitch q_1 and position p_x, p_y, disturbance rejection when coupled to the environment is less direct than in the flight case. Motor forces must generate moments for rejection, instead of directly using their forces as restoring forces. This complication in disturbance rejection is fundamental in motivating this paper's collision approach to aerial manipulation. By limiting total interaction time to be near instantaneous during a collision, disturbance rejection can be undertaken during flight.

2.2.2 Flight Dynamics in 3-D

For the 3-D case, we consider the full dynamics of the hexacopter platform. As presented in [6] the following set of equations govern the accelerations $\ddot{p}_x$, $\ddot{p}_y$, $\ddot{p}_z$ in the global frame, with I_{xx}, I_{yy}, I_{zz} as central moments of inertia with respect to the body axes, with ϕ, q_1, and ψ as roll, pitch, and yaw, respectively, and with α, β, γ as the angular velocities with respect to the body axes.

$$\begin{bmatrix} \ddot{p}_x \\ \ddot{p}_y \\ \ddot{p}_z \end{bmatrix} = \begin{bmatrix} T(\cos\psi \sin q_1 \cos\phi + \sin\psi \sin\phi)/m \\ T(\sin\psi \sin q_1 \cos\theta - \cos\psi \sin\theta)/m \\ g - T\cos\theta\cos\phi/m \end{bmatrix}, \tag{2.4}$$

with the following angular rates and accelerations:

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin q_1 \\ 0 & \cos\phi & \cos\theta\cos\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{q_1} \\ \dot{\psi} \end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} (I_{yy} - I_{zz})\beta\gamma/I_{xx} \\ (I_{zz} - I_{xx})\alpha\gamma/I_{yy} \\ (I_{xx} - I_{yy})\gamma\beta/I_{zz} \end{bmatrix} + \begin{bmatrix} \tau_\phi/I_{xx} \\ \tau_\theta/I_{yy} \\ \tau_\psi/I_{zz} \end{bmatrix}. \tag{2.6}$$

It is apparent from these equations that not only pitch q_1 but also roll ϕ and yaw ψ angles contribute to changes in world-frame position p_x and velocity $\dot{p}_x$, whereas only pitch contributes in the 2-D flight dynamics (see Eq. 2.1). However, we are leveraging our cascaded PID controller (see Fig. 2.3 below) to maintain 0° pitch and yaw throughout flight tests.

2.2.3 Collision Response

For modeling collisions in simulation, we have chosen the rigid contact model described in [26]. This rigid body collision response model offers predictions that match experiment so long as backspin is not present [26]. Although the pogo stick mounted to our UAS includes a spring, we assume a rigid pogo stick in the model used for collision response, controller design, and simulation. To compute collision response, we adopt the linear complementary problem solver outlined in [69].

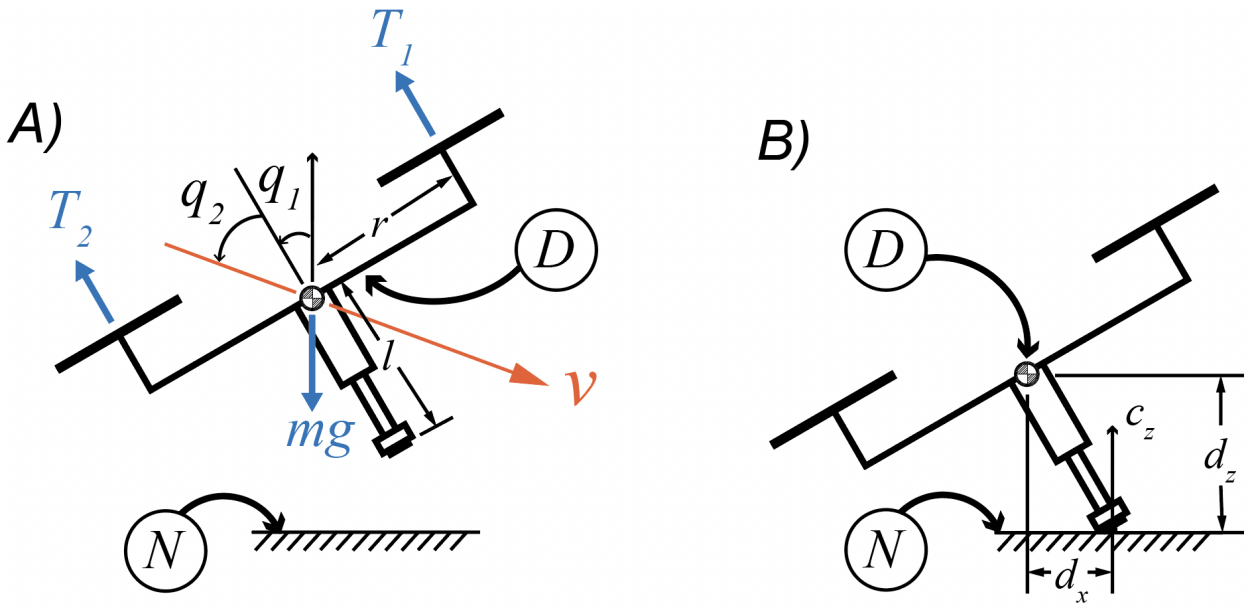

Figure 2.2: A) Free Body and Configuration Diagram of 2-D UAS in Flight. B) Dimensions of Collision Contact from UAS Center of Mass and Ground Contact Normal

At the instant of collision, it is necessary to establish the point at which contact is made and its relationship to the UAS center of mass. This mapping is visualized in Fig. 2.2B, where d_x and d_z are the horizontal and vertical distances from the UAS center of mass to

the point of contact, respectively, and c_x (not shown as it is zero on flat horizontal ground) and c_z are the horizontal and vertical contact normal components of the contact surface, respectively. With this information, a contact Jacobian J_c consisting of tangential J_t and normal J_n components can be constructed according to the following formulation:

$$J_c = \begin{bmatrix} J_t & J_n \end{bmatrix} = \begin{bmatrix} c_z & c_x \\ -c_x & c_z \\ -d_x c_x - d_z c_z & d_x c_z - d_z c_x \end{bmatrix} \tag{2.7}$$

At the instant of contact, the total forces acting on the UAS are

$$F_e + J_c F_c = Ma, \tag{2.8}$$

where $a = [\ddot{p}_x, \ddot{p}_z, \ddot{q}_1]^T$, M is the UAS inertia matrix, F_e are the external forces in the world frame and F_c is the contact force in the contact surface frame. The contact force is broken into normal and tangential components that are found in the z column matrix after the formulation of the linear complementary problem is solved as follows:

$$Vz + p \geq [0] \tag{2.9}$$

$$z \geq [0], \tag{2.10}$$

where the matrices V and p are defined as outlined in [69].

2.3 Hybrid System Controller Design

2.3.1 Velocity Matching

Velocity Matching is our method of foot placement. The appeal of this Velocity Matching mode is that it can run blind of altitude and position, only running with attitude control as shown in the block diagram in Fig. 2.4. By aligning the axis of the pogo stick with the velocity vector of the UAS, Velocity Matching ensures that the tip of the pogo stick is the first point of contact the system has with the environment. This allows for easier recovery, as the propellers are protected from striking the ground during a collision. Velocity Matching also minimizes the moments of any contact impulses about the UAS center of mass. The impulse that comes from the collision even acts as a restoring force, driving the UAS back away from the ground. The angle of the realtime velocity vector $q_3 = q_1 + q_2$ is used as the setpoint for the attitude controller such that the axis of the UAS pogo stick aligns with the velocity of the UAS center of mass. In effect, q_2 is driven to zero (see Fig. 2.2A). The value

of q_2 is calculated from the velocity of the UAS center of mass.

During Velocity Matching mode, the maximum pitch and roll of the UAS was set to 45° and the maximum angular velocity was set to 4 rad/s to prevent UAS tip-over and propeller strike.

For the 2-D simulation case, velocities are given in the world frame and pitch setpoint q_3 is calculated as follows:

$$q_3 = q_1 = \frac{\pi}{2} - \arctan\frac{\dot{p}_z}{\dot{p}_x} \tag{2.11}$$

For the 3-D experimental case, the additional roll setpoint $\phi_{desired}$ is calculated as follows:

$$\phi_{desired} = \arctan\frac{\dot{p}_z}{\dot{p}_y} - \frac{\pi}{2} \tag{2.12}$$

2.3.2 Hybrid System Integration

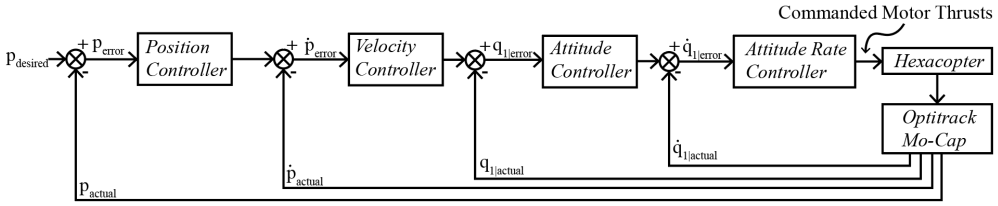

Figure 2.3: Cascaded control loop for the UAS flight controller managing all flight control tasks other than Velocity Matching.

Our UAS operates in one of two distinct flight control modes: Conventional Waypoint Following and Velocity Matching Control. From the Conventional Waypoint Control mode, there are three possible transitions: Approach Initiation, Envelope Violation, and Recovery. The mode switching is described in a Hybrid Automaton Diagram in Fig. 2.5.

When in Conventional Waypoint Following, time-dependent 3-D waypoints w_{conv} are used for guidance and navigation and are generally defined with 3-D position and yaw setpoints associated with specific times, as outlined in [25].

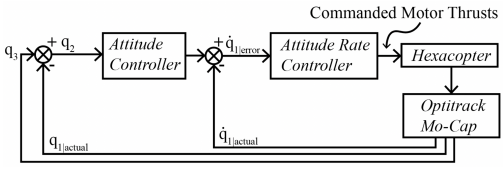

Figure 2.4: Inner loop attitude controller that is solely used in the Velocity Matching flight mode.

$$w_{conv} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ \psi \end{bmatrix} \tag{2.13}$$

To begin, a manual trigger sends the UAS into the Approach Initiation transition with a special waypoint w_{ai}, which sets a predetermined target velocity $\dot{p}_{x|t}$. The UAS uses feedback rate control to achieve the desired $\dot{x}$ while maintaining $\dot{p_y} = \dot{p_z} = 0$.

If at any point the $\dot{p}_x$ is within $\dot{p}_{x|t} \pm 0.1$m/s, then the Approach Initiation transition is triggered and the UAS is in Velocity Matching Mode, see (Fig. 2.5)

Waypoints w_{ai} in the Approach Initiation transition are time independent, only focusing on hitting a target horizontal velocity $\dot{p}_{x|t}$ and maintaining a target altitude $p_{z|t}$, as seen in the following formulation:

$$w_{ai} = \begin{bmatrix} p_y = 0 \\ p_z = p_{z|t} \\ \dot{p}_x = \dot{p}_{x|t} \\ \dot{p}_y = 0 \\ \dot{p}_z = 0 \\ \psi = 0 \end{bmatrix} \tag{2.14}$$

All other waypoints in Conventional Waypoint Following mode use the normal standard of time-based position and yaw waypoints.

In Velocity Matching mode, a collision is determined to be complete if the controller detects $\dot{p_z} = 0$ (after a period of $\dot{p_z} < 0$), at which point the UAS goes through a recovery transition to the Conventional Waypoint Following mode. If at any point while in the Velocity Matching mode the controller detects a positive $\dot{p_z}$, the controller saves the system with a Recovery transition.

The Recovery transition is a simple routine to achieve straight and level flight with $\dot{\vec{p}} = \vec{0}$. This transition is triggered on the detection of a successful collision or unsafe approach.

The Envelope Violation transition can occur during any portion of the Conventional Waypoint Following, and is triggered when the UAS leaves a predefined safety barrier on p_x and p_y. When this transition is triggered, the UAS is commanded to safely hover in the center of the testing area.

The inner loop for the UAS in all flight modes is the same combination of a proportional controller for attitude and a proportional-integral-derivative controller for attitude rate, see Fig. 2.4. The gains for these two controllers were determined empirically through flights with the UAS and attached pogo stick.

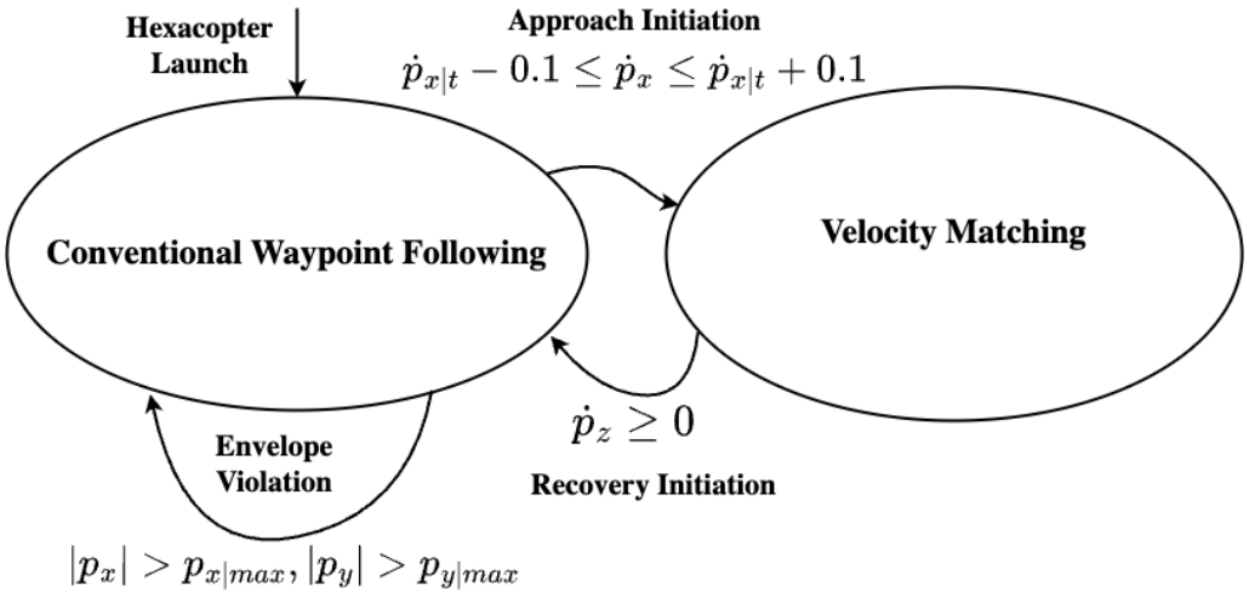

Figure 2.5: Hybrid control model for UAS using physical parameters as guard conditions

In the Conventional Waypoint Following mode, the additional use of the outer loop of the cascaded control presented in Fig. 2.3 maintains altitude (for use with the Approach Initiation transition), and 3-D position (for Recovery, Envelope Violations, and normal flight).

2.3.3 Thrust Limitation in Velocity Matching Mode

When the UAS is in Velocity Matching mode, the total thrust of the system is set to an experimentally determined total thrust of 9.4 N. This limit allows for minimal lift while maintaining a degree of authority for attitude control. Minimizing lift is important to having a forceful collision, so that the pogo stick is compressed and the UAS is launched upwards after contact. For this design, we assume that the system has much more influence over its pitch q_1 than its position p_x and p_z when thrust is limited.

By linearizing the 2-D flight dynamics outlined in Section 2.2.1, a relationship between T_{dif}, T_{tot}, q_1, $\ddot{q}_1$, $\ddot{p}_x$, and $\ddot{p}_z$ can be found. To linearize the equations governing UAS flight, we define state vector x and input vector u to express the state derivative $\dot{x}$ as a linear combination of each state in x and each input in u, as derived from the nonlinear system presented earlier in Eq. 2.1, with the disturbance force $F_d = 0$.

$$x = x_o + \delta x = \begin{bmatrix} q_1 \\ p_x \\ p_z \\ \dot{q}_1 \\ \dot{p}_x \\ \dot{p}_z \end{bmatrix}, \; u = u_o + \delta u = \begin{bmatrix} T_{dif} \\ T_{tot} \\ mg \end{bmatrix} \tag{2.15}$$

$$\dot{x} = f(x,u) = \begin{bmatrix} \dot{q_1} \\ \dot{p_x} \\ \dot{p_z} \\ \frac{rT_{dif}}{J} \\ \frac{T_{tot}\sin q_1}{m} \\ \frac{-T_{tot}\cos q_1 + mg}{m} \end{bmatrix} \tag{2.16}$$

We can relate local perturbation states x their derivatives $\dot{x}$ and inputs u as

$$\delta\dot{x} = A\delta x + B\delta u. \tag{2.17}$$

By linearizing the system with $T_{tot|0} = 9.4\,\text{N}$ and $q_{1|0}$ =45°, where $r = 0.275$ m, $m = 1.687$ kg, and $J = 0.00915$ kg m^2 [51], we can derive the following relations:

$$\delta\dot{q}_1 = \frac{r}{J}\delta T_{dif} = 30.055\, T_{dif} \tag{2.18}$$

$$\delta\dot{p}_x = \frac{T_{tot|0}\cos q_{1|0}}{m}\delta q_1 + \frac{\sin q_{1|0}}{m}\delta T_{tot} = 2.93\,\delta q_1 + 0.311\,\delta T_{tot} \tag{2.19}$$

$$\delta\dot{p}_z = \frac{T_{tot|0}\sin q_{1|0}}{m}\delta q_1 - \frac{\cos q_{1|0}}{m}\delta T_{tot} = 2.93\,\delta q_1 - 0.311\,\delta T_{tot} \tag{2.20}$$

The coefficients in Eqs. 2.18, 2.19, and 2.20 govern the relationships between $\delta\dot{q_1}$, δT_{dif}, δq_1, δt_{tot}, $\delta\dot{p_x}$, and $\delta\dot{p_z}$. We see that the coefficient on δT_{dif} in Eq. 2.18 is two orders of magnitude greater than the coefficients on δT_{tot} in Eqs. 2.19 and 2.20. This means that the UAS should have significant authority over pitch q_1 while having lower authority over the change of the velocity vector when pitching with lower limited T_{tot}.

This lower influence over velocity change additionally makes the velocity matching easier to track. If the influence of pitch q_1 over velocity was highly coupled, velocity matched pitches would cause drastic changes in the direction of velocity vector meaning that the already moving setpoint q_3 more difficult to track. The lesser coupling outlined (see Eqs. 2.18, 2.19, and 2.20), means that changes in velocity and thus q_3 will be slower.

2.4 Flight Testing Methods

We used a DJI F550 Hexacopter frame as the platform for testing collision-based aerial manipulation (see Fig. 2.6). The hexacopter used six 935kV brushless motors controlled at 48MHz each equipped with a 32-bit electronic speed controller and paired with a 1045 plastic propeller. We used a single 3000mAh 4 cell LiPo battery to power the UAS.

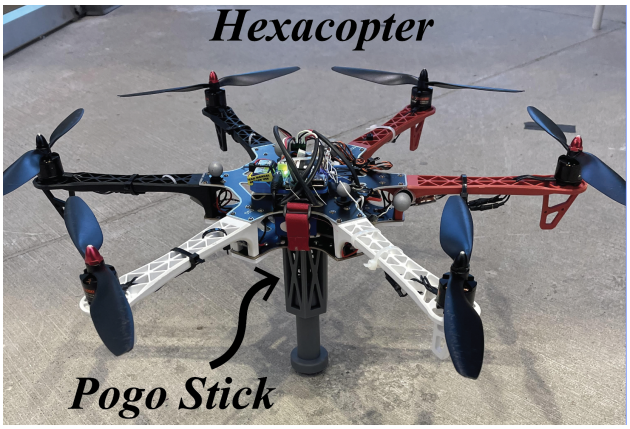

Figure 2.6: Modified DJI F550 Hexacopter (550 mm wheelbase) that is acting as the UAS for boucing interactions. The attached pogo stick was used to facilitate collision.

We fabricated the 22.86 cm long, two-part pogo stick (see Fig. 2.6) with SLA printed resin in the form of a cylinder frame with a rubber tip (for increased friction on contact). The springs in the pogo stick could be replaced to change the collision response of the UAS. We selected a 2.68 kgf/cm spring, stored in the upper housing of the pogo stick. The total system weight was 1.687 kg. The coefficient of static friction μ between the floor of the flight space and pogo stick was experimentally determined to be 0.42 using a specialized apparatus documented in [23]. The effective coefficient of restitution for the system with the 2.68 kgf/cm spring was experimentally determined to be 0.744 (used in collision response simulations).

We used an open source modified version of Ardupilot (`rc_pilot_a2sys`) to implement the flight control system described above. The flight control software ran on a BeagleBone Blue. Position and velocity information was streamed from an Optitrack system to a BeagleBone ground control station, while pitch, roll, and yaw states plus angular rates were collected from an onboard IMU.

We developed a 2-D simulation software in Python to compute rigid body collision response profiles between the UAS and a flat ground. This simulation models the complete system seen in Fig. 2.2. The time step for the simulation was 0.005s, the same as control loop time step on the flight controller. Simulation dynamics were computed via Euler integration.

2.5 Results

2.5.1 Velocity Matching Performance

Fig. 2.7A shows the overall trajectories of the UAS from approach through recovery. This also shows that collision is not instantaneous, as assumed in simulation. We have identified a region in time (contact) where the pogo stick spring is engaged and applying a restoring force. After the contact period, the Velocity Matching pitch setpoint, $q_{1|setpoint}$, is set to 0°, demarcated by the asterisk on Fig. 2.7A. Fig. 2.7B shows that the pitch tracking itself is performing poorly with a roughly 10° error at collision for this flight test ($p_{z|initial}$ = 1.5 m, $\dot{p}_{x|initial}$ = 1.8 m/s). This large error still allowed for a recoverable collision, and slower speeds at the same altitude show less error at collision.

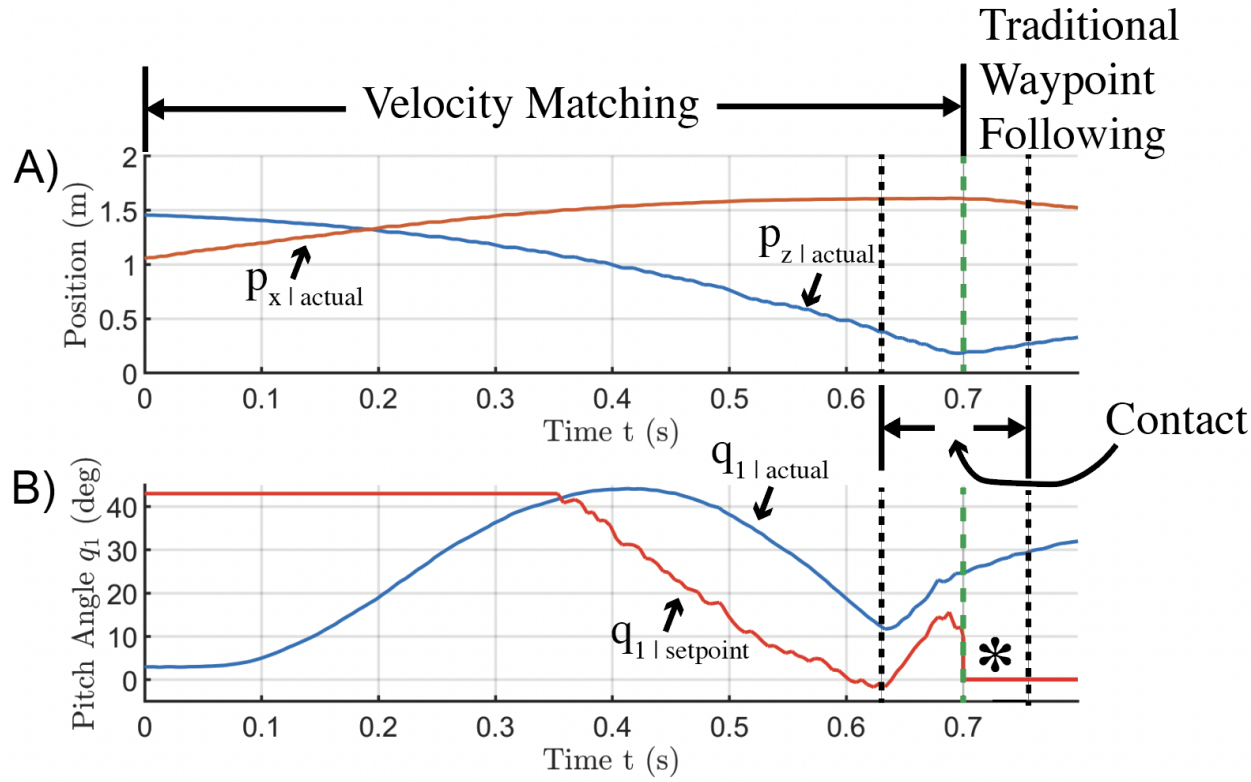

Figure 2.7: A) Visualization of UAS collision with $p_{z|initial}$ = 1.5m, $\dot{p}_{x|initial}$ = 1.8m/s. B) Performance visualization of pitch in Velocity Matching controller over the same time period. Asterisk demarcates where the Velocity Matching controller is toggled off and a zero pitch setpoint is being commanded after a bounce has occured.

2.5.2 Flight Envelopes for Collision Approach

A typical flight envelope is a constraint set based on aerial vehicle properties (ie. velocities, thrusts, control surfaces, and structures) that allows the vehicle to stay within its operational limits (ie. avoid structural failure or stalling) [41]. For aerial manipulation with collisions, we define our flight envelope as the initial altitudes and horizontal velocities that will allow for recovery from intentional collisions. A recoverable collision is defined as a collision in which a hover can be reached post-collision without propellers striking the ground.

In order to identify the simulation flight envelope 250,000 simulations were run at varying horizontal speeds and altitudes. The combined grey and yellow region in Fig. 2.8 indicates those initial altitudes $p_{z|initial}$ and horizontal speeds $\dot{p}_{x|initial}$ that are recoverable, describing the simulation flight envelope while the orange region is outside the simulation flight envelope.

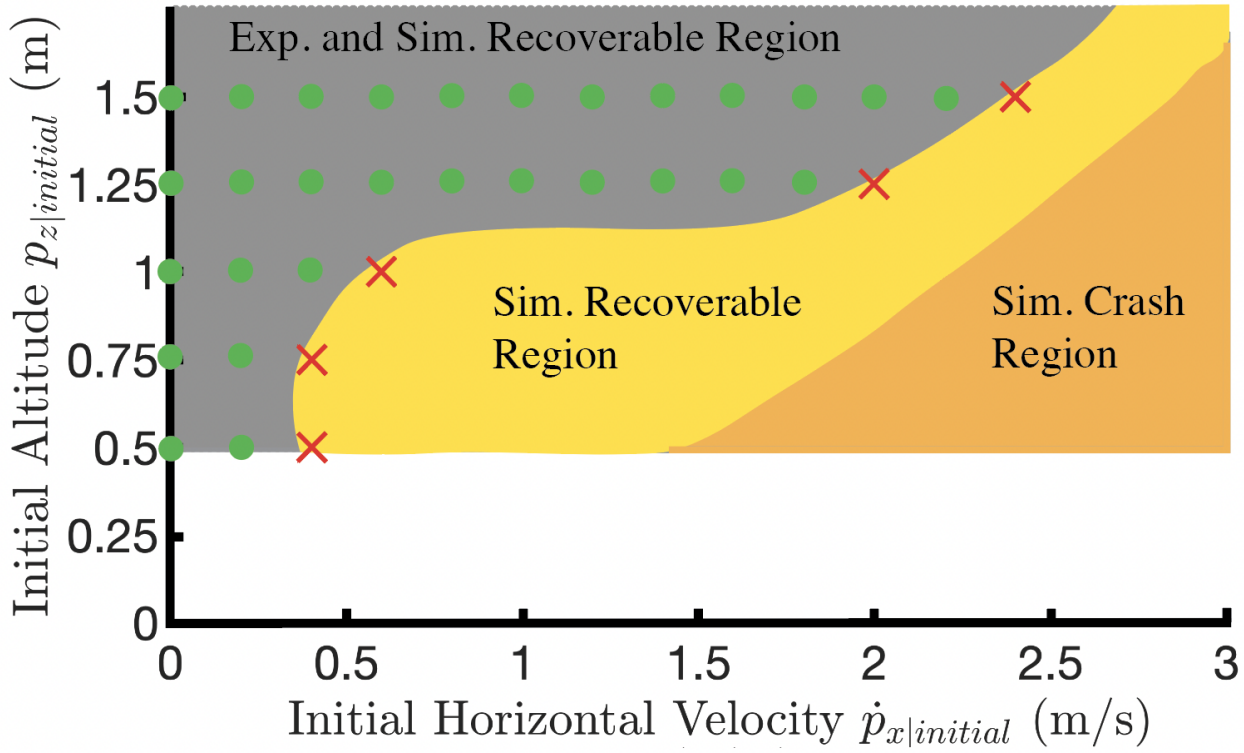

Figure 2.8: Two-dimensional flight envelope mapping initial altitudes $p_{z|initial}$ and horizontal velocities $\dot{p}_{x|initial}$ to successful collisions. The area shaded in yellow and grey represents successful collisions, while the area shaded in orange are failed collisions in simulation. The green dots and red x's represent successful and unsuccessful experimental flights that each included a collision, which form the grey shaded area. The unshaded region represents altitudes that were too low to obtain consistent collision data, due to the length of the pogo stick.

Additionally, 35 experimental collision flights were conducted (in increments of 0.25 m

$p_{z|initial}$ and 0.2 m/s $\dot{p}_{x|initial}$) to identify the experimental flight envelope. These test results can be seen in Fig. 2.8, with the green dots representing successful collisions and the red x's representing crashes. The experimental flight data was used to create the grey shaded experimental recoverable region that separates successful collisions and crashes. No tests were conducted past the first horizontal speeds that resulted in a crash at a given altitude.

2.6 Discussion and Conclusions

We have presented and compared 2-D models for a UAS during flight and during perch to assess the impact of the constraints imposed by perch on disturbance rejection control properties. We argue that a UAS can interact with its environment without sacrificing disturbance rejection when limiting overall interaction time. By using collisions as our method of choice for environmental interaction, we limit the total interaction time to be near instantaneous and turn collision impulses into forces that move the UAS away from the ground after collision is completed. Aerial manipulation with collisions was made possible through the development of a Velocity Matching controller, which simplifies approach control to an attitude and velocity matching controller.

A 2-D, rigid body simulator was developed both for control design and validation but also for uncovering a flight envelope for the Velocity Matching flight mode. This simulation flight envelope covers the experimental flight envelope (see Fig. 2.8) showing that it is a predictor of safe Velocity Matching conditions for the UAS. Experiments were run in the longitudinal plane, but the Velocity Matching controller had to be 3-D to reject lateral disturbances. Discrepancies in the flight envelopes can be explained partially by the difference between rigid body and pogo stick contact with the ground (see Fig. 2.7A) as well as other unknowns in the physical system that were not modelled. The pogo stick in simulation has no compression and the propellers provided instant torque while this is untrue in the physical experiment. The poor performance of the velocity matching controllers at low $p_{z|initial}$ with higher horizontal speeds may also be influenced by latency in the experimental setup, as motion capture data must be initially recorded, then streamed via USB to the ground station, and finally streamed via XBee radio to the UAS during free-fall before velocity matching can run and motors can begin being commanded. This latency was not modelled in simulation, but was found to have a mean of 0.0084 s with a standard deviation of 0.0011 s during experiments. Even for low latency communications, the shortest falling time during Velocity Matching (average of 0.32 s for altitude of 0.5 m) exacerbates the problem. Nonetheless, the existence of these flight envelopes is essential for extending this work further into the domain of aerial manipulation, where fully understanding when and where the UAS can safely collide is paramount (i.e.

during tree leaf sampling).

The goal of this work was to lay the foundation for solving autonomous rainforest tree sampling. We believe that aerial manipulation with collisions can be extended to any manipulation task that can completed in a single pass with a short interaction window. Follow-on work for this project includes extending the current foot placement strategy to allow for bouncing on a specified target in 3-D and developing a pogo stick manipulator that can cut and grasp during collision to further pursue the purpose of rainforest tree sampling.

CHAPTER 3

Overcoming Underactuation in UAS by Switching Cascaded Controllers

3.1 Introduction

Uncrewed aircraft systems (UAS) are, by design, underactuated, which limits their overall control authority [22]. Given that stereotypical UAS have between four and eight propellors facing parallel, these actuators combined to one thrust and three body torques. Because the UAS have effectively four actuators and six degrees of freedom, it is not possible to control all degrees of freedom at once. Angular accelerations and linear accelerations are coupled based on the thrust and attitude of the UAS. This coupling between attitude and position does not preclude the ability to control position through simple cascaded PID control [14], but it also makes tasks that demand precise control of both position and attitude infeasible without a change in approach.

This work aims to overcome the underactuation of UAS to enable aerial manipulation via preplanned collisions. Building on our prior work [48], we leverage intentional impacts to manipulate objects of unknown dynamics, with a particular focus on autonomous tree sampling. Collisions constrain the interaction time and thereby reduce the risk associated with prolonged contact with objects of unknown dynamics. While pogo-stick–enabled or collision-based payloads on UAS have been explored, most applications target energy savings during hover or terrestrial locomotion rather than manipulation [79, 76, 7].

As shown in [47], the rigid-body collision response is largely governed by the center-of-mass linear and angular velocity at strike, and the angle between the center of mass of the UAS and the environment at strike (the contact angle). Even small attitude deviations can induce dramatic post-impact responses due to misalignment of the contact impulse and the CoM of the UAS relative to the point of contact. These dramatic post-impact responses impart disturbance moments on the UAS that are difficult to reject, often leading to a crash.

A variety of nonlinear controllers, such as MPC [31] and optimal trajectory replanning [17], could mitigate these issues, but they are often computationally intensive and difficult to deploy online. Given that collisions are nearly instantaneous and the setup trajectories typically last under two seconds [48, 47], such approaches are impractical for our goal of a simple, low-cost rainforest tree-sampling UAS.

Instead, we draw on switching control [28, 19, 80], wherein a system with multiple actuators, subsystems, or tasks switches between control strategies. We propose that intentional collisions fit naturally in this framework: disabling position control near the end of flight may allow the attitude controller to make last-minute adjustments with minimal impact on the strike location.

In this work we present a switching scheme between two cascaded PID modes: (i) position cascaded and (ii) attitude cascaded control. Using an offline trajectory generator, we identify an optimal switch time for a given collision trajectory in simulation, targeting minimal position and attitude errors at impact. We then validate these predictions experimentally, showing that the simulation can predict the appropriate switching time that minimizes angular and positional error for a given initial state (displacement from strike location) before strike. This approach enables reliable execution of preplanned collisions without online optimization, offering a simple tool to work around UAS underactuation for aerial manipulation.

3.2 Model Development and System Architecture

3.2.1 Flight Dynamics in 2-D

To simplify the dynamics of the pogo-stick–enabled UAS (see Fig. 2.6), we have provided Fig. **??**, where horizontal displacement is q_1, vertical displacement (altitude) is q_2, and pitch angle is q_3. Let $\mathbf{Mg}$ denote the force vector due to gravity, $\mathbf{f}$ denote the total thrust of the system, and $\boldsymbol{\tau}$ denote the body torque acting on the UAS. From the figure, the following equations of motion are pulled with thrust $\mathbf{f}$ and torque $\boldsymbol{\tau}$ acting as the two actuators:

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} = \begin{bmatrix} -\dfrac{\mathbf{f}\sin q_3}{M} \\ \dfrac{\mathbf{f}\cos q_3}{M} - g \\ \dfrac{\boldsymbol{\tau}}{I_z} \end{bmatrix}, \tag{3.1}$$

where M and I_z are the mass and moment of inertia of the system, respectively.

As discussed previously, the UAS is underactuated: attitude q_3 directly affects the linear

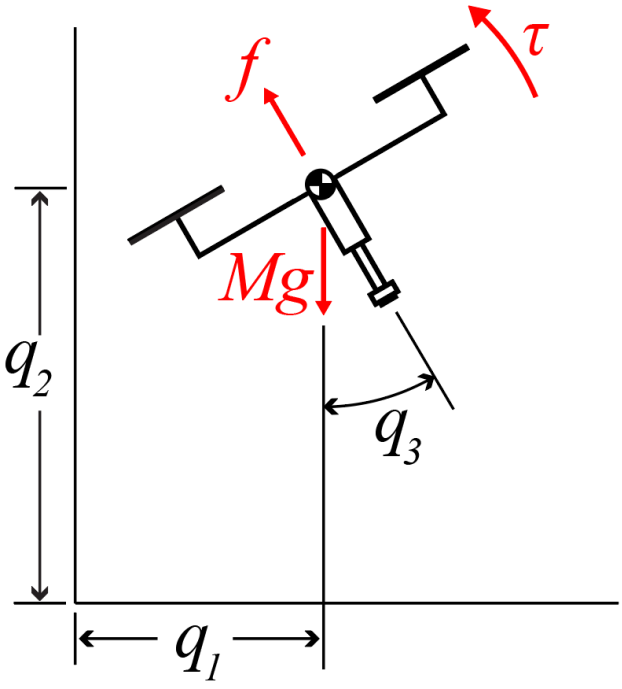

Figure 3.1: 2-D UAS Flight Dynamics

accelerations, but there are only two inputs for three states. In practice, that means we can tightly control either A) horizontal position and altitude or B) attitude and altitude, assuming altitude must be regulated to stay airborne. Leaving position uncontrolled for most of a flight is risky (the vehicle can drift anywhere) so position control remains necessary for safe trajectory generation.

3.2.2 Cascaded Switching Controller

We adopt the cascaded control structure from [48], augmented with a toggleable switch that can disable the position cascaded controller and swap into attitude cascaded control. This is shown in Fig. 3.2, where red traces denote the switch that can be turned on and off as needed.

By allowing a switch between position and attitude control, we can shape the final strike position and orientation. If position is paramount, the switch stays with the position PID for most of the flight; if attitude is paramount, we bias toward the attitude loop. However, intentional collisions for aerial manipulation highlight a core conflict: we cannot drop position control (or we may miss the target), and we cannot drop attitude control (or the collision response will be wrong). As noted in [47] and previously stated, collision responses depend on velocities, position, and attitude.

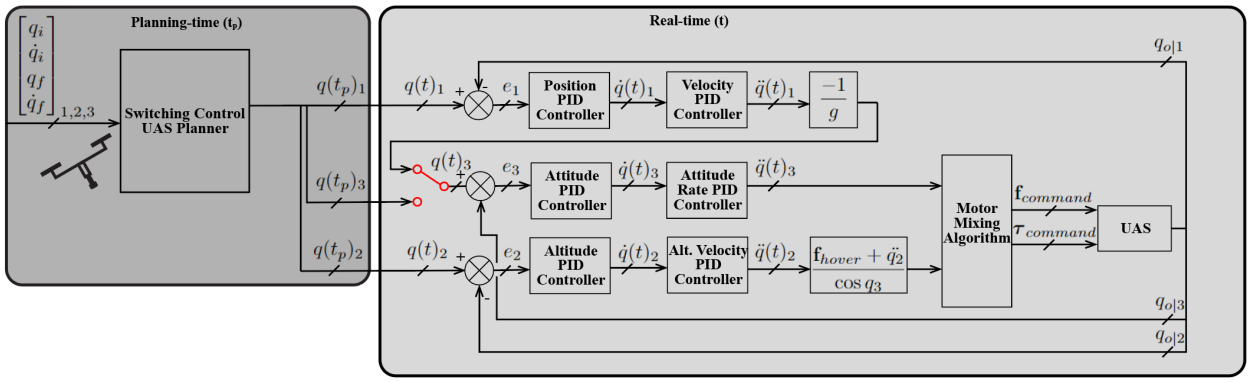

Figure 3.2: Cascaded switching control. The offline trajectory planner feeds waypoints into either the position+altitude controller pair or the attitude+altitude controller pair. The red toggle shows where the switching controller can bypass position control, sending the planner's commands directly to the attitude loop. Let $q(t_p)_k$ denote the generated trajectory and $q(t)_k$ a single waypoint provided to the controller. The notation $q()_k$ denotes which configuration variable is being traced, horizontal position q_1, altitude q_2, and pitch q_3.

Because neither position nor attitude can be ignored, we propose using simulation to compute the optimal switch time (expressed as a percentage through the flight) for different initial strike configurations (horizontal and vertical displacement from the target, desired strike velocity, and desired strike angle). Trajectories are therefore expressed as waypoints that combine position and attitude commands:

$$w_{\text{switch}} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ t \end{bmatrix}. \tag{3.2}$$

The interpretation of each waypoint depends on the active controller: position control ignores q_3, while attitude control ignores q_1. With no disturbances, following either set perfectly will place the vehicle at the desired position; attitude at strike is not guaranteed under position control, though. This is because the position cascade feeds the velocity PID's output into the attitude loop, so attitude may track what the position controller demands rather than the trajectory's explicit attitude commands.

This is why switching is needed. Let position control be toggled on for most of the flight, then toggle to attitude control to track the tail end of the trajectory. This enables last-moment attitude corrections with minimal deviation from the intended strike location.

Table 3.1: Switching Controller: Trajectory Generation Costs

Penalizing	Cost Function	Cost Weight
Flight Time T (s)	$w_T\, T$	1.0
Positive $\dot{q}_2$	$w_{\dot{q}_2} \sum_{k=1}^{N} \left(\max(0, \dot{q}_2)\right)^2$	0.1
Control Effort U	$w_U \sum_{k=1}^{N} U^2$	0.0001

Table 3.2: Switching Controller: Trajectory Generation Constraints

Constrained Variable	Constraint Equation
$q_{f\|1}$, $q_{f\|2}$, $q_{f\|3}$	$q_{uf\|1}$, $q_{uf\|2}$, $q_{uf\|3}$
$\dot{q}_{f\|1}$, $\dot{q}_{f\|2}$	$\dot{q}_{uf\|1}$, $\dot{q}_{uf\|2}$
q_3	$\|q_3\| \leq 0.75$ rad
$\dot{q}_1$, $\dot{q}_2$	$\|\dot{q}_1\|$, $\|\dot{q}_2\| \leq 2.0$ m/s
$\dot{q}_3$	$\|\dot{q}_3\| \leq 3.0$ rad/s

3.2.3 Optimal Trajectory Solver

To generate trajectories feasible for both position and attitude tracking, we built an optimal trajectory solver using the Interior Point OPTimizer (IPOPT) via the CasADi Python library. This lets us plan directly on the system model to produce dynamically feasible trajectories. Given starting states $(q_i, \dot{q}_i)$ and desired final states $(q_f, \dot{q}_f)$, the solver finds the fastest flight path that satisfies strike conditions. Additional costs penalize control effort U and upward motion, biasing the vehicle to fly toward the target with minimal effort. The costs are summarized in Table 3.1, and constraints in Table 3.2. Trajectories are generated offline so that no optimization runs on the computationally limited BeagleBone Blue flight controller.

3.3 Hexacopter Simulation Environment

3.3.1 Simulation Architecture

We built a 3D Python simulator that mirrors the open-source flight controller (`rc_pilot_a2sys`) used on the actual UAS (Fig. 2.6). The simulator includes the system dynamics (mass properties, motor delays, and communication delays) that were experimentally validated prior to use. The simulator runs the control loop at 400 Hz and receives motion-capture data at 120 Hz, matching the real system. The simulator runs alongside the trajectory generator on a Windows desktop (32 GB RAM, 12-core AMD Ryzen 9 3900XT), enabling rapid batch trajectory generation and simulation.

This simulator is the backbone of the switching architecture. By leveraging offline compute, it can batch-generate trajectories and then simulate the switching controller to find the optimal switch time for a given initial configuration.

3.3.2 Flight Testing Methods

For experimental flights we used the same UAS shown in Fig. 2.6, built on a DJI F550 hexacopter with the added pogostick payload. The UAS was modified with 30 A ESCs, 1000 KV 2826 brushless motors, and 10×3.8 APC Slow Flyer propellers, chosen for higher thrust and faster response than the actuators used in Chapter 2. The rigid pogostick payload was unchanged.

The UAS ran a modified version of the open-source flight controller (`rc_pilot_a2sys`) at 400 Hz on a BeagleBone Blue. A Vicon motion-capture system streamed pose over XBee radios to minimize latency and provide ground-truth poses to the control loops. As in simulation, motion capture ran at 120 Hz.

3.4 Results

3.4.1 Flight Sample Test Results

Flight tests were conducted to validate the simulator's predicted best switching times, with initial q_2 displacements of 1 m above the strike location and varying q_1 displacements (-0.1 m, -0.3 m, -0.5 m, -0.8 m, and -1.0 m). The strike position was set 1 m above the ground as a virtual strike point to reduce crash risk from unfavorable collision responses during real strikes (when strike position/angle errors are high). Switching times were tested and recorded as percentages of the trajectory duration to standardize across trajectories of different lengths. The "switching time" is when the horizontal position loop is disabled and attitude–altitude control is used. We tested switch times of $50\%, 60\%, 70\%, 80\%, 90\%$, and 100%. For safety, we did not test switch times earlier than 50% since most of the flight would be open-loop in position. In total, we flew 150 trials: 5 flights for each (q_1, switch) pair.

Figure 3.3 shows a sample flight's q_1 and q_3 performance with the switching time marked by a dashed red line. After the switch, position setpoints are toggled off; prior to switching, the attitude setpoints come from the position-controller cascade (Fig. 3.2). This example shows the cost of switching too early: even with $< 1^\circ$ attitude error at strike, position is uncontrolled for 40% of the flight and overshoots by 0.2 m.

Figure 3.4 shows the same initial conditions but with a later switching time (90% through the flight). Here, switching improves performance by allowing last-minute attitude adjustments once the vehicle is near the desired strike location. The switch is late enough that the uncontrolled position stays close to the goal (< 0.03 m), and the strike angle is within a degree of the target.

3.4.2 Summary Results: Simulation and Experiments

As noted above, we built a 3D Python simulation that mirrors the experimental UAS flight controller. We used it to estimate the best switching time for a given displacement from the target strike location. Summary results were generated by sweeping q_1 (x) displacements at a fixed q_2 (z) of -1.0 m and scanning switching times from 0% to 100% in 5% increments. Each condition was simulated for 50 flights. The subsequent flight experiments tested along this slice with switching times greater than 50%. Figure 3.5 shows a slice in the x–z (q_1–q_2) plane comparing predicted and experimentally determined best switching times for each q_1. We defined the "best" switching time as the one that minimizes the average cumulative error across trials:

$$e_{\text{tot}} = |e_x| + |e_y| + |e_{\text{angular}}|. \tag{3.3}$$

Predicted best switching times closely match the experimentally determined values (Fig. 3.5) when x displacements are large, indicating the simulator is effective for choosing switching times across varying initial conditions. The largest discrepancies appear for q_1 displacements under 0.5 m.

Figure 3.6 shows that while the best experimental switching times are very close to the predicted switching times, the associated errors are substantially higher in the experiments and there a large standard deviation of error between trials.

3.5 Discussions and Conclusions

We present a method for working around underactuation in rotary UAS for intentional collision based aerial manipulation, where both position and attitude tracking matter. The method, Switching Control, swaps from a conventional position–altitude cascaded PID to an attitude–altitude cascaded PID during flight. With an appropriate switch time, this enables last-second attitude corrections while minimally perturbing strike location, as validated in both experiments and the 3D simulator (which we also used to aid in choosing the switch time).

Results show that switching earlier helps more when the UAS starts farther from the strike

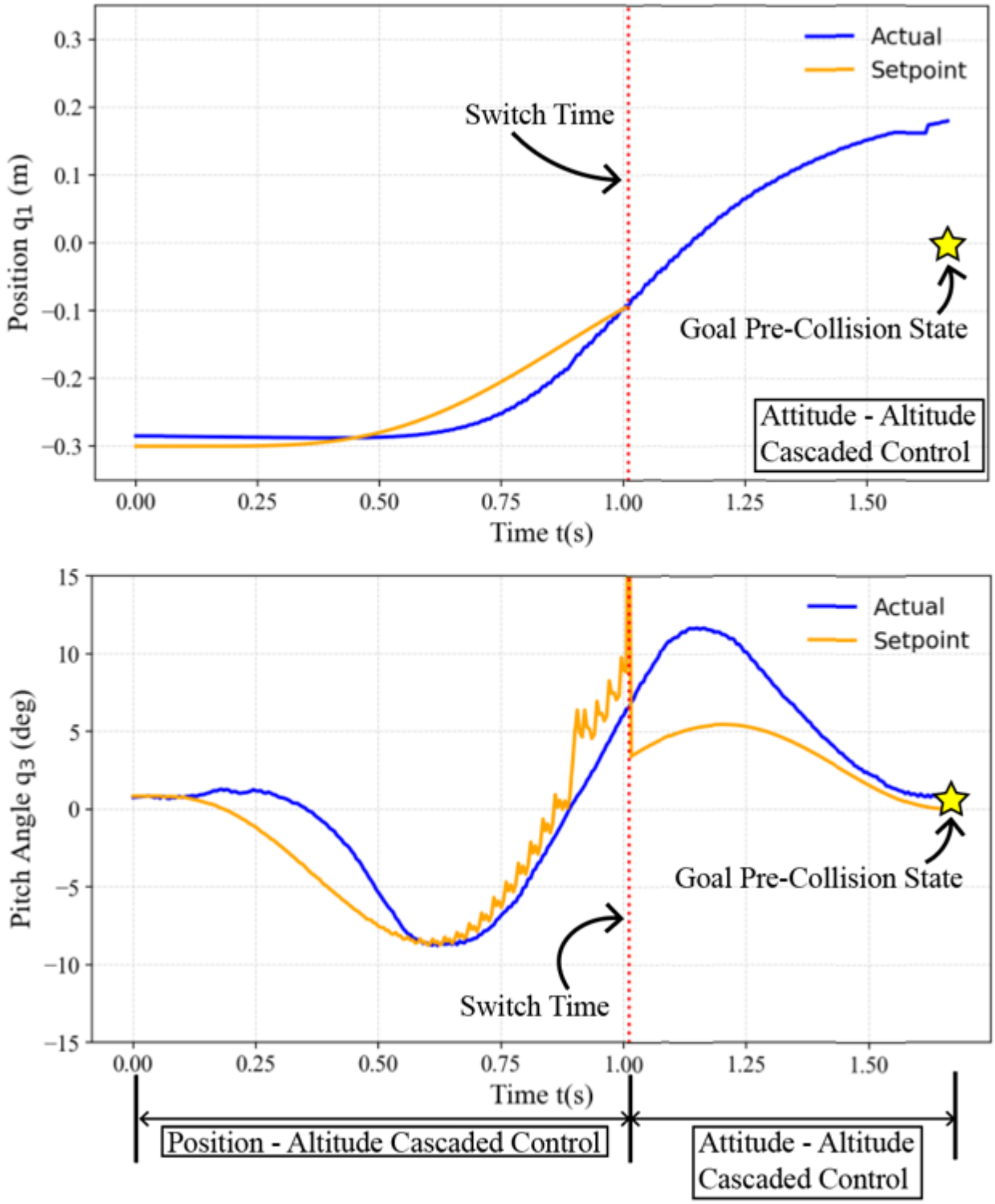

Figure 3.3: Example of a poor switching time (60%) for initial $q_1 = -0.3$ m, from experimental data. The dotted red line denotes the switch from position–altitude to attitude–altitude cascaded control. Goal states are denoted by a yellow star.

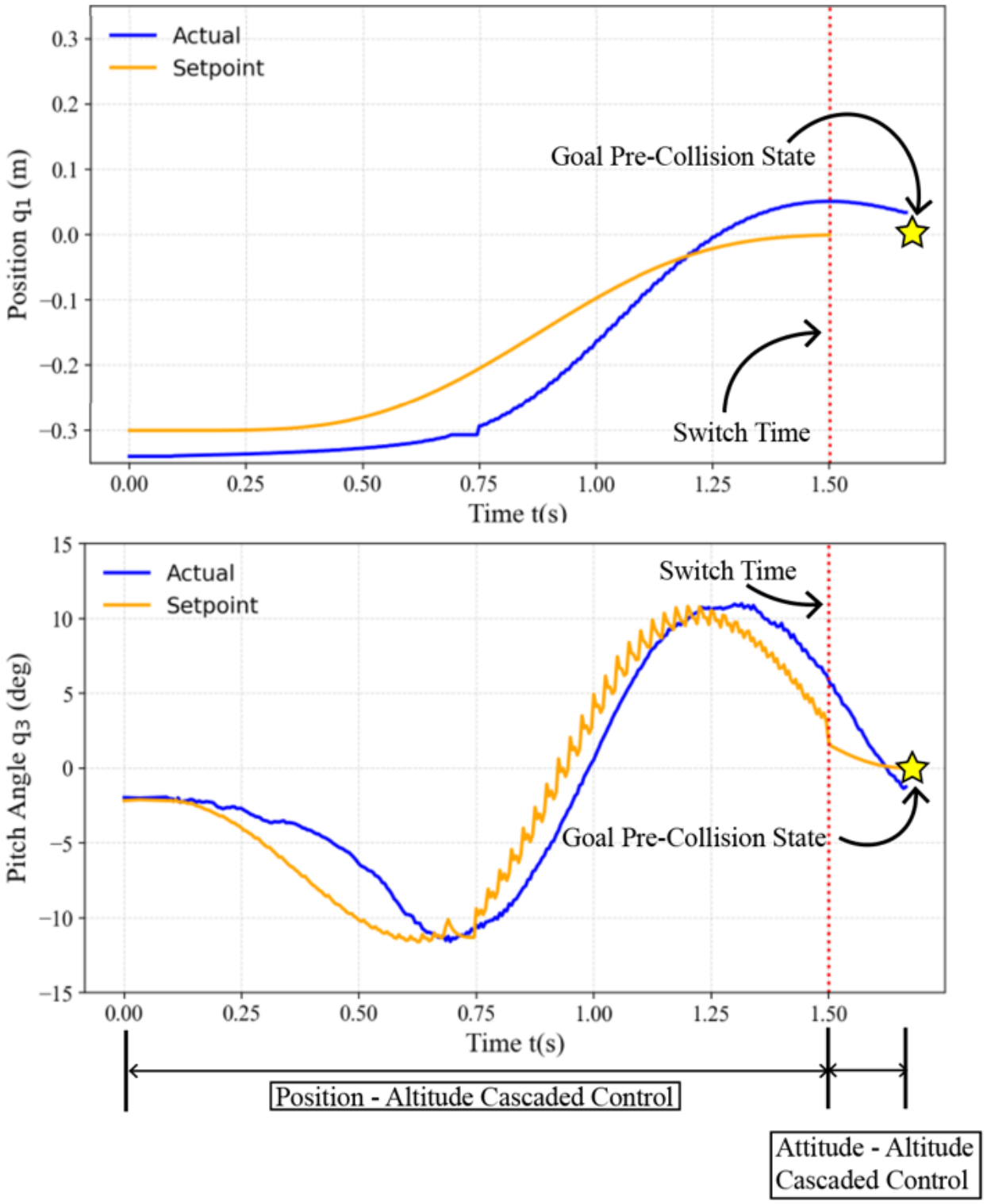

Figure 3.4: Example of a good switching time (90%) for initial $q_1 = -0.3$ m, from experimental data. The dotted red line denotes the switch from position–altitude to attitude–altitude cascaded control. Goal states are denoted by a yellow star.

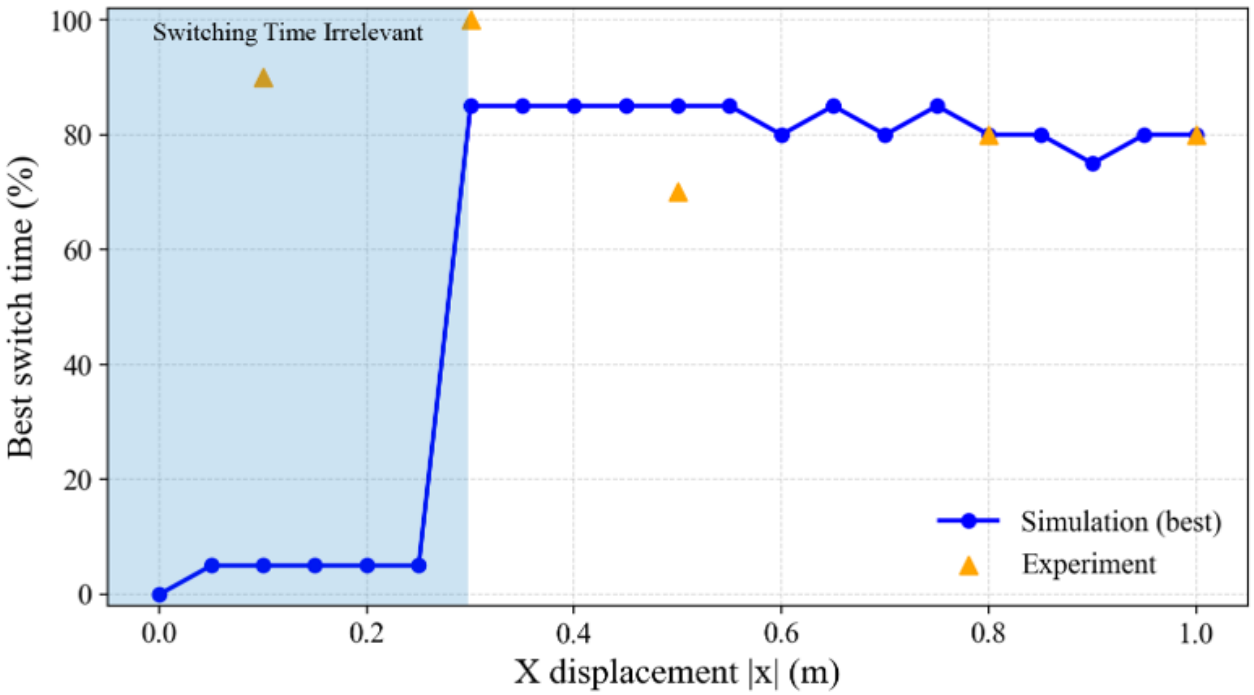

Figure 3.5: Experimentally determined best switching times overlaid on simulation-predicted best switching times. Switching time is the percentage of the flight at which position control is toggled off. Light shaded blue region shows were switching time was irrelevant and errors for all switch times were low in simulation and experiments.

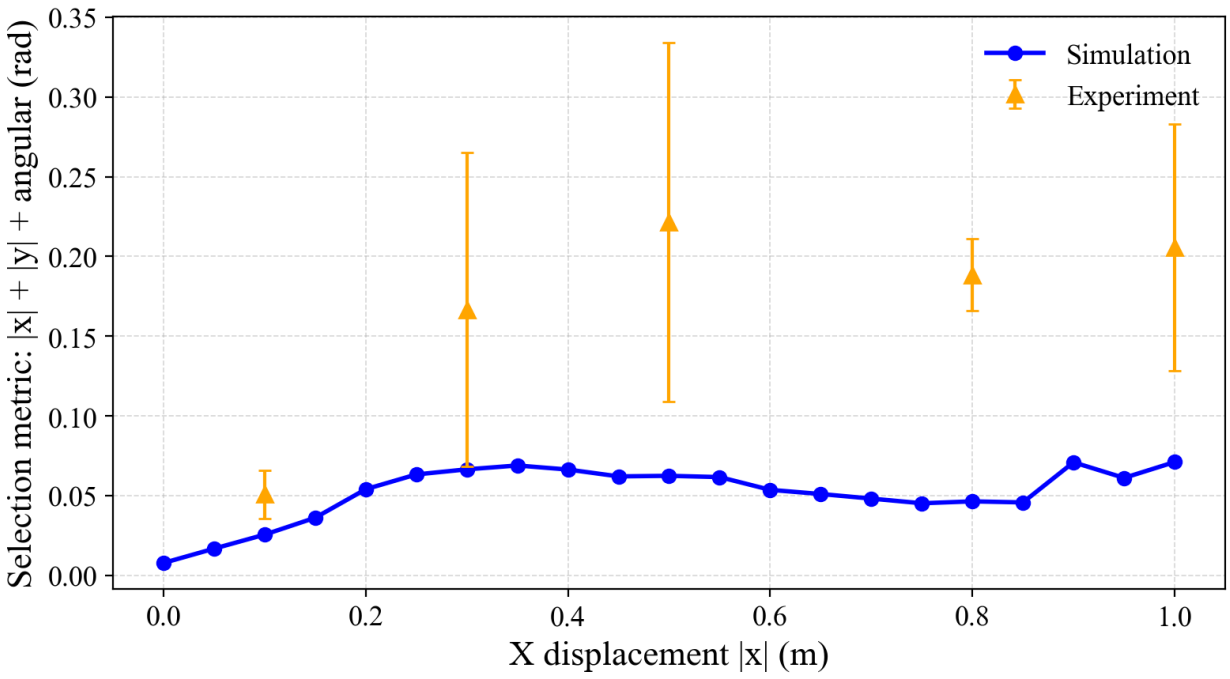

Figure 3.6: Experimental errors at the best switching times overlaid on simulated errors at the predicted best switching times. Errors are the sum of lateral errors in the x–y plane and angular error from vertical.

location. In those cases, the position controller commands large attitude changes to reach the target quickly, so attitude deviates more from the optimal trajectory and needs extra time to recover. When the UAS starts closer, the position corrections (and thus attitude deviations) are smaller, so switching is less critical. Once Switching Control becomes effective, there is a clear cliff around ∼80% switch time that remains near-optimal for all x displacements in both simulation and experiments. Given that errors for small x displacements are minuscule, a switch time of roughly 80% may be effective for most pre-collision state configurations even when x displacement is small.

The summary results also show that experimental errors were much higher than simulation errors, although the overall shape is similar. This suggests unmodeled dynamics in the UAS plant or in the communication stack used during experiments. Even so, the simulation's predicted switching times were accurate when validated experimentally.

Our goal was to improve on the controller used in [48] for aerial manipulation with intentional collisions. The prior approach used a launching controller and a velocity-matching controller to align the pogostick with the UAS velocity vector at strike, assuming a ballistic fall. This assumption breaks when motors generate thrust for attitude changes. Switching Control makes no such assumption and supports arbitrary combinations of strike location,

velocity, and angle, as long as a dynamically feasible trajectory exists. Future work includes more flight tests to build a full surface of optimal switching times at a higher resolution (not just a single slice) and bounce testing with real collisions to map a flight envelope that separates recoverable from unrecoverable pre-collision conditions, paralleling the envelope in [48]. It will also be useful to further refine the simulator so it predicts errors (not just switch times) more reliably, and to spend more time experimentally validating additional initial-condition configurations with finer switching-time resolution. Further enhancement of the switching controller could be made and tested using a smooth transition similar to [72] from cascaded position-altitude control into attitude-altitude control. This smooth transition could take the outputs of the position-velocity PID cascaded and send them as a weighted feedforward term, while percentage through the switching of controllers adjusts the weight for the feedforward term. This could provide less abrupt switching and lessen aggressive flight performance surrounding the switch time.

CHAPTER 4

Shifting Underactuated Configuration Variables In Aerial Manipulation By Adding An Actuated Arm

4.1 Introduction

Uncrewed aircraft systems (UAS) promise to change the way humans interact with the world around them. While it has been typical to use UAS for tasks like inspection [59][24] and surveillance [45] that do not involve direct interaction with the environment, aerial manipulation produces contact and forceful interactions while in flight.

Broadly, aerial manipulation systems either involve perching [35], grasping [27], or pushing [50]. Of these three types of interaction, perching and pushing require extended contact between the UAS and its environment. This means that the UAS either needs to rely on the interaction surface for a base of support or to support itself during contact. Grasping tasks, on the other hand, are primarily undertaken with objects that are not rigidly attached to the environment, so quick interactions like swooping and grabbing can be completed to minimize external forces acting on the UAS.

The present work is focused on extending a system designed for sampling leaves and small branches from rainforest trees [48], in which the sample must be cut and then grasped. Tree branch mechanics vary from tree to tree presenting stability concerns over long-term contact, so our previous work relied on a pogostick attached to the UAS for short-term aerial manipulation via intentional collisions. Through collisions, we hope to lower the sensitivity concerns around control and stability during prolonged contact with tree branches or any item being manipulated, while still offering an effective avenue for aerial manipulation.

Pogosticks on UAS have been proposed for energy savings to extend times [79] [9] [15] and for locomotion [76][7]. All of these systems control the UAS's altitude and attitude, compromising position control during the approach to the collision, as did our previous work

[48].

Because only the magnitude and not the direction of thrust is modulated through parallel-mounted propellers, it is not possible to simultaneously track all position and attitude variables. The lack of simultaneous attitude and position tracking causes problems when wanting to make contact with a surface at a specific location. Collision response is a function of pre-contact velocities and contact configuration; as such, collisions that do not impart angular momentum on the UAS require full attitude control. This means the previously mentioned systems with rigid pogosticks cannot choose where to make contact while planning their post-contact velocities.

The problem of underactuation in UAS is not new. Several methods have been proposed to work around the limitations of underactuation. The two most common methods are optimization-based trajectory planning and nonlinear model predictive control (NMPC). Looking at the former, systems utilizing optimization-based trajectory planners incorporate the complete dynamics of the UAS and trajectory constraints to create dynamically feasible trajectories that meet specific goals. Example trajectory goals vary from aggressive flying around and through obstacles [43] to perching on walls [73]. These systems typically leverage the differential flatness [71] or other differential geometric properties [39] of the UAS flight dynamics to generate trajectories that ensure, when tracked, that a final state is reached. While tasks that require certain attitudes and positions at given times can be completed through these means, the tracking controllers themselves are still only controlling positions and the yaw angle of the vehicle. This means that any deviations or disturbances from the trajectory require re-planning of the trajectory because attitude is not directly controlled.

In contrast, NMPC does planning during flight through a “look ahead" window that continually produces new trajectories and thereby accommodates deviations and disturbances. The main challenge with NMPC is its computational overhead. NMPC requires optimization at regular intervals, requiring systems to have powerful onboard computers [55] [31] or for control calculation to be done on a separate machine [4]. Neither of these options is feasible for our candidate system which would ideally be inexpensive and encapsulated in one package for large-scale adoption in the tree sampling/botany community.

In the following, we propose the addition of an actuated pogostick to the UAS that contributes a new state variable but also a new control input. With the use of a partial feedback linearizing controller, the particular state that remains unactuated is chosen but is shifted such that when the arm collides with the environment, the resulting impulse has no moment about the UAS center of mass. The UAS without an actuated arm is presented first, called the Rigid Pogostick UAS (RP-UAS), which is the foundational system from Figure 2.6 that this paper is focused on improving [48]. Due to the underactuation of the RP-UAS,

closing the loop on contact position and contact angle for the simplified system is not feasible without NMPC or in-flight trajectory replanning. More control authority is required, so an Actuated Pogostick UAS (AP-UAS) is proposed that allows for control authority over both contact position and contact angle. Both of these systems are then tested using the same architecture: an optimal trajectory planner creating dynamically feasible trajectories for each system and a partial feedback linearizing controller tracking those trajectories.

4.2 Model Development

4.2.1 2-D Flight Dynamics for a Rigid Pogostick UAS

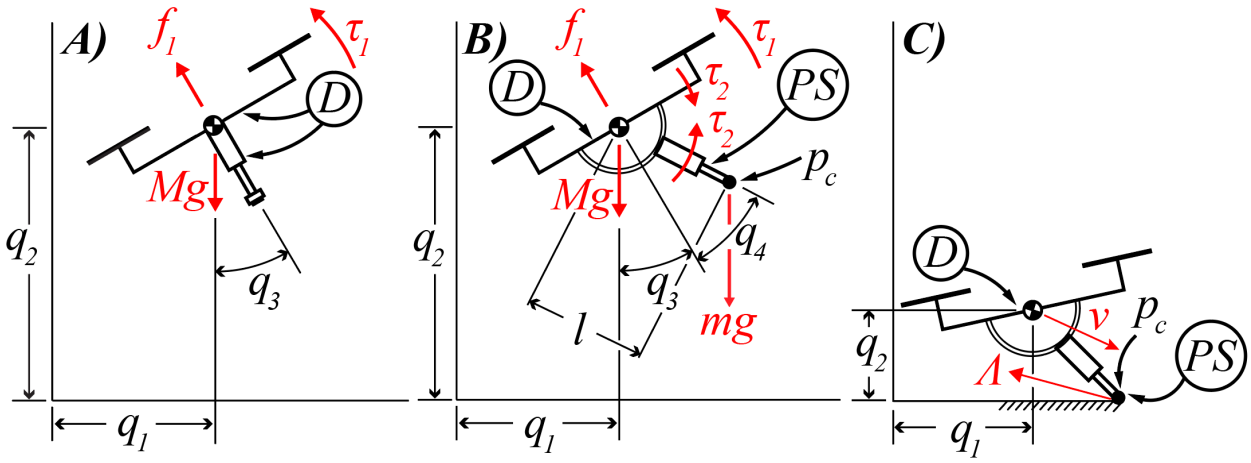

Figure 4.1: A) 2-D model of a Rigid Pogostick UAS in flight with body force $\mathbf{f_1}$ and torque $\boldsymbol{\tau_1}$ from the propellers. B) 2-D model of an Actuated Pogostick UAS in flight with body force $\mathbf{f_1}$ and torque $\boldsymbol{\tau_1}$ from the propellers. Torque $\boldsymbol{\tau_2}$ comes from a motor between UAS Body D and pogostick PS. C) Collision Response Model of an Actuated Pogostick UAS with rigid body D, point-mass p_c, contact angle $q_3 + q_4$, velocity for the rigid body $\mathbf{v}$, pogostick length l, and contact impulse $\boldsymbol{\Lambda}$.

Figure 4.1A presents a diagram of the Rigid Pogostick UAS (RP-UAS) which operates with only two actuators, thrust $\mathbf{f_1}$ (fixed in the UAS frame) and body torque $\boldsymbol{\tau_1}$, but has three states q_1, q_2, and q_3 making the system underactuated. The flight dynamics (see [48]) extracted from Figure 4.1A show that world-frame translational accelerations $\ddot{q}_1$ and $\ddot{q}_2$ are coupled with UAS pitch q_3.

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} = \begin{bmatrix} \frac{-\mathbf{f_1}\sin q_3}{M} \\ \frac{\mathbf{f_1}\cos q_3}{M} - g \\ \frac{\boldsymbol{\tau_1}}{I_z} \end{bmatrix}, \tag{4.1}$$

where M and I_z are the mass and moment of inertia of the system, respectively.

Because the system is underactuated, either position q_1, q_2 can be tracked or a combination of one or the other position variable and pitch q_3 can be tracked. Because tracking q_2 is necessary to stay in flight, commanded waypoints (pre-planned trajectories) are limited to the two possibilities:

$$q(t) = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \text{ or } \begin{bmatrix} q_2 \\ q_3 \end{bmatrix}. \tag{4.2}$$

If position q_1 is to be tracked, then authority over pitch q_3 is lost. Underactuated control problems also extend to 3-D flight dynamics where a typical UAS has four actuators (thrust and body torques) with six states to control. In most UAS applications, position control is more important than direct attitude control so a cascaded controller is typically applied to regulate positions x, y, and z and yaw ψ [57].

4.2.2 2-D Flight Dynamics for an Actuated Pogostick UAS

Figure 4.1B presents a diagram of the Actuated Pogostick UAS (AP-UAS) that features an added actuator that acts on the pogostick angle q_4 relative to the UAS body. Equal and opposite torques $\boldsymbol{\tau_2}$ applied to the UAS body D and pogostick account for the action of an onboard motor. The pogostick pivot is aligned with the center of mass, such that any forces along the pogostick axis do not impart moments about the UAS center of mass.

The flight dynamics of the AP-UAS have been derived with generalized coordinates q_1, q_2, q_3, and q_4 and are presented in Eq. 4.4. The symbols M and m represent the mass of the UAS body D and the mass of particle p_c, l represents the length of the pogostick, and s_i, c_i, and s_{ij} are shorthand for $\sin q_i$, $\cos q_i$, and $\sin(q_i + q_j)$ etc. The pogostick was simplified to a point mass to aid understanding, and this approach is common in the dynamic walking robots community [61] [2].

This system is still underactuated, but the additional actuator allows for complete control of two position variables and one angle, making for three possible waypoints formulations:

$$q(t) = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}, \begin{bmatrix} q_1 \\ q_2 \\ q_4 \end{bmatrix}, \text{or } \begin{bmatrix} q_1 \\ q_2 \\ q_3 + q_4 \end{bmatrix}. \tag{4.3}$$

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \\ \ddot{q}_4 \end{bmatrix} = \begin{bmatrix} \frac{Mml^2\dot{q}_3^2 s_{34} + 2L^2 Mm\dot{q}_3\dot{q}_4 s_{34} + Mml^2\dot{q}_4^2 s_{34} - Ml\mathbf{f_1}s_3 + ml\mathbf{f_1}s_4 c_{34} - M\boldsymbol{\tau_2}c_{34} - m\boldsymbol{\tau_2}c_{34}}{Ml(M+m)} \\ \frac{Mml^2\dot{q}_3^2 c_{34} - 2Mml^2\dot{q}_3\dot{q}_4 c_{34} - Mml^2\dot{q}_4^2 c_{34} - M^2 lg - Mlgm + Ml\mathbf{f_1}c_3 + ml\mathbf{f_1}s_{34}s_4 - M\boldsymbol{\tau_2}s_{34} - m\boldsymbol{\tau_2}s_{34}}{Ml(M+m)} \\ \frac{\boldsymbol{\tau_1} - \boldsymbol{\tau_2}}{I_z} \\ \frac{-\mathbf{f_1}s_4}{Ml} + \frac{\boldsymbol{\tau_2}}{ml^2} + \frac{\boldsymbol{\tau_2}}{Ml^2} - \frac{\boldsymbol{\tau_1}}{I_z} + \frac{\boldsymbol{\tau_2}}{I_z} \end{bmatrix} \tag{4.4}$$

4.2.3 Collision Planning with a Pogostick

Figure 4.1C depicts the AP-UAS during collision, modeled as a rigid body D (mass M, inertia I_z) with an actuated pogostick represented as a particle p_c (mass m) attached via a massless rod of length l. Generalized coordinates q_1 and q_2 locate the mass center of D (their derivatives yield velocity $\mathbf{v}$, see Fig. 4.1), while the sum $q_3 + q_4$ defines the pogostick's angle relative to vertical. We seek a relationship between the contact impulse $\boldsymbol{\Lambda}$ and the pre-collision states, aligning $\boldsymbol{\Lambda}$ along the pogostick direction. The constraint vector $\mathbf{c_w}$ specifies the position of p_c relative to the world frame origin (contact point)

$$\mathbf{c_w} = \begin{bmatrix} q_1 + ls_{34} \\ q_2 - lc_{34} \end{bmatrix}. \tag{4.5}$$

To describe the configuration at contact, we 'constrain' the particle p_c to contact the ground, yielding a contact Jacobian

$$J_\lambda = \frac{\partial \mathbf{c_w}}{\partial \mathbf{q}} = \begin{bmatrix} 1 & 0 & lc_{34} \\ 0 & 1 & ls_{34} \end{bmatrix}, \tag{4.6}$$

where $\mathbf{q} = [q_1 \ q_2 \ (q_3 + q_4)]^T$

The mass matrix of the system M_D is defined as

$$M_D = \begin{bmatrix} M+m & 0 & mlc_{34} \\ 0 & M+m & mls_{34} \\ mlc_{34} & mls_{34} & ml^2 + I_z \end{bmatrix}. \tag{4.7}$$

The contact space mass matrix M_λ is defined as

$$M_\lambda = (J_\lambda M_D J_\lambda^T)^{-1} \tag{4.8}$$

The impulse $\boldsymbol{\Lambda}$ can be found using

$$\boldsymbol{\Lambda} = M_\lambda(\dot{\mathbf{c}}_\mathbf{w}^+ - \dot{\mathbf{c}}_\mathbf{w}^-), \tag{4.9}$$

where $\dot{\mathbf{c}}_{\mathbf{w}}^{+}$ is the post-contact velocity of p_c and $\dot{\mathbf{c}}_{\mathbf{w}}^{-}$ is the pre-contact velocity of p_c in the world frame. The post-contact velocity $\dot{\mathbf{c}}_{\mathbf{w}}^{+}$ can be expressed as

$$\dot{\mathbf{c}}_{\mathbf{w}}^{+} = e\dot{\mathbf{c}}_{\mathbf{w}}^{-}, \tag{4.10}$$

where e is the coefficient of restitution for the contact. The contact impulse $\mathbf{\Lambda}$ expressed in the generalized coordinate space can be found by combining Eq. 5.8 and Eq. 5.9 (see [62]):

$$\mathbf{\Lambda} = M_{\lambda}(e\dot{\mathbf{c}}_{\mathbf{w}}^{-} - \dot{\mathbf{c}}_{\mathbf{w}}^{-}). \tag{4.11}$$

For a collision without slipping, the post-contact horizontal velocity of p_c is constrained to zero

$$\dot{\mathbf{c}}_{\mathbf{w}}^{+} = e\dot{\mathbf{c}}_{\mathbf{w}}^{-} = \begin{bmatrix} 0 \\ e(\dot{q}_2^{-} + ls_{3^-4^-}(\dot{q}_3^{-} + \dot{q}_4^{-})) \end{bmatrix}. \tag{4.12}$$

Using Eq. 5.10 to find $\mathbf{\Lambda}$ for any given pre-contact states means that we can solve for $q_3^{-} + q_4^{-}$ given $\dot{\mathbf{c}}_{\mathbf{w}}^{-}$ such that $\mathbf{\Lambda}$ and the pogostick are aligned:

$$\mathbf{\Lambda} \times \mathbf{c_D} = 0, \tag{4.13}$$

where $\mathbf{c_D}$ denotes the position vector of p_c relative to the UAS frame D.

Previous work in [48] relied on intuition to determine an ideal contact angle by using a method we called velocity matching. In velocity matching, the contact angle matches the UAS's velocity vector. This approach ensured that the pogostick contacted the environment first, although it did not limit the moment of contact impulse about the CoM.

4.2.4 Trajectory Planning

We determined optimal trajectories for the RP-UAS and AP-UAS using a non-convex optimizer, Interior Point OPTimizer (IPOPT), available in the CasADi python library. Cost functions were built to optimize for shortest flight time T while penalizing high angular velocities $\dot{q}_3$, $\dot{q}_4$ and positive $\dot{q}_2$. The cost functions are presented in Table 5.1 and hard constraints are presented in Table 5.2. In both tables, * denotes that the constraints/costs were only involved in the AP-UAS, $f|i$ denotes planner final states, and the subscript $uf|i$ means a variable is a user-defined final value prior to running the planner.

We imagine that this trajectory planning step would be undertaken prior to run-time, either on the ground or while the UAV is preparing for a collision (perhaps hovering), where and when computational resources are available. Trajectories were generated on a Windows

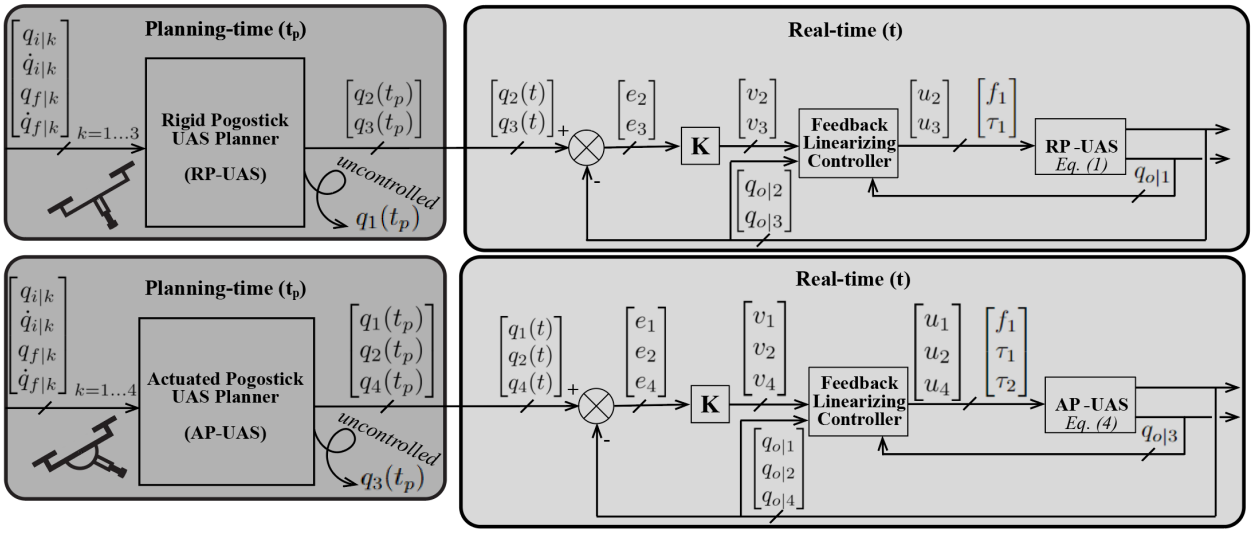

Figure 4.2: Top: Planner and run-time controller (based on partial feedback linearization around $\{q_2,\ q_3\}$) for the RP-UAS (UAS without additional actuator). Bottom: Planner and run-time controller (based on partial feedback linearization around $\{q_1,\ q_2,\ q_4\}$) for the AP-UAS (UAS with actuated pogostick). User-defined initial states q_i and final pre-contact states q_f are fed into the planner, $q(t_p)$ is an output trajectory from the planner (available for all states), $q(t)$ is the reference being tracked by the run-time controller, and q_o are the system output states. Only a subset of the planned states are sampled and tracked by the run-time controller, as indicated by the subscripts. The state variables planned but not tracked are indicated with a pigtail arrow.

Desktop machine equipped with 32 GB of RAM and a 12-Core AMD Ryzen 9 3900XT Processor, and still trajectory generation takes around 20 seconds on average. Given that each flight is typically under 10 seconds, this means that in-flight replanning is impossible and even more so if generated on the Beagle Bone Blue flight controller with its single-core 1GHz ARM Cortex-A8 processor and 512 MB of RAM. There is a possibility to use other methods like explicit MPC, where the calculations are done ahead of runtime but these were not considered given the necessary computational overhead and sensitivity to errors [75]. The controller responsible for tracking the planned trajectory at run-time is designed for maximal computational efficiency and is based on partial feedback linearization. The block diagram in Figure 4.2 shows the distinction between planning and runtime in gray round-corner boxes, highlighting the trajectories $q(t_p)$ as functions of planning time t_p that are sampled during runtime t to produce the reference inputs to the run-time controller. Each system (RP-UAS and AP-UAS) has its own planner which incorporates the dynamics of the respective system.

Table 4.1: Trajectory Generation Costs

Penalizing	Cost Function	Cost Weight
Flight Time T (s)	$w_T T$	1.0
Positive $\dot{q}_2$	$w_{\dot{q}_2} \sum_{k=1}^{N} (max(0, \dot{q}_2))^2$	0.1
*High $\lvert\dot{q}_3\rvert$	$w_{\dot{q}_3} \sum_{k=1}^{N} (\dot{q}_3))^2$	0.05
*High $\lvert\dot{q}_4\rvert$	$w_{\dot{q}_4} \sum_{k=1}^{N} (\dot{q}_4))^2$	0.1

Table 4.2: Trajectory Generation Constraints

Constrained Variable	Constraint Equation
$q_{f\|1}, q_{f\|2}$	$q_{uf\|1}, q_{uf\|2}$
$^*q_{f\|3} + q_{f\|4}$	$q_{uf\|3} + q_{uf\|4}$
$\dot{q}_{f\|1}, \dot{q}_{f\|2}$	$\dot{q}_{uf\|1}, \dot{q}_{uf\|2}$
q_3	$\lvert q_3\rvert \leq 0.75$ rad
$^*q_3 + q_4$	$\lvert q_3 + q_4\rvert \leq 1.5708$ rad
$\dot{q}_1, \dot{q}_2$	$\lvert\dot{q}_1, \dot{q}_2\rvert \leq 2.0$ m/s
$\dot{q}_3$	$\lvert\dot{q}_3\rvert \leq 3.0$ rad/s
$^*\dot{q}_4$	$\lvert\dot{q}_4\rvert \leq 0.75$ rad/s

4.2.5 Partial Feedback Linearization

Feedback linearization uses an inner state feedback loop to transform nonlinear equations of motion into a linear input-output map suitable for linear control design. The outer loop linear feedback controller allows the performance and stability of the system to be determined by design with pole placement. However, underactuated systems cannot be fully feedback linearized. As such, the RP-UAS and AP-UAS were partially feedback linearized.

The partial feedback linearization of the two systems follows the methods outlined in [66]; and while not novel, partial feedback linearization is crucial to effective control of the AP-UAS. In both systems, virtual output equations are generated to specify the states to be tracked. The virtual output equations $h_{2,3}$ for the RP-UAS are

$$h_{2,3}^{RP-UAS} = \begin{bmatrix} q_2 \\ q_3 \end{bmatrix}. \tag{4.14}$$

The approach of partial feedback linearization of the RP-UAS to close an outer control loop around q_1 and q_2 is not possible unless feedback linearization is coupled with dynamic extension [66] [53]. For the AP-UAS, dynamic extension is required to close an outer control loop on q_1, q_2, and $q_3 + q_4$ together. Dynamic extension would then require snap trajectories from the planner, complicating the planning step. Dynamic extension is avoided for the AP-UAS by instead closing an outer control loop on q_1, q_2, and q_4. Thus the virtual outputs

$h_{1,2,4}$ can be defined as

$$h_{1,2,4}^{AP-UAS} = \begin{bmatrix} q_1 \\ q_2 \\ q_4 \end{bmatrix}. \tag{4.15}$$

Virtual inputs for the RP-UAS $v_{2,3}$ and the AP-UAS $v_{1,2,4}$ are defined as linear error equations of state

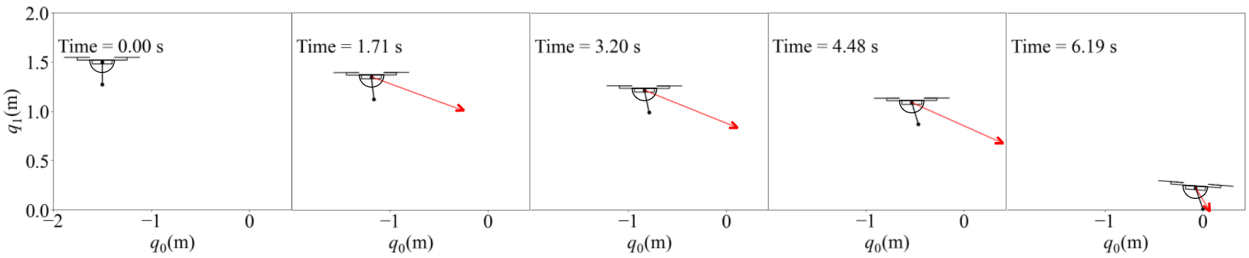

Figure 4.3: Frames from simulation animation for AP-UAS. Red Arrow depicts the velocity vector of the UAS.

$$v_i = \ddot{q}(t)_i - K_{0|q_i}(h_i - q(t)_i) - K_{1|q_i}(L_F h_i - \dot{q}(t)_i), \tag{4.16}$$

where $(t)_i$ denotes desired states, K_0 and K_1 are position and velocity error coefficients. The Lie derivatives of the output equation are $L_F h_i$ where F is from the reformulation of Eq. 4.1 and Eq. 4.4 as $\dot{x} = F(x) + G(x)u$. The virtual input equations are set to

$$y_i^2 = L_F^2 h_i + L_{G_1} L_F^1 h_i \mathbf{f_1} + L_{G_2} L_F^1 h_i \boldsymbol{\tau_1} + L_{G_3} L_F^1 h_i \boldsymbol{\tau_2}, \tag{4.17}$$

and solved such that the virtual inputs can be mapped to real input values for $\mathbf{f_1}$, $\boldsymbol{\tau_1}$, and $\boldsymbol{\tau_2}$. Note that the same mathematical calculations are undertaken for the UAS with the rigid pogostick, but $\boldsymbol{\tau_2}$ is excluded.

The partial feedback linearization of the two systems, RP-UAS and AP-UAS are shown in block diagram form in Figure 4.2. Using partial feedback linearization allows the response of the controller to be tuned using pole-placement of the gains $K_{0|q_i}$ and $K_{1|q_i}$ in Eq. 4.16. These equations are equivalent to

$$0 = \ddot{e} - K_1 \dot{e} - K_0 e = (s^2 + \alpha s + \alpha^2) E(s), \tag{4.18}$$

where $K_1 = -\alpha$ and $K_0 = -\alpha^2$. Choosing gains that ensure the poles are in the left half-plane will provide a stable controller. For both controllers, poles were placed at -8 for a stable and fast response which was tuned manually.

4.2.6 Zero Dynamics of Each System

Given that both systems feature partial feedback linearizing controllers, there are states which are unobservable for the controller. For the RP-UAS q_1 and $\dot{q}_1$ are the unobservable states and for the AP-UAS q_3 and $\dot{q}_3$. To ensure system stability, we must analyze the zero dynamics of the partially feedback linearizing controller. We compute the zero dynamics by taking the dynamical equations of each system (Eq. 4.1 and Eq. 4.4) and substituting the control inputs $\mathbf{f_1}$, $\boldsymbol{\tau_1}$, and $\boldsymbol{\tau_2}$ (for AP-UAS) with the feedback linearizing control law derived from Eq. 4.16 and Eq. 4.17, while setting the tracking errors to zero ($q_i = q_i(t)$).

We found the following zero dynamics for the partial feedback linearizing controller of the RP-UAS:

$$\ddot{q}_1 = (-g - \dot{q}_2)\tan(q_3). \tag{4.19}$$

We found the following zero dynamics for the partial feedback linearizing controller of the AP-UAS:

$$\ddot{q}_3 = -\dot{q}_4 + \tan(q_4)(\dot{q}_4 + \dot{q}_3)^2 - \frac{(M+m)(g\,s_3 + \dot{q}_1\,c_3 + \dot{q}_2\,s_3)}{L\,m\,c_4}. \tag{4.20}$$

We linearized both systems around the fixed point where all q_i and $\dot{q}_i$ are equal to zero. This linearization reveals that the A matrices for both systems' zero dynamics have eigenvalues $\lambda = 0$, suggesting that the zero dynamics are marginally stable. Although they will not diverge, we cannot guarantee convergence; thus, we expect the system to oscillate about an equilibrium point.

4.3 Simulation Results

4.3.1 Simulation Environment

For testing the control of the full system, a 2-D simulation environment was created in Python. This simulation handled flight dynamics and controls of the system. The candidate physical system for testing the partial feedback linearizing control of the AP-UAS will be run on a Beagle Bone Blue. The Beagle Bone Blue operates with a 400 Hz control interval which is proven to be effective for multirotor flight control [42]. To mirror the candidate flight system, the simulator was run with the same 400 Hz control interval. The dynamics of the system were computed via Runge-Kutta integration. The physical parameters of the simulation are modeled after the DJI F550 Hexacopter from our previous work in [48]. This system had a total mass of 1.687 kg, with the pogostick modeled as a 0.1 kg point mass, as stated previously, the length of the pogostick l was 0.2286 m, and rigid body D had moment

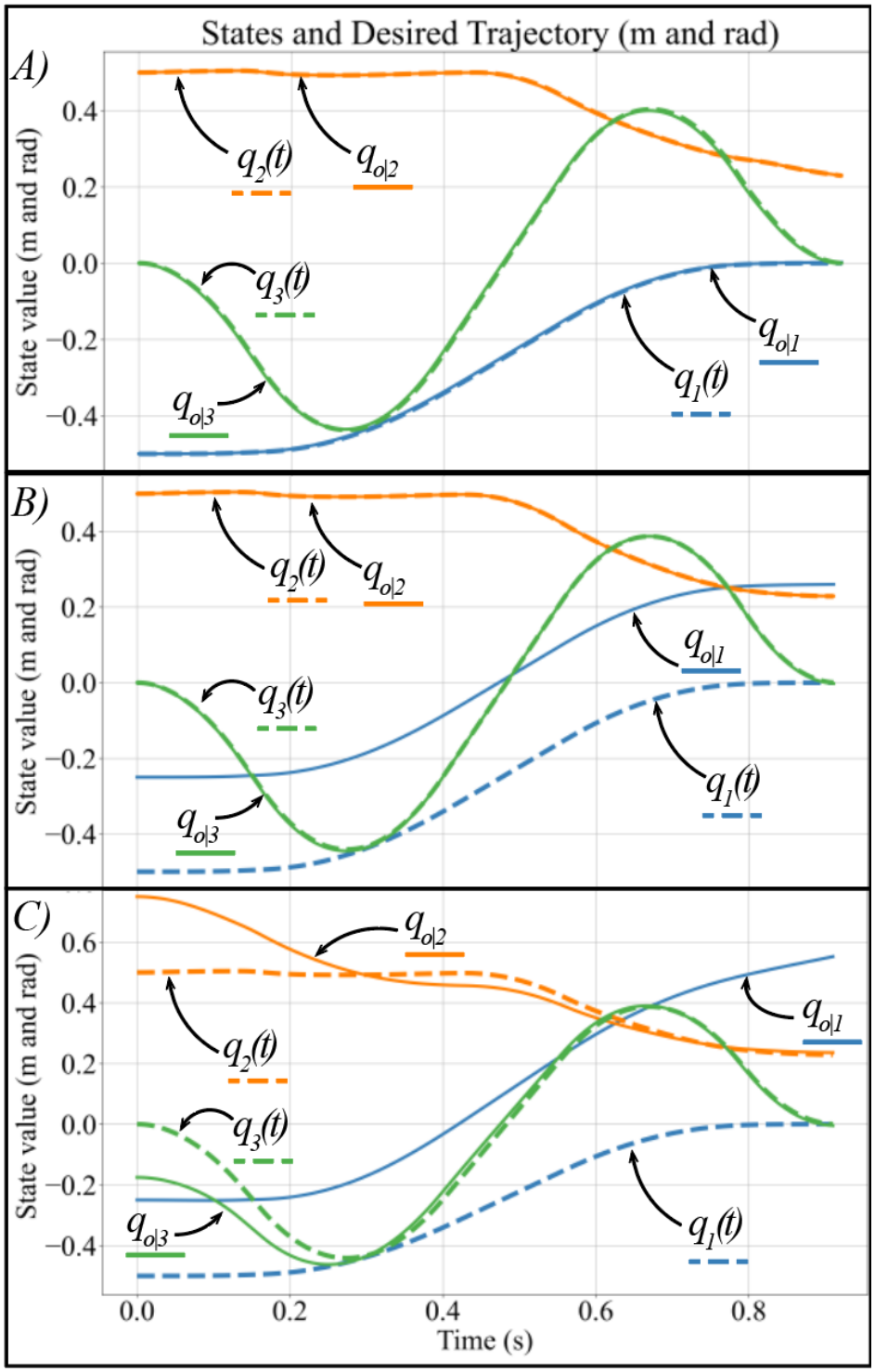

Figure 4.4: RP-UAS partial feedback linearizing controller performance. A) No initial state errors, B) Initial error of 0.25 m in q_1, and C) Initial error of 0.25 m in q_1 and q_2 and error of -0.17533 rad in q_3.

of inertia of 0.174 kg m^2. During flight, there were no external disturbances.

4.3.2 Rigid Pogostick UAS Simulation Results

Simulations were run with three different initial state configurations to test the performance of the RP-UAS's partial feedback linearizing controller. The three different initial state configurations were run on the same trajectory generated in our planner. The trajectory and controller performance for all three initial state configurations can be seen in Figure 4.4. Note that the planner trajectory is labeled as $q_i(t)$ and actual simulation trajectories are labeled as $q_{o|i}$.

The resultant trajectories show that the partial feedback linearizing controller can make errors in q_2 and q_3 converge regardless of initial error, but convergence is not guaranteed for q_1 if initial errors are present.

4.3.3 Actuated Pogostick UAS Simulation Results

Similarly to the RP-UAS, simulations were run with three different initial state configurations to test the performance of the AP-UAS's partial feedback linearizing controller. The three different state configurations were run on the same trajectory generated in our planner, the trajectory and controller performance for all three initial state configurations can be seen in Figure 4.5. Note that the planner trajectory is labeled as $q_i(t)$ and actual simulation trajectories are labeled as $q_{o|i}$.

The resultant trajectories for the AP-UAS show that the partial feedback linearizing controller can make errors in q_1, q_2, and q_4 converge regardless of initial error, but convergence is not guaranteed for errors in q_3. Nevertheless, q_3 has bounded oscillations around the desired q_3 trajectory ($\pm 3^\circ$ in this example).

A visualization of the trajectory being flown by the AP-UAS can be seen in Figure 4.3. Figure 4.6 presents the summary results for testing the AP-UAS partial feedback linearizing controller. In these simulations, we added an offset to one state variable and recorded the final error of that state at strike. The figure shows that the controlled states maintain very low errors—the highest being q_2 with a maximum error of 0.007 m. The pitch q_3 curve exhibits errors varying between $0 - 0.07$ rad. These oscillations match those observed in the sample results and indicate that the final error in q_3 depends on the timing of the strike as q_3 oscillates.

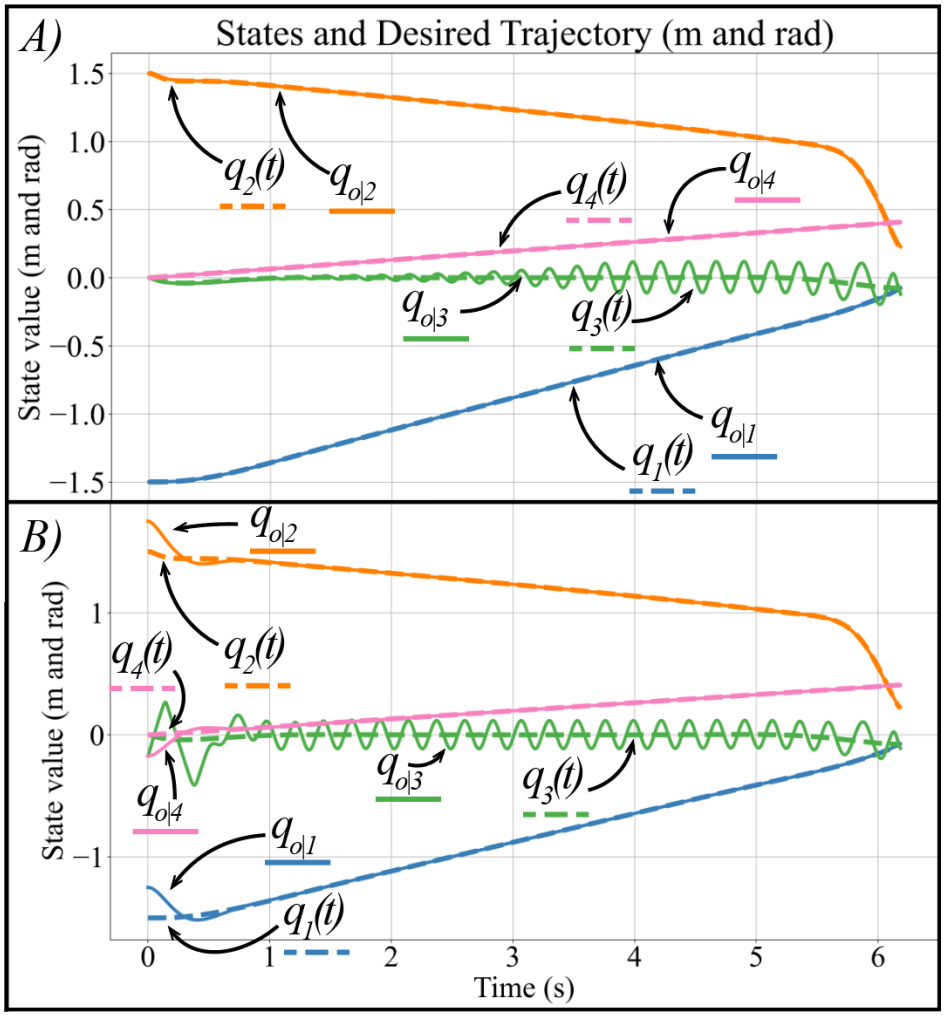

Figure 4.5: AP-UAS partial feedback linearizing controller performance. A) No initial state errors and B) Initial error of 0.25 m in q_1 and q_2 and error of -0.17533 rad in q_3 and q_4.

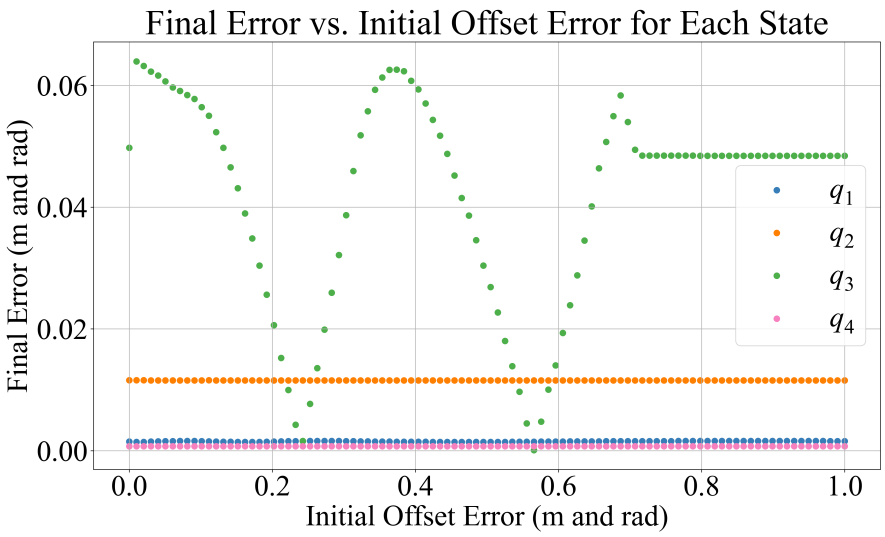

Figure 4.6: Plot of final error in q_i at strike based on initial error in q_i $(i = 1, ..., 4)$ at the start of the trajectory. Each state was tested with an initial error while the other states started with zero error.

4.4 Discussion and Conclusion

We have presented two distinct UAS models in 2-D for achieving collision-based aerial manipulation. We argue that adding articulation and an actuator to the UAS's pogostick allows for tracking on both positions and final contact angle during a collision when coupled with a partial feedback linearizing controller. Controlling both of these values is imperative to have a safe and recoverable collision (one that does not impart changes in angular momentum).

We demonstrated that partial feedback linearization in the RP-UAS does not solve the underactuated system's limitation to control both position and attitude at contact. Any initial errors in a state cause divergence in the untracked state q_1. Importantly, when adding an additional actuator, the partial feedback linearizing approach is designed to close a loop around the pogostick angle q_4 and bound the pitch angle q_3 to oscillate around the desired pitch angle q_3 with low error. The experimental investigation and characterization of the q_3 oscillations will be completed in future work. Indirect tracking of q_3 is due to the dynamics of the system coupling $\ddot{q}_1$ and $\ddot{q}_2$ with q_3 in Eq. 4.4. If tracking is done perfectly, the pitch angle q_3 of the UAS averages to the the value for q_3 from our optimal trajectory planner. Otherwise, deviations in q_3 would appear as deviations in q_1 and q_2. By directly tracking the angle of the actuated arm q_4 and indirectly tracking q_3, the system can reliably collide with a desired target position and contact angle $q_3 + q_4$. The added control means that collisions

can be planned to leverage different collision dynamics that are necessary for specific aerial manipulation tasks.

The goal of this work was to build upon the system presented in [48] and create a more effective and stable flight controller for bouncing with a pogostick payload. The previous method could not fully control position and prioritized the pogostick angle. Using an additional actuator and a partial feedback linearizing controller, we believe that both collision contact location and angle can be controlled reliably. Direct follow-up work for this project will be centered on conducting experimental tests with a physical system in our laboratory to corroborate the findings from these simulation studies, as well as incorporating disturbances in the simulation and experimental results to analyze how performance is effected. Subsequent work will be focused on developing manipulators to be fixed on the end of the pogostick for generic aerial manipulation and more specifically rainforest tree sampling (target application in [48]).

CHAPTER 5

Aerial Manipulation via Intentional Collisions with a Two Degree of Freedom Pogostick

5.1 Introduction

We are entering a new age of robotics in which humanoid robots [20], autonomous cars [54], and Uncrewed Aircraft Systems (UAS) [52] are commonplace. These robots are being built, and in some cases learning [12], to replace humans in unsafe and dangerous tasks. In particular, UAS are excellent at operating in environments that are difficult or hazardous for humans to access directly. Such capabilities are relevant to tasks like inspection [59], rainforest tree sampling (the motivating task behind this research) [48], and interacting in cluttered or risky environments [68].

In this work, we are interested in using a UAS not just as a sensor platform, but as a tool that physically interacts with its environment, and doing so while keeping the vehicle as close as possible to a conventional, low-cost hexacopter. This is essential for the motivating task of rainforest tree sampling, where tree branches must be cut and grasped while keeping the system cheap and simple.

Interactions using UAS can be broadly grouped into three categories: interaction via detachable payloads (payloads that leave the UAS before interacting) [21], post-flight interaction (interaction after the UAS has finished flying) [56], and in-flight interaction (interaction while the UAS is flying) [37]. In this work, we define Aerial Manipulation as in-flight interaction via UAS and specifically focus on leveraging intentional collisions to simplify the interaction between the UAS and its environment into a single impulse. This builds directly on prior work that models these interactions as impulsive collisions [48] and shifts collision planning into the trajectory design process [47].

Intentional collision-based interaction introduces challenges in both control and execution of the strikes. Collision responses are functions of the pre-collision state, namely the velocity, angular velocity, and orientation. Position states also matter when a collision needs to strike a

target. The system considered here is a traditional hexacopter with six parallel-facing motors [5]. As a result, the vehicle is underactuated: it effectively has one body force and three body torques to control all six states (positions and velocities) of the UAS. This underactuation means that directly controlling all states needed to define a targeted intentional collision is not possible with a traditional hexacopter design.

Other UAS have been designed with tiltable rotors to provide omnidirectional forces [8]. These systems essentially allow the UAS to point and fly in any direction, which is ideal for aerial manipulation via intentional collisions, but requires departing from the traditional hexacopter architecture and adds mechanical complexity. Additional research using Model Predictive Control (MPC) [44] and trajectory replanning [43] exploits the system dynamics and optimization to generate control inputs that drive the UAS toward a desired state (positions and velocities). While these methods do not provide true, independent control over all system states, they often achieve acceptable performance. However, MPC and trajectory replanning are computationally expensive, typically requiring high-end onboard compute or an offboard computer performing the optimization [43]. This leads to a more complex and costly overall system, limiting the applications in which such a UAS is practical.

In parallel, a variety of mechanical approaches have been explored for orientation control, namely spherical linkage mechanisms [1] and pointing devices like a gimbal [3]. This work investigates adding additional degrees of freedom via a 2 Degree-of-Freedom (DoF) pogostick to enable orientation compensation, and serves as a direct follow-up to [47]. The 2 DoF pogostick is designed so that the center of rotation of the pogostick coincides with the Center-of-Mass (CoM) of the UAS. This simplifies collision planning because any impulse applied along the pogostick does not impart a disturbance moment about the UAS. Enforcing this constraint requires a spherical mechanism, where all motion is about a specified center of rotation. Spherical mechanisms are well studied, with gimbals [78] and spherical four-bar linkages [46] as classic examples. The main limitation of these mechanisms is that the center of rotation must be physically connected to the structure, meaning it cannot be displaced away from the gimbal or linkage itself. Given the typical construction of a hexacopter, implementing such mechanisms with a center of rotation at the UAS CoM is not feasible without a complete airframe redesign.

Other systems, such as delta robots, can mimic a displaced center of rotation, but they rely on connected arms for orientation control [63, 67]. This creates issues during collisions, as the arms must both withstand impact and maintain their orientation throughout the strike. Due to these limitations, a new mechanism has been built.

In the following, we propose a cable-driven, two DoF pogostick to enable Aerial Manipulation via intentional collisions. The design focuses on mitigating alignment errors at impact

on a pogostick-enabled hexacopter by allowing the actuated two DoF pogostick to compensate for orientation errors. The proposed mechanism uses a truncated sphere along which the pogostick base can slide, yielding a mechanism that shares a common center of rotation with the UAS center of mass and thereby simplifying collision response planning. Pre-collision trajectories are generated using an optimal trajectory planner so that only minimal adjustments of the pogostick attitude are needed at impact. By colocating the pogostick's center of rotation with the vehicle CoM, the mechanism offloads orientation correction at impact from the underactuated vehicle to the actuated pogostick: the hexacopter focuses on position control, while the pogostick handles residual orientation error just before collision.

The contributions of this work include: A novel cable-driven, two DoF pogostick mechanism whose center of rotation is colocated with the UAS center of mass via a truncated-sphere, enabling orientation compensation for intentional aerial collisions while preserving a traditional hexacopter layout. A collision-planning framework that treats intentional impacts as impulsive interactions and leverages the pogostick mechanism to decouple impact orientation from the underactuated vehicle dynamics. An optimal pre-collision trajectory planning scheme that aligns the UAS body frame with a desired strike angle, reducing the necessary motion required from the pogostick at impact. Experimental results demonstrating that the proposed system can reliably achieve intentional collision orientations prior to strike.

5.2 Physical Design

The physical design of this system is highlighted in Figure 5.1 and represents an evolution of the design shown in Figure 2.6. The current pogostick design allows for control of pogostick attitude to counteract pitching changes that occur during normal flight, similar to the theoretical payload described in [47]. The primary difference in dynamics between these two designs lies in the actuation of the pogostick itself.

The previously proposed theoretical pogostick relied on a direct torque source to control its angle. While this simplified the design for the partial feedback linearized controller, it proved difficult to implement in practice.

A major constraint that drove the design change was the requirement for the pogostick to be aligned with the center of mass (CoM) of the UAS. This alignment ensures that when a collision occurs and the impulse is aligned with the long axis of the pogostick, the forces acting on the pogostick do not induce unwanted moments on the UAS. This design also simplifies collision response planning, since variables such as the distance l from the UAS CoM to the contact point p_c remain constant, as seen in Fig. 5.2. Because the CoM of

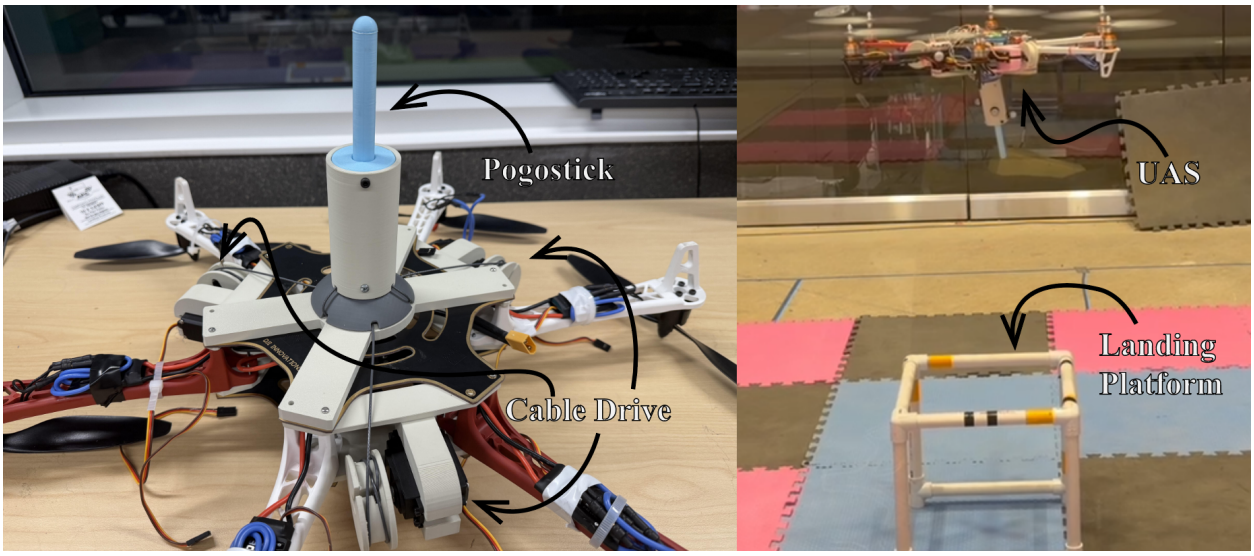

Figure 5.1: Image of cable-driven pogostick payload, the UAS in flight, and the landing platform used for testing. The pogostick and cable tensioners are labelled. Three cable tensioners provide the two degrees of freedom by allowing the pogostick to slide along the hemispherical base.

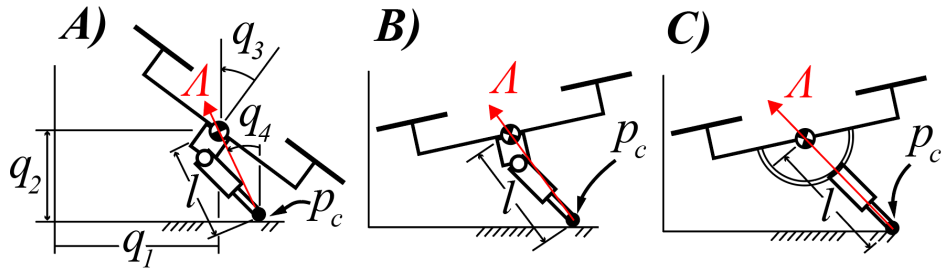

Figure 5.2: A and B) Collision using an actuated pogostick where the center of rotation is not aligned with the CoM of the UAS. The varying distance l from the point of contact p_c to the UAS CoM depends on pitch q_3 and strike angle q_4, which complicates collision planning at a targeted strike location (q_1, q_2). C) Collision using an actuated pogostick where the center of rotation is aligned with the UAS CoM. In this case, l remains constant, which simplifies collision planning.

the UAS is offset from the available mounting points, traditional linkage designs became impractical, especially given the need for a lightweight mechanism capable of withstanding impact forces.

Our design incorporates a truncated hemispherical base centered at the UAS CoM. The hemisphere provides a gliding surface for the pogostick, and the interface between the pogostick and the base shares the same radius. This equality of radii ensures that the pogostick remains perpendicular to the base and thus aligned with the UAS CoM at all times.

To control this design, we installed three cable-drum-equipped MG995 servo actuators in the free space between the propeller arms, spaced 120° apart. These actuators provide full 3-DoF control of the pogostick orientation by tightening or slackening the cables as needed. Importantly, the actuators are arranged so their combined mass is aligned with the UAS CoM, which prevents any significant shift in the overall CoM after installation. The servo controller runs on a Beaglebone Blue that operates independently from the Beaglebone Blue used as the flight controller.

Tensioning of the cables was done manually. A level was used to set the zero position, and then to sweep the maximum angular displacements that needed to be mapped to motor commands.

The hexacopter itself is built on a DJI F550 frame with the added pogostick. This hexacopter uses 30 A ESCs, 1000 KV 2826 brushless motors, and 10×3.8 APC Slow Flyer propellers. The open-source flight controller, (rc_pilot_a2sys), runs at 400 Hz on a Beaglebone Blue. A Vicon motion capture system, running at 120 Hz, provides ground truth pose for experiments.

5.3 System Dynamics

5.3.1 Planar Hexacopter Dynamics

Figure 5.3 serves as a visual guide for the dynamics of a typical planar UAS. This UAS is modeled with a single net thrust $\mathbf{f}$ and one body torque $\boldsymbol{\tau}$ (there would be three body torques in 3D). The system dynamics of the UAS are

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} = \begin{bmatrix} \frac{-\mathbf{f}\sin q_3}{M} \\ \frac{\mathbf{f}\cos q_3}{M} - g \\ \frac{\boldsymbol{\tau}}{I_z} \end{bmatrix}, \tag{5.1}$$

where M and I_z are the mass and moment of inertia of the system, respectively.

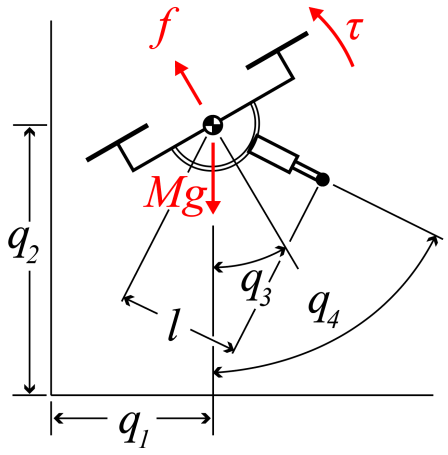

Figure 5.3: 2-D free body diagram of the proposed hexacopter system.

Due to limited actuator authority, this system is underactuated. From the equations of motion it is clear that the positions, both horizontal position q_1 and vertical position q_2, depend on the overall thrust $\mathbf{f}$ and the current pitch q_3 of the UAS. This limits control to either of two possibilities:

$$q(t) = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} q_2 \\ q_3 \end{bmatrix}. \tag{5.2}$$

Previous methods in Chapter 3 required trajectories that incorporated both attitude and position to control all states. With the addition of the actuated pogostick, explicit attitude trajectories are no longer needed. This changes our final trajectories to position and yaw waypoints (following standard conventions) in 3D,

$$w_{conventional} = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix}. \tag{5.3}$$

The pogostick is assumed to be lightweight enough that movements of the pogostick will be assumed to have minimal impact on the UAS dynamics.

5.3.2 Collision Response

Figure 5.4 depicts the UAS during a rigid collision, following the formulation in [62]. Generalized coordinates q_1 and q_2 locate the CoM of the UAS, and their derivatives yield the translational velocity $\mathbf{v}$. The angle q_4 defines the pogostick angle relative to vertical. The goal of collision planning is to find an impulse $\mathbf{\Lambda}$ and corresponding pre-collision states such

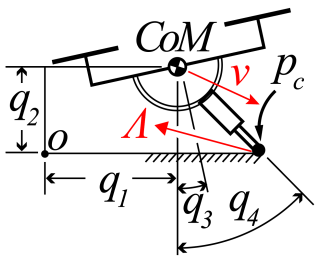

Figure 5.4: 2-D free body diagram of collision response.

that $\mathbf{\Lambda}$ is aligned with the pogostick. This alignment ensures that the UAS does not experience disturbance moments during contact. The constraint vector $\mathbf{c}_w$ specifies the position of p_c relative to the world frame origin,

$$\mathbf{c}_w = \mathbf{CoM} + \mathbf{c}_D = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} ls_4 \\ lc_4 \end{bmatrix} = \begin{bmatrix} q_1 + ls_4 \\ q_2 - lc_4 \end{bmatrix}, \tag{5.4}$$

where $\mathbf{c}_D$ constrains the pogostick orientation at strike.

To describe the configuration at contact, we constrain the particle p_c to contact the ground, which yields the contact Jacobian

$$J_\lambda = \frac{\partial \mathbf{c}_w}{\partial \mathbf{q}} = \begin{bmatrix} 1 & 0 & lc_4 \\ 0 & 1 & ls_4 \end{bmatrix}, \tag{5.5}$$

where $\mathbf{q} = [q_1 \ q_2 \ (q_4)]^T$.

The mass matrix of the system M_D is

$$M_D = \begin{bmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & I_z \end{bmatrix}, \tag{5.6}$$

and the contact-space mass matrix M_λ is

$$M_\lambda = \left(J_\lambda M_D J_\lambda^T \right)^{-1}. \tag{5.7}$$

The contact impulse $\mathbf{\Lambda}$ is

$$\mathbf{\Lambda} = M_\lambda \left(\dot{\mathbf{c}}_\mathbf{w}^+ - \dot{\mathbf{c}}_\mathbf{w}^- \right), \tag{5.8}$$

where $\dot{\mathbf{c}}_\mathbf{w}^+$ and $\dot{\mathbf{c}}_\mathbf{w}^-$ are the post- and pre-contact velocities of p_c in the world frame. The

post-contact velocity can be written as

$$\dot{\mathbf{c}}_{\mathbf{w}}^{+} = e\,\dot{\mathbf{c}}_{\mathbf{w}}^{-}, \tag{5.9}$$

where e is the coefficient of restitution for the contact. Combining Eq. 5.8 and Eq. 5.9 yields [62]

$$\mathbf{\Lambda} = M_{\lambda}\left(e\,\dot{\mathbf{c}}_{\mathbf{w}}^{-} - \dot{\mathbf{c}}_{\mathbf{w}}^{-}\right). \tag{5.10}$$

For a collision without slipping, the post-contact horizontal velocity of p_c is constrained to zero,

$$\dot{\mathbf{c}}_{\mathbf{w}}^{+} = e\,\dot{\mathbf{c}}_{\mathbf{w}}^{-} = \begin{bmatrix} 0 \\ e\left(\dot{q}_2^{-} + ls_4\left(\dot{q}_4^{-}\right)\right) \end{bmatrix}. \tag{5.11}$$

Using Eq. 5.10 to compute $\mathbf{\Lambda}$ for any given pre-contact state lets us solve for q_4^{-}, given $\dot{\mathbf{c}}_{\mathbf{w}}^{-}$, and given the requirement that $\mathbf{\Lambda}$ and the pogostick are aligned:

$$\mathbf{\Lambda} \times \mathbf{c}_D = 0, \tag{5.12}$$

where $\mathbf{c}_D$ denotes the position vector of p_c relative to the UAS center of mass.

5.4 Hexacopter Controller Design

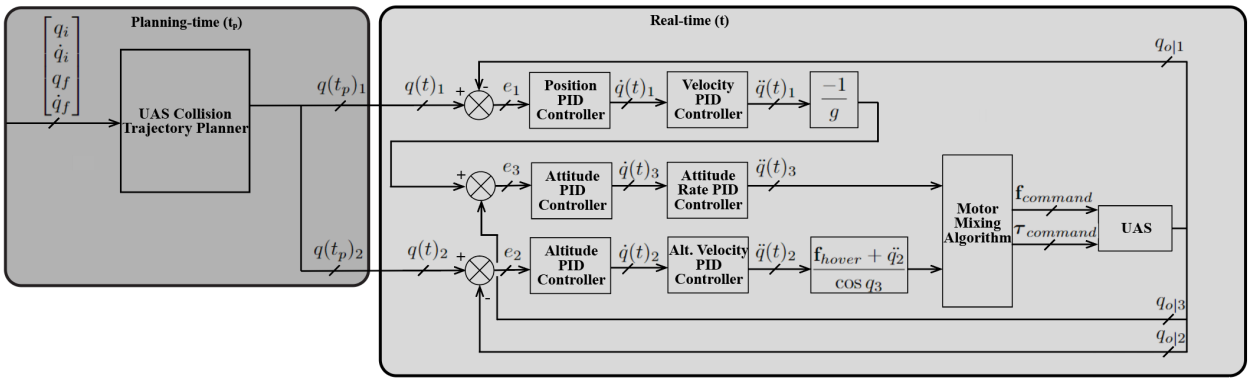

Figure 5.5: Block diagram outlining the hexacopter portion of planning and control. The user specifies a final strike location and desired strike velocities. The planner then creates a dynamically feasible trajectory from the initial UAS location to the target strike location using optimization. This trajectory is then fed into a cascaded PID controller to command position commands into motor thrusts.

The hexacopter's overall flight control is summarized in Fig. 5.5. It consists of two main components, an offline trajectory planner and an online cascaded PID controller that tracks

the generated trajectories.

5.4.1 Optimal Trajectory Planner

Flight trajectory generation uses the CasADi Python library with the Interior Point OPTimizer (IPOPT) to produce optimal, dynamically feasible trajectories. The trajectory generator takes an initial offset from the strike location and desired strike characteristics, including position, velocity, and strike angle q_4, and generates a trajectory that minimizes flight time and overall control effort. Because the actuated pogostick is intended to correct residual alignment errors, the desired strike orientation is passed into the optimizer. The optimizer attempts to align the final orientation of the UAS q_3 with the desired strike angle q_4. As a result, the UAS flies a trajectory that ends with an orientation close to the desired one, which reduces the required change in pogostick orientation.

The costs used in the trajectory optimization are summarized in Table 5.1, and the constraints are listed in Table 5.2. Trajectories are generated offline so the computationally limited Beaglebone Blue flight controller does not need to run the optimizer in real time.

Table 5.1: Trajectory Generation Costs

Penalizing	Cost Function	Cost Weight
Flight Time T (s)	$w_T\, T$	1.0
Positive $\dot{q}_2$	$w_{\dot{q}_2} \sum_{k=1}^{N} \left(\max(0, \dot{q}_2)\right)^2$	0.1
Control Effort U	$w_U \sum_{k=1}^{N} U^2$	0.0001

Table 5.2: Trajectory Generation Constraints

Constrained Variable	Constraint Equation
$q_{f\|1}$, $q_{f\|2}$, $q_{f\|3}$	$q_{uf\|1}$, $q_{uf\|2}$, $q_{uf\|3}$
$\dot{q}_{f\|1}$, $\dot{q}_{f\|2}$	$\dot{q}_{uf\|1}$, $\dot{q}_{uf\|2}$
q_3	$\|q_3\| \leq 0.75$ rad
$\dot{q}_1$, $\dot{q}_2$	$\|\dot{q}_1\|$, $\|\dot{q}_2\| \leq 2.0$ m/s
$\dot{q}_3$	$\|\dot{q}_3\| \leq 3.0$ rad/s

5.4.2 Cascaded PID Control

From Fig. 5.5, we see that the UAS uses a traditional cascaded PID controller, illustrated here in 2D for clarity. The cascaded structure takes position waypoints as inputs. Horizontal position commands feed into velocity loops, which then feed into attitude loops and finally into attitude rate loops. Altitude commands cascade into altitude velocity control. This

cascaded architecture allows the UAS to follow the pre-generated trajectories accurately, without modifications to standard UAS flight control approaches.

5.5 Pogostick Controller Design

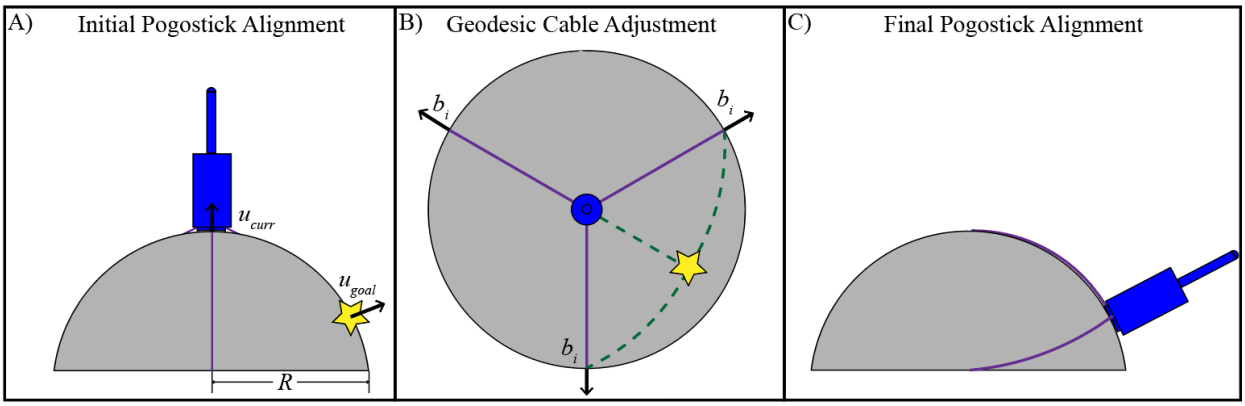

Figure 5.6: Geodesic-based control of the cable-driven pogostick. A) the pogostick is shown in blue, cables in purple. B) the yellow star marks the desired goal orientation, and the green dashed curves show the geodesic cable paths that align the pogostick with the new goal state. C) the cables lie along the geodesic curves intersecting at the goal position.

The controller for the cable-driven pogostick does not follow a typical PID or cascaded design. Because the servos act as position sources instead of torque or velocity sources, the control problem reduces to finding the servo positions required for the pogostick to point in a desired direction. Figure 5.6 provides a visual illustration of this idea. The three cables are equally spaced by 120°. Since the cables run close to the spherical surface, we assume they lie on that surface with minimal separation. This assumption allows us to use geodesics, the shortest paths on the hemisphere, to compute the required cable lengths [16].

Let the desired pointing direction be the unit normal vector $u \in \mathbb{R}^3$ on a sphere of radius R, and let the three cable inlet directions on the edge of the hemisphere be unit vectors b_i separated by 120°. On the hemisphere, the angle $\alpha_i(u)$ between u and inlet b_i is

$$\alpha_i(u) = \arccos(u^\top b_i). \tag{5.13}$$

The corresponding geodesic distance $\ell_i(u)$ from u to b_i along the surface is

$$\ell_i(u) = R\,\alpha_i(u) = R\,\arccos(u^\top b_i). \tag{5.14}$$

Because the cable is inextensible and hugs the surface, this $\ell_i(u)$ represents the cable length

from the inlet to the contact point for orientation u. To move from the current orientation u_{cur} to the goal u_{goal}, the required change in cable i is the difference in geodesic lengths,

$$\Delta\ell_i = \ell_i(u_{\text{goal}}) - \ell_i(u_{\text{cur}}) = R\Big[\arccos(u_{\text{goal}}^\top b_i) - \arccos(u_{\text{cur}}^\top b_i)\Big]. \tag{5.15}$$

With identical servos and an effective spool radius r, this maps directly to a servo position change,

$$\Delta\theta_i = \frac{\Delta\ell_i}{r}. \tag{5.16}$$

All three servos are driven synchronously over the same movement duration, which provides coordinated control of the pogostick attitude at any given point in time. Cable-specific constants for tensioning and slackening are tuned per cable to ensure that the cables stay taut across the entire reachable hemisphere.

5.6 Combined System Design

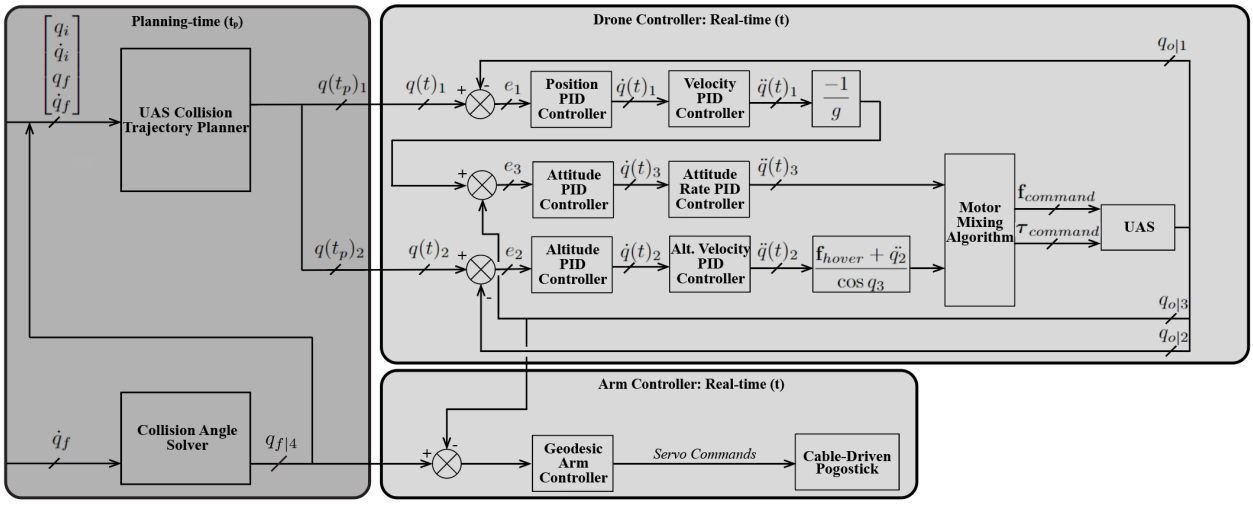

Figure 5.7: Block diagram showing how collision response planning influences trajectory generation and how both outputs feed into the pogostick and UAS controllers. Trajectory generation starts with a desired strike velocity vector $\dot{q}_f$, which is then fed into a collision angle solver to determine the appropriate strike angle. After the desired collision angle is found, that angle as well as the desired pre-impact states are fed into the UAS Collision Trajectory Planner, where a dynamically feasible trajectory is created to get to the desired states.

The combined system follows the architecture shown in Fig. 5.7. The UAS controller is independent from the pogostick controller. In this architecture, the desired strike angle $q_{4|f}$

is determined from the collision response formulation presented earlier. This collision angle solver takes the desired pre-impact velocities, and then solves for a contact angle $q_{4|f}$ that aligns the impulse with the pogostick for the given pre-impact velocities. These desired state values and contact angle are passed to the trajectory generator, which produces trajectories from a user-specified initial state that are then sent to the UAS flight controller.

The pogostick controller bypasses the trajectory generator and instead monitors $q_{4|f}$ and the UAS orientation at every timestep. It applies corrections so that the world-frame strike orientation $q_{4|world}$ remains aligned with the desired strike orientation $q_{4|f}$.

5.7 Experimental Methods

Experimental testing was completed in an indoor fly lab. This three-story flight space is instrumented with Vicon motion capture cameras spanning all floors. We evaluated the system in three primary ways, pre-flight mechanism testing, hover testing with the pogostick moving, and trajectory tracking tests with and without pogostick actuation.

In-flight testing served two purposes. First, to see how the motion of the pogostick affected the flight of the UAS, and second, to determine how accurately the pogostick could compensate for UAS orientation errors at strike. Strike tests were repeated using a virtual floor at 1 m above the real floor, so the vehicle "strikes" in midair. This virtual floor allows us to test the limits of the mechanism before performing actual physical strikes and it keeps ground effect negligible.

5.8 Results

The following table provides experimentally determined specifications of the cable-driven pogostick system when attached to the UAS.

Table 5.3: Cable-driven pogostick specifications

Total Pogostick UAS Mass	2.254 kg
Max Angular Deflection	25° from neutral
Max Angular Velocity	400°/s
Max Angular Resolution	0.46° pogostick angle/1° servo rotation

We also tested whether motion of the cable-driven pogostick impacts flight performance during hover and trajectory tracking. Figure 5.8 shows the UAS position and pogostick orientation during a hover, both with and without pogostick motion. The pogostick moves

at the time stamps with the pink overlay, while it is held stationary at its last position for the remaining time. The plots show no noticeable degradation in position tracking or hover performance when the pogostick is moving compared with when it is stationary. This suggests that pogostick motion has relatively low impact on the UAS dynamics. Figure 5.9 shows that the motor commands also remain relatively unchanged when the pogostick is moving.

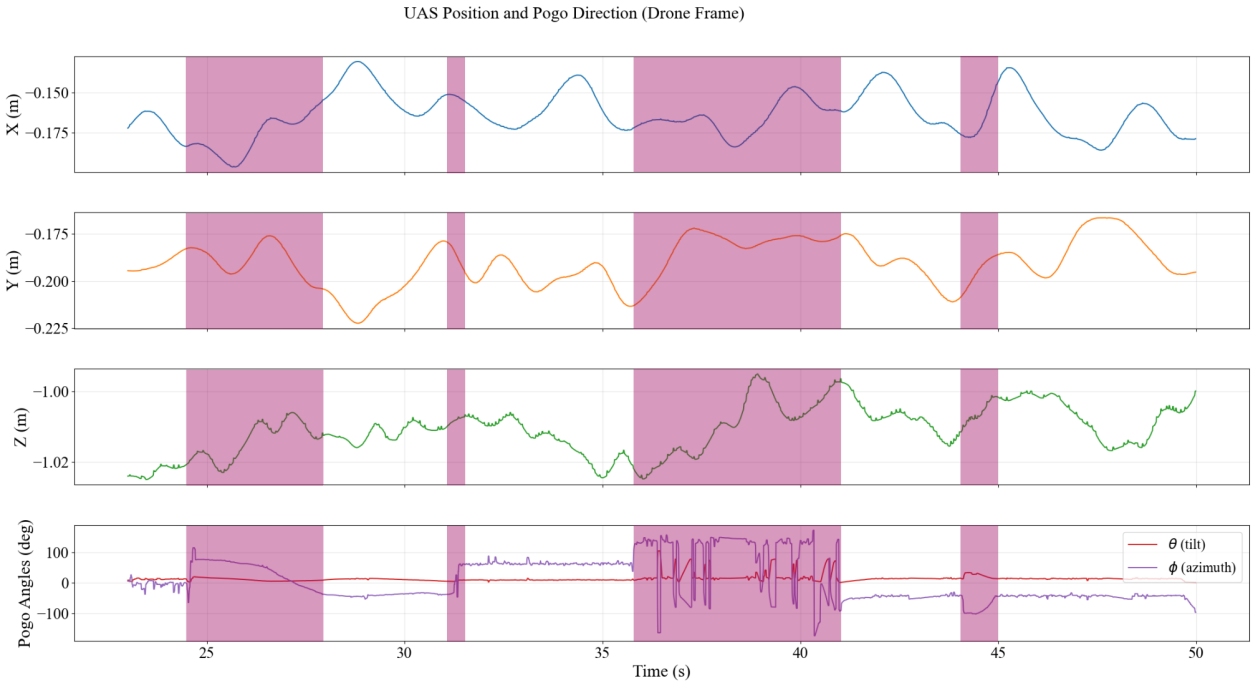

Figure 5.8: Results from hover testing while the pogostick is moving. The plots show UAS x, y, and z position and pogostick tilt and azimuth angles around the hemisphere. The pink overlayed boxes denote when the pogostick is moving.

Table 5.4: Averaged Root Mean Square Error for 25 Flights with and without Pogostick Actuation

RSME	X (m)	Y(m)	Z (m)	Pitch (deg)	Roll (deg)	Yaw (deg)
No Pogo Actuation	0.104	0.030	0.115	0.025	0.074	0.013
Pogo Actuation	0.090	0.039	0.113	0.023	0.073	0.012
RMSE Change	**-0.014**	**0.009**	**-0.002**	**-0.002**	**-0.001**	**-0.001**

Table 5.4 provides summary results from 25 flight tests that also show that changes in Root Mean Squared Error (RMSE) in position and orientation from when the pogostick is actuated versus not are minimal. This error is the total RMSE along the full trajectory, when compared to the setpoints that are trying to be tracked. The maximum average difference

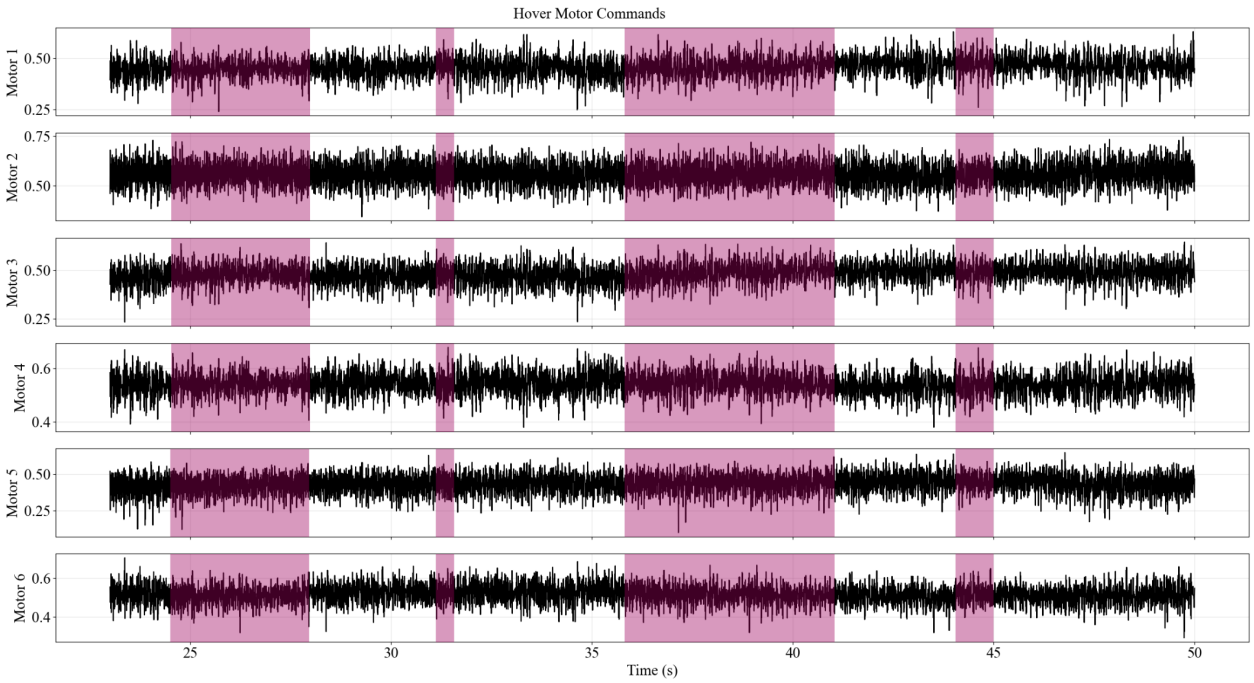

Figure 5.9: Motor commands during hover testing with and without pogostick motion. The pink overlayed boxes denote when the pogostick is moving.

in position error was 14 mm and the maximum average difference in orientation error was 0.002°. Both of these errors are small and well within the typical errors from the flight controller regardless of if the arm is present, which can be seen in the pose variation even during a hover in Fig. 5.8.

Figure 5.10 shows a representative strike trajectory flown with the pogostick actuated and with the pogostick unactuated. This sample shows how the pogostick actuation affects trajectory tracking performance. The trajectory overlays coupled with the information from 5.4 indicate that any performance impact is negligible.

Of the 25 flights flown with the pogostick actuated focused on striking perpendicular to the virtual surface normal, Table 5.5 provides summary results of accuracy at strike. Angular accuracy is determined by taking the orientation vector (roll, pitch, and yaw) of the pogostick and converting it into a orientation vector that points the same direction as the pogostick vector, but with yaw constrained to zero. This allows for errors to be calculated in the 2 DoF (roll and pitch) of the arm relative to the desired strike orientation. The average pitch error is $-0.711°$ and the average roll error is 2.988°, with standard deviations of 1.251° and 1.402°, respectively.

Five additional flights striking a virtual surface with a horizontal velocity of 0.25 m/s, were conducted with the pogostick targeting and angle of 15° in the pitch axis. Of these

Table 5.5: Pointing Accuracy at Strike

Flight Number	Pitch Error	Roll Error
1	-0.8818°	5.871°
2	-0.088°	2.473°
3	-0.998°	3.113°
4	-1.394°	1.913°
5	-1.801°	3.631°
6	0.107°	2.499°
7	-2.672°	3.458°
8	0.025°	2.693°
9	0.162°	1.717°
10	-0.771°	2.717°
11	2.909°	1.161°
12	0.036°	3.685°
13	-1.337°	1.920°
14	-3.442°	-0.807°
15	-1.356°	2.682°
16	-0.925°	2.710°
17	-1.246°	5.212°
18	-0.471°	3.259°
19	-1.616°	3.362°
20	0.527°	1.649°
21	-2.112°	4.145°
22	-1.160°	2.291°
23	0.246°	4.612°
24	0.260°	4.346°
25	0.288°	4.392°
Average Error	**-0.711°**	**2.988°**
Standard Deviation	**1.251°**	**1.402°**

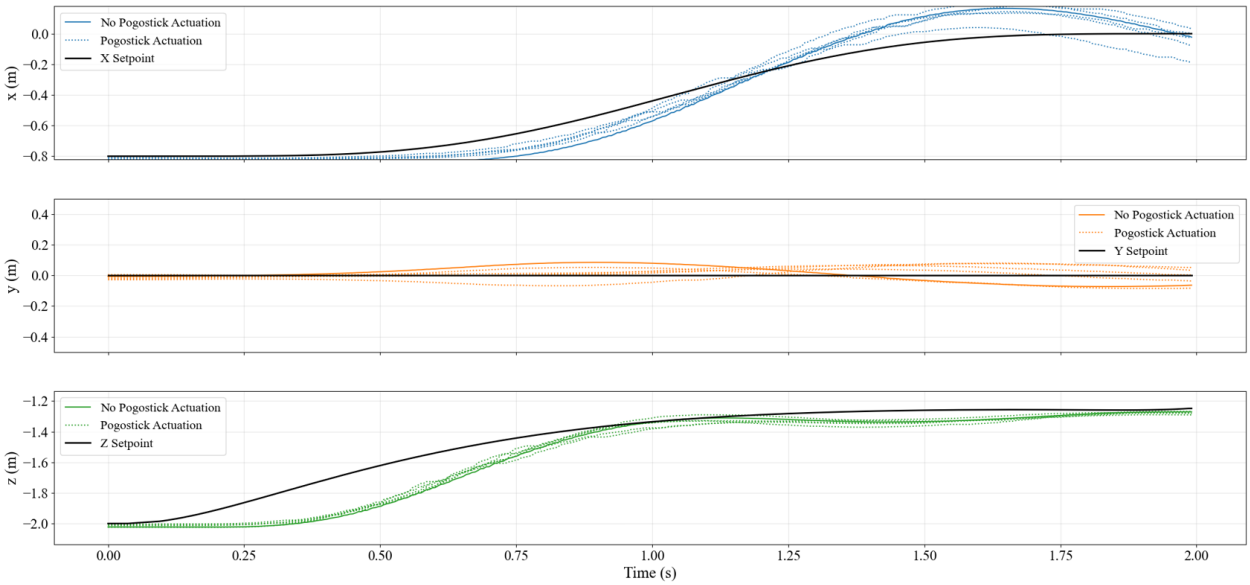

Figure 5.10: Sample results from 25 trajectory tracking flights with and without pogostick actuation.

five flights, the system had an average pitch error of $-1.223°$ and an average roll error of $2.569°$ with standard deviations of $2.023°$ and $1.549°$ respectively. These are relatively small changes in angular velocities, which resulted in the UAS flight control being able to easily stabilize a hover after each strike. The UAS struck the steel mesh each time, and then hovered directly over the strike location before issuing a landing command.

Experimental testing was also conducted with striking a physical steel mesh (to prevent ground effects). Of the 15 flights conducted striking the steel mesh, the system had an average pitch error of $-0.683°$ and an average roll error of $2.787°$ with standard deviations of $1.078°$ and $1.339°$ respectively. The goal of these strikes was to create collisions where the impulse of the collision was aligned with the pogostick of the UAS, limiting moments acting about the center of mass of the UAS. During these strikes, the average change in total angular velocity was $8.645°/s$ with a standard deviation of $4.298°/s$.

5.9 Discussion and Conclusion

We present a method for overcoming the limitations of an underactuated UAS for intentional collision-based aerial manipulation through the addition of an actuated 2-DoF pogostick. The pogostick receives a desired strike orientation, along with the pose of the parent UAS, and solves for the orientation corrections required for the pogostick to point in the desired strike direction. This enables more accurate intentional collisions, where collision responses

can be planned in advance to achieve a specified post-collision outcome.

Testing shows that motion of the pogostick has essentially no effect on either position tracking or motor commands during hover or during trajectory tracking. This indicates that the pogostick can continually correct for orientation errors without degrading UAS performance.

Experimental results also demonstrate that the actuated pogostick helps to overcome the limitations of the underactuated UAS by allowing the pogostick to make last-minute corrections to its orientation at strike, so the strike angle better matches the desired angle relative to the strike surface normal. The average pointing errors, $-0.711°$ in pitch and $2.988°$ in roll, are nonzero, but the associated standard deviations, $1.251°$ in pitch and $1.402°$ in roll, are relatively small. Similar errors were seen in horizontal strike testing with a virtual floor and test flights with actual collisions. The collisions themselves imparted little angular velocity changes, and the UAS was able to stabilize a hover after each strike. Both of these errors either meet (roll) or beat (pitch) the errors seen in the previous theoretical work in [47]. Further tuning of the cable tension and slackening gains could reduce this bias and improve pointing accuracy, as well as the addition of a cascaded arm orientation controller using motion capture feedback.

Future work on this payload would focus on improving pointing accuracy. Ideally, the system would incorporate encoders that accurately track the pogostick position, so that orientation corrections can be applied with higher precision. Such improvements would yield a payload with very little impact on the flight performance of the parent vehicle and even better strike performance (sub-degree accuracy).

CHAPTER 6

Conclusion

This dissertation lays the foundation for how intentional collisions can be leveraged as a tool for aerial manipulation, even when working with an underactuated system like a typical hexacopter. The original motivation was to find a way to interact with unknown objects that minimized interaction time while in flight, which naturally led to using intentional collisions. Intentional collisions have the benefit of collapsing all aerial manipulation interactions and forces into a single impulse, which simplifies disturbance rejection. However, with typical UAS, overcoming underactuation is critical for planning and executing these collisions reliably. This dissertation investigates several control strategies and payload designs to help work around that underactuation.

The idea of intentional collisions is first tested with a flight mode called Velocity Matching (Chapter 2). In this mode, the vehicle is assumed to be launched with a given speed and uses an experimentally determined, minimum-capped motor thrust to align the attitude of the UAS's rigid pogostick with the vehicle's velocity vector, treating the UAS as approximately ballistic. By aligning with the velocity vector, Velocity Matching makes sure the pogostick is the first point of contact with the environment or object being manipulated. A companion flight envelope analysis maps initial displacements from the target location to either recoverable or unrecoverable states, defining what we call the basin of recovery for the system. The main limitations of this approach come from the open-loop position control and the ballistic flight assumption, which turned out not to hold in practice. This reduced the usefulness of the simulated flight envelope as a predictor of the true flight envelope. Future work should start by updating the simulation environment to match the setups used in later chapters, so that predictions of the UAS flight envelope are more accurate. After that, higher-resolution testing of the envelope should be done, along with better tuning of the maximum thrust so that the ballistic assumption is closer to reality.

Chapter 3 introduces a Switching Control strategy and simulation framework to help overcome the underactuation of the UAS. The controller switches from cascaded position–attitude control to cascaded attitude-only control, using dynamically feasible trajec-

tories that include both attitude and position. This allows the UAS to perform last-minute orientation corrections before impact. The simulation tool is used to predict good switching times for a given pre-bounce initial condition, and the experimental tests show that these predictions are reasonably reliable. This demonstrates that it is possible to work around the vehicle's underactuation by switching between different controllers at the right time. Going forward, the switching strategy needs more extensive validation across a wider range of initial conditions to fully trust the simulation's predictions prior to bouncing.

In Chapter 4, a theoretical torque-sourced actuator is added to change the pogostick orientation directly. The idea here is that the UAS can remain fully position-controlled at all times, while the pogostick is actuated to maintain alignment for the strike. This chapter looks at using a partially feedback-linearized controller so that the flight and arm control laws are coupled into a single framework. Simulation studies show that this method can work reliably, with strike angle errors of roughly 3 degrees or less. The natural next step for this work is to build a physical torque-sourced actuator with the same center of rotation as the UAS center of mass and test it with the feedback-linearized controller.

Chapter 5 focuses on testing a physical actuated pogostick. Due to constraints on locating the pogostick's rotation about the UAS center of mass, the theoretical design from Chapter 4 could not be implemented directly. Instead, a cable-driven mechanism was developed that moves the pogostick along a hemispherical surface, providing 2-DoF control. Testing showed that the UAS flight performance with the pogostick stationary versus moving was largely unchanged, and final pointing accuracies were significantly better than those in Chapter 4 (on average less than 1 degree of pitch error and less than 3 degrees of roll error). Further improvements could come from better initial alignment and more precise tensioning of the cables, but the real next step is to add encoders that can accurately measure the pogostick orientation and feed that information back to the controller.

Next steps toward making this system field-deployable should focus on the navigation and guidance components of aerial manipulation in environments without motion capture. Because the original intent was to develop a platform for rainforest sampling, several additional capabilities must be implemented to achieve reliable operation in the field. In dense rainforest vegetation, the system should be able to function in GPS-denied conditions.

Our proposed direction for future work is a shared autonomy approach that combines onboard autonomy with a human operator. Ideally, the system would support vision or LiDAR-based navigation for maneuvering through the rainforest. Once the vehicle reaches a target tree, the human operator could provide input and select the branch to be sampled through a user interface. This reduces the autonomy burden of branch selection, allowing the perception and navigation stack to focus on state estimation and maintaining robust

tracking of the chosen branch during sampling.

Overall, this dissertation shows that overcoming underactuation in traditional UAS is possible, either by introducing new control strategies or by offloading some of the alignment task to an actuated payload. Both approaches are effective in their own ways and together they help build the groundwork for using intentional collisions as a practical tool for aerial manipulation.

APPENDIX A

Cutting and Grasping Ideas

A.1 Non-Collision-Based Cutting Ideas

Initial concepts for cutting and grasping tree branches emphasized maintaining extended contact with the branch during manipulation. Early ideation focused on two cutting approaches, both intended to occur during a perching maneuver. The first approach used a cable saw (a common survival tool for cutting small branches), but this would require long interaction times and a complex mechanism to wrap and tension the cable around the branch from the UAS. The second approach considered a circular saw; however, it would still involve extended contact time, introduce cutting torques that must be rejected, and require additional battery power (reducing flight time and limiting overall payload capacity).

A.2 Collision-Based Grasping and Cutting Mechanism

Because our approach to aerial manipulation relies on intentional collisions, we developed a collision-triggered grasping and cutting mechanism. The concept described here has been designed but not yet experimentally validated, which is necessary future work.

The proposed mechanism, shown in Fig. A.1, is inspired by bear traps. Tt is spring-loaded and uses a trigger located between the cutting and grasping jaws. The trigger (red) holds both the grasping arms (yellow) and the cutting arms (blue) in the open configuration during flight and prior to interaction. Upon collision with a branch, the collision force F_b dislodges the trigger from its detents, causing both the cutting and grasping arms to close. This sequence is illustrated in Fig. A.2.

A key feature of this design is that the collision initiates the cutting action, while the collision response occurs only after the cut is completed (the pogostick bottoms out). This ensures that the branch is both cut and securely grasped before the vehicle separates. This ensures that not only is the branch cut and grasped before flying away, but the collision

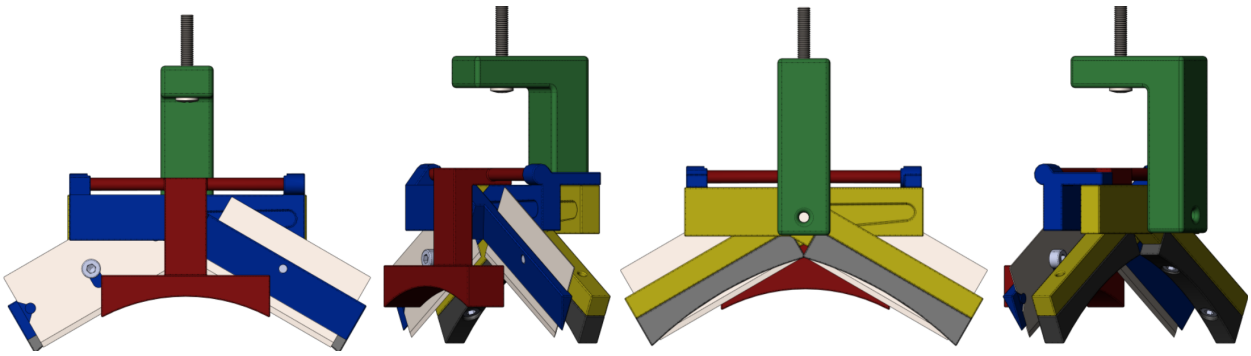

Figure A.1: Initial design of a bear-trap-inspired collision-based cutting and grasping mechanism. Images show all sides of the mechanism.

itself can help cut through the branch as the blades strike with the collision strike velocity and angular velocity from the closing mechanism.

The mechanism is entirely passively actuated using torsion springs on both the cutting arms (blue) and grasping arms (yellow). This passive design minimizes mass and eliminates onboard power requirements. The cutting arms are designed with standard razor blades, enabling low-cost maintenance and rapid blade replacement. One set of torsion spring slots is shown in Fig. A.3; the torsion spring can be adjusted to increase or decrease the cutting torque, as different tree species and branch sizes will require different cutting forces. Both the cutting and the grasping arms have their own set of torsion springs that can be independently tuned for any given grasp and cut task.

As shown in the middle panel of Fig. A.3, two forces F_1 and F_2 act between the cutting and grasping subsystems through a pair of rigid retaining arms. These retaining arms mechanically couple the two jaw sets such that when the cutting arms are open, the grasping arms are also held open. More importantly, the retaining arms constrain the grasping arms so that the grasping arm cannot close further than the cutting arms. This coupling provides several benefits. It prevents the mechanism from grasping objects that the cutters cannot penetrate. It allows both subsystems to be held open using a single trigger. It enables the grasping-arm torsion springs to assist closure without hindering cutting. If the grasping arms make contact first and begin to clamp the branch, the cutting arms can continue closing under their own torsion springs to complete the cut.

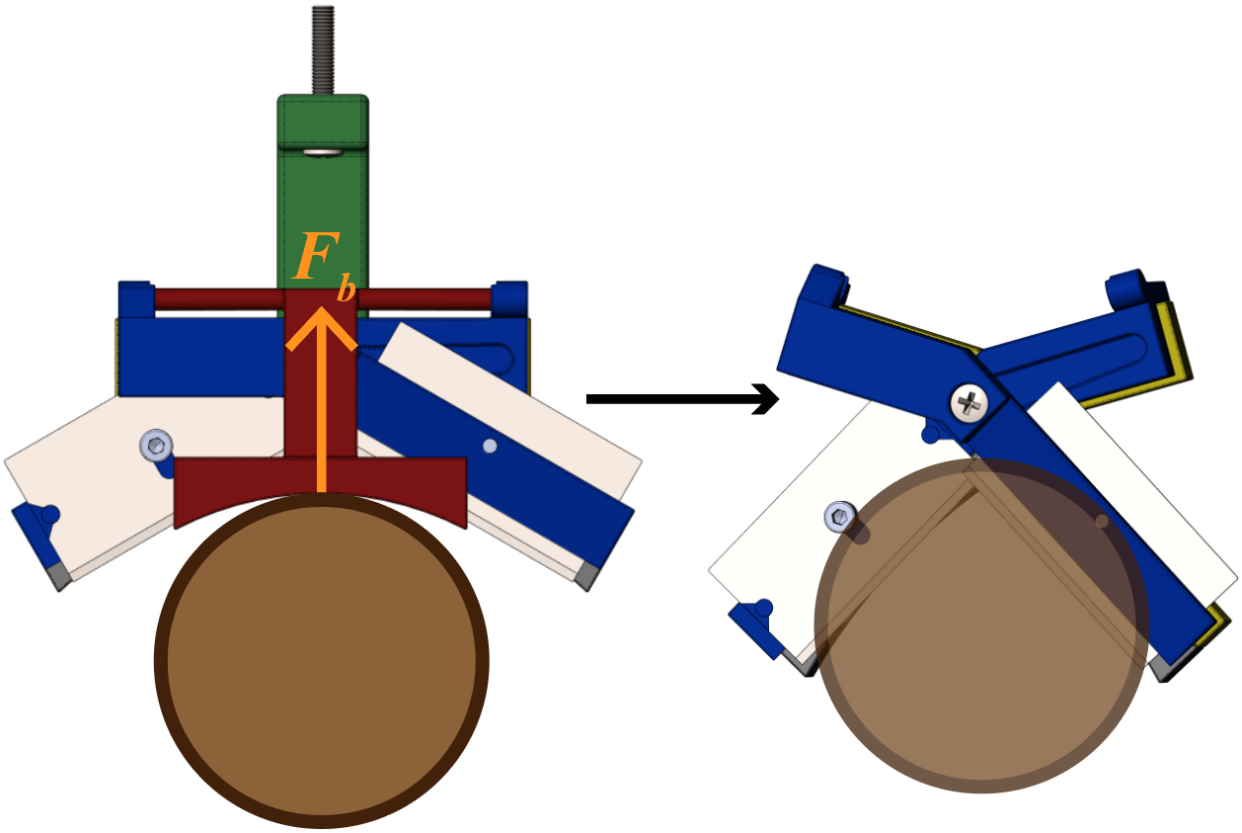

Figure A.2: Visualization of mechanism triggering and cutting.

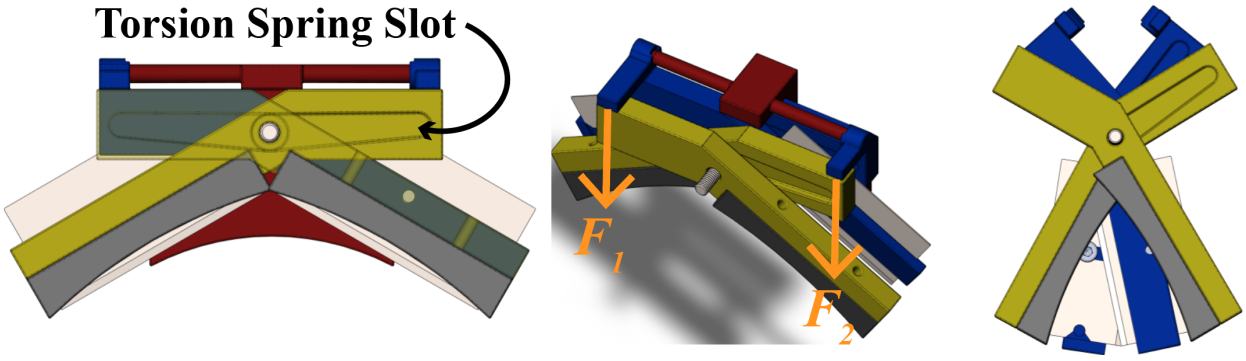

Figure A.3: Visualization of torsion springs and arm retention linkage.

BIBLIOGRAPHY

[1] Osman Acar, Hacı Sağlam, and Ziya Şaka. Measuring curvature of trajectory traced by coupler of an optimal four-link spherical mechanism. Measurement, 176:109189, 2021.

[2] Jeffrey Aguilar, Tingnan Zhang, Feifei Qian, Mark Kingsbury, Benjamin McInroe, Nicole Mazouchova, Chen Li, Ryan Maladen, Chaohui Gong, Matt Travers, et al. A review on locomotion robophysics: the study of movement at the intersection of robotics, soft matter and dynamical systems. Reports on Progress in Physics, 79(11):110001, 2016.

[3] Aytaç Altan and Rıfat Hacıoğlu. Model predictive control of three-axis gimbal system mounted on uav for real-time target tracking under external disturbances. Mechanical Systems and Signal Processing, 138:106548, 2020.

[4] Jon Arrizabalaga and Markus Ryll. Towards time-optimal tunnel-following for quadrotors. In 2022 International Conference on Robotics and Automation (ICRA), pages 4044–4050. IEEE, 2022.

[5] Valeria Artale, CLR Milazzo, and Angela Ricciardello. Mathematical modeling of hexacopter. Applied mathematical sciences, 7(97):4805–4811, 2013.

[6] Valeria Artale, Cristina L. R. Milazzo, Calogero Orlando, and Angela Ricciardello. Particle swarm optimization applied to hexarotor flight dynamics. In Giovanni Russo, Vincenzo Capasso, Giuseppe Nicosia, and Vittorio Romano, editors, Progress in Industrial Mathematics at ECMI 2014, pages 625–632, Cham, 2016. Springer International Publishing.

[7] Songnan Bai, Qiqi Pan, Runze Ding, Huaiyuan Jia, Zhengbao Yang, and Pakpong Chirarattananon. An agile monopedal hopping quadcopter with synergistic hybrid locomotion. Science Robotics, 9(89):eadi8912, 2024.

[8] Ivan Bao, José C Pacheco, Atharva Navsalkar, Andrew Scheffer, Sashreek Shankar, Andrew Zhao, Hongyu Zhou, and Vasileios Tzoumas. Morpheus: Morphable omnidirectional unmanned system. arXiv preprint arXiv:2505.18270, 2025.

[9] Shraddha Barawkar and Manish Kumar. An Energy Efficient Jumping Drone — A Simple Projectile Motion Approach. IFAC-PapersOnLine, 56(3):175–180, 2023.

[10] Martin G Barker and Michelle A Pinard. Forest canopy research: sampling problems, and some solutions. In Tropical forest canopies: ecology and management, pages 23–38. Springer, 2001.

[11] Evan Beachly, Carrick Detweiler, Sebastian Elbaum, Dirac Twidwell, and Brittany Duncan. Uas-rx interface for mission planning, fire tracking, fire ignition, and real-time updating. In 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), pages 67–74. IEEE, 2017.

[12] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. arXiv preprint arXiv:2503.14734, 2025.

[13] Robert Bogue. Growth in e-commerce boosts innovation in the warehouse robot market. Industrial Robot: An International Journal, 43(6):583–587, 2016.

[14] Peter Burggräf, Alejandro Ricardo Pérez Martínez, Hubert Roth, and Johannes Wagner. Quadrotors in factory applications: design and implementation of the quadrotor's p-pid cascade control system: modeling and implementation. SN Applied Sciences, 1(7):722, 2019.

[15] Samuel Burns and Matthew Woodward. Design and control of an energy accumulative hopping robot. arXiv preprint arXiv:2312.08301, 2023.

[16] Herbert Busemann. The geometry of geodesics. Courier Corporation, 2012.

[17] Abbas Chamseddine, Youmin Zhang, Camille Alain Rabbath, and Didier Theilliol. Trajectory planning and replanning strategies applied to a quadrotor unmanned aerial vehicle. Journal of Guidance, Control, and Dynamics, 35(5):1667–1671, 2012.

[18] Guillaume Charron, Thomas Robichaud-Courteau, Hughes La Vigne, Samantha Weintraub, Andy Hill, Douglas Justice, Nicolas Bélanger, and Alexis Lussier Desbiens. The deleaves: a uav device for efficient tree canopy sampling. Journal of Unmanned Vehicle Systems, 8(3):245–264, 2020.

[19] Nana Cheng and Chaoli Wang. A switching mode control scheme for the hovering control of quadrotor unmanned aerial vehicles. Mathematics, 11(4), 2023.

[20] Kourosh Darvish, Luigi Penco, Joao Ramos, Rafael Cisneros, Jerry Pratt, Eiichi Yoshida, Serena Ivaldi, and Daniele Pucci. Teleoperation of humanoid robots: A survey. IEEE Transactions on Robotics, 39(3):1706–1727, 2023.

[21] Carrick Detweiler, Sebastian Elbaum, James Higgins, Christian Laney, Craig Allen, Dirac L Twidwell Jr, and Evan Michale Beachly. Fire suppression and ignition with unmanned aerial vehicles. UNL Digial Commons, 2021.

[22] Bara J Emran and Homayoun Najjaran. A review of quadrotor: An underactuated mechanical system. Annual Reviews in Control, 46:165–180, 2018.

[23] Parker Ewen, Adam Li, Yuxin Chen, Steven Hong, and Ram Vasudevan. These maps are made for walking: Real-time terrain property estimation for mobile robots. IEEE Robotics and Automation Letters, 2022.

[24] Jorge Furtado Falorca, João PND Miraldes, and João Carlos Gonçalves Lanzinha. New trends in visual inspection of buildings and structures: Study for the use of drones. Open Engineering, 11(1):734–743, 2021.

[25] Ghulam Farid, Haris TAHIR Hamid, Shahid Karim, and Sohaib Tahir. Waypoint-based generation of guided and optimal trajectories for autonomous tracking using a quadrotor uav. Studies in Informatics and Control, 27(2):225–236, 2018.

[26] Nima Fazeli, Samuel Zapolsky, Evan Drumwright, and Alberto Rodriguez. Fundamental limitations in performance and interpretability of common planar rigid-body contact models. In Robotics Research, pages 555–571. Springer, 2020.

[27] Joshua Fishman, Samuel Ubellacker, Nathan Hughes, and Luca Carlone. Dynamic grasping with a "soft" drone: From theory to practice. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4214–4221. IEEE, 2021.

[28] Minyue Fu and B. Barmish. Adaptive stabilization of linear systems via switching control. IEEE Transactions on Automatic Control, 31(12):1097–1103, 1986.

[29] Alejandro Ernesto Gomez-Tamm, V Perez-Sanchez, Begoña C Arrue, and Aníbal Ollero. Sma actuated low-weight bio-inspired claws for grasping and perching using flapping wing aerial systems. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 8807–8814. IEEE, 2020.

[30] Yukai Gong, Ross Hartley, Xingye Da, Ayonga Hereid, Omar Harib, Jiunn-Kai Huang, and Jessy Grizzle. Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway. In 2019 American Control Conference (ACC), pages 4559–4566. IEEE, 2019.

[31] Drew Hanover, Philipp Foehn, Sihao Sun, Elia Kaufmann, and Davide Scaramuzza. Performance, precision, and payloads: Adaptive nonlinear MPC for quadrotors. IEEE Robotics and Automation Letters, 7(2):690–697, 2021.

[32] Mark A Higgins and Kalle Ruokolainen. Rapid tropical forest inventory: a comparison of techniques based on inventory data from western amazonia. Conservation Biology, 18(3):799–811, 2004.

[33] Sheng-Wei Huang, Edward Chen, and Jenhwa Guo. Efficient seafloor classification and submarine cable route design using an autonomous underwater vehicle. IEEE Journal of Oceanic Engineering, 43(1):7–18, 2017.

[34] Hossein Bonyan Khamseh, Farrokh Janabi-Sharifi, and Abdelkader Abdessameud. Aerial manipulation—a literature survey. Robotics and Autonomous Systems, 107:221–235, 2018.

[35] Ryan Kitchen, Nick Bierwolf, Sean Harbertson, Brage Platt, Dean Owen, Klaus Griessmann, and Mark A Minor. Design and evaluation of a perching hexacopter drone for energy harvesting from power lines. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1192–1198. IEEE, 2020.

[36] Sean Krisanski, Mohammad Sadegh Taskhiri, James Montgomery, and Paul Turner. Design and testing of a novel unoccupied aircraft system for the collection of forest canopy samples. Forests, 13(2):153, 2022.

[37] Robert Ladig, Hannibal Paul, Ryo Miyazaki, and Kazuhiro Shimonomura. Aerial manipulation using multirotor uav: a review from the aspect of operating space and force. Journal of Robotics and Mechatronics, 33(2):196–204, 2021.

[38] Dongjae Lee, Hoseong Seo, Dabin Kim, and H Jin Kim. Aerial manipulation using model predictive control for opening a hinged door. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 1237–1242. IEEE, 2020.

[39] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor UAV on SE (3). In 49th IEEE conference on decision and control (CDC), pages 5420–5425. IEEE, 2010.

[40] Daniel R McArthur, Arindam B Chowdhury, and David J Cappelleri. Autonomous control of the interacting-boomcopter uav for remote sensor mounting. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 5219–5224. IEEE, 2018.

[41] N Harris McClamroch. Steady aircraft flight and performance. Princeton University Press, 2011.

[42] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In 2015 IEEE international conference on robotics and automation (ICRA), pages 6235–6240. IEEE, 2015.

[43] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In 2011 IEEE international conference on robotics and automation, pages 2520–2525. IEEE, 2011.

[44] Hengameh Mirhajianmoghadam, Nicholas Grijalva, and Luis Rodolfo Garcia Carrillo. Fast trajectory tracking for commercial off-the-shelf uas using model predictive control: Real-time implementation and performance analysis. In 2024 International Conference on Unmanned Aircraft Systems (ICUAS), pages 658–665. IEEE, 2024.

[45] Balmukund Mishra, Deepak Garg, Pratik Narang, and Vipul Mishra. Drone-surveillance for search and rescue in natural disaster. Computer Communications, 156:1–10, 2020.

[46] Glen Mullineux. Atlas of spherical four-bar mechanisms. Mechanism and machine theory, 46(11):1811–1823, 2011.

[47] Mark Nail, Ella Atkins, and R Brent Gillespie. Shifting underactuated configuration variables in aerial manipulation by adding an actuated arm. In 2025 International Conference on Unmanned Aircraft Systems (ICUAS), pages 316–322. IEEE, 2025.

[48] Mark Nail, Nick Jänne, Olivia Ma, Gabriel Arellano, Ella Atkins, and R Brent Gillespie. Simplifying aerial manipulation using intentional collisions. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 5359–5365. IEEE, 2023.

[49] Sylvio P Netto, Allan L Pelissari, Vinicius C Cysneiros, Marcelo Bonazza, and Carlos R Sanquetta. Sampling procedures for inventory of commercial volume tree species in amazon forest. Anais da Academia Brasileira de Ciências, 89:1829–1840, 2017.

[50] Huan Nguyen and Kostas Alexis. Forceful aerial manipulation based on an aerial robotic chain: Hybrid modeling and control. IEEE Robotics and Automation Letters, 6(2):3711–3719, 2021.

[51] Ngoc Phi Nguyen, Nguyen Xuan Mung, and Sung Kyung Hong. Actuator fault detection and fault-tolerant control for hexacopter. Sensors, 19(21):4721, 2019.

[52] Janet Mayowa Nwaogu, Yang Yang, Albert PC Chan, and Hung-lin Chi. Application of drones in the architecture, engineering, and construction (aec) industry. Automation in Construction, 150:104827, 2023.

[53] Reza Olfati-Saber. Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles. PhD thesis, Massachusetts Institute of Technology, 2001.

[54] Budi Padmaja, CH VKNSN Moorthy, N Venkateswarulu, and Myneni Madhu Bala. Exploration of issues, challenges and latest developments in autonomous cars. Journal of Big Data, 10(1):61, 2023.

[55] Julio L Paneque, Jose Ramiro Martínez-de Dios, Anibal Ollero, Drew Hanover, Sihao Sun, Angel Romero, and Davide Scaramuzza. Perception-aware perching on powerlines with multirotors. IEEE Robotics and Automation Letters, 7(2):3077–3084, 2022.

[56] Adam Plowcha, Justin Bradley, Jacob Hogberg, Thomas Ammon, Mark Nail, Brittany Duncan, and Carrick Detweiler. Autonomous, long-range, sensor emplacement using unmanned aircraft systems. Field Robotics, 2:437–467, 2022.

[57] Quan Quan. Introduction to multicopter design and control. Springer, 2017.

[58] Marc H Raibert, H Benjamin Brown Jr, Michael Chepponis, Jeff Koechling, and Jessica K Hodgins. Dynamically stable legged locomotion. Technical report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab, 1989.

[59] Tarek Rakha and Alice Gorodetsky. Review of unmanned aerial system (UAS) applications in the built environment: Towards automated building inspection procedures using drones. Automation in construction, 93:252–264, 2018.

[60] P Ramon-Soria, Alejandro Ernesto Gomez-Tamm, Francisco Javier Garcia-Rubiales, Begoña C Arrue, and Aníbal Ollero. Autonomous landing on pipes using soft gripper for inspection and maintenance in outdoor environments. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5832–5839. IEEE, 2019.

[61] Jenna Reher and Aaron D Ames. Dynamic walking: Toward agile and efficient bipedal robots. Annual Review of Control, Robotics, and Autonomous Systems, 4(1):535–572, 2021.

[62] C David Remy. Ambiguous collision outcomes and sliding with infinite friction in models of legged systems. The International Journal of Robotics Research, 36(12):1252–1267, 2017.

[63] L Rey and R Clavel. The delta parallel robot. In Parallel Kinematic Machines: Theoretical Aspects and Industrial Requirements, pages 401–417. Springer, 1999.

[64] Matthew M Romano, Yuxin Chen, Prince Kuevor, Owen Marshall, and Ella Atkins. Nailed it: Autonomous roofing with a nailgun-equipped octocopter. In AIAA AVIATION 2021 FORUM, page 3211, 2021.

[65] Fabio Ruggiero, Vincenzo Lippiello, and Anibal Ollero. Aerial manipulation: A literature review. IEEE Robotics and Automation Letters, 3(3):1957–1964, 2018.

[66] Jean-Jacques E Slotine, Weiping Li, et al. Applied nonlinear control, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

[67] KJ Smelt, JJ de Jong, CM de Blok, and DM Brouwer. Spherical mechanism, with adjustable center of rotation. Mikroniek, 2020(04), 2020.

[68] Enrica Soria, Fabrizio Schiano, and Dario Floreano. Distributed predictive drone swarms in cluttered environments. IEEE Robotics and Automation Letters, 7(1):73–80, 2021.

[69] David E Stewart. Rigid-body dynamics with friction and impact. SIAM review, 42(1):3–39, 2000.

[70] Yue Sun, Adam Plowcha, Mark Nail, Sebastian Elbaum, Benjamin Terry, and Carrick Detweiler. Unmanned aerial auger for underground sensor installation. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1374–1381. IEEE, 2018.

[71] Ezra Tal and Sertac Karaman. Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. IEEE Transactions on Control Systems Technology, 29(3):1203–1218, 2020.

[72] Kazuo Tanaka, Masaaki Iwasaki, and Hua O Wang. Switching control of an r/c hovercraft: stabilization and smooth switching. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 31(6):853–863, 2001.

[73] Justin Thomas, Giuseppe Loianno, Morgan Pope, Elliot W Hawkes, Matthew A Estrada, Hao Jiang, Mark R Cutkosky, and Vijay Kumar. Planning and control of aggressive maneuvers for perching on inclined and vertical surfaces. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, volume 57144, page V05CT08A012. American Society of Mechanical Engineers, 2015.

[74] Justin Thomas, Joe Polin, Koushil Sreenath, and Vijay Kumar. Avian-inspired grasping for quadrotor micro uavs. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, volume 55935, page V06AT07A014. American Society of Mechanical Engineers, 2013.

[75] Petter Tøndel, Tor Arne Johansen, and Alberto Bemporad. An algorithm for multi-parametric quadratic programming and explicit mpc solutions. Automatica, 39(3):489–497, 2003.

[76] Yi Wang, Jiarong Kang, Zhiheng Chen, and Xiaobin Xiong. Terrestrial locomotion of PogoX: From hardware design to energy shaping and step-to-step dynamics based control. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 3419–3425. IEEE, 2024.

[77] Adam C Watts, Vincent G Ambrosia, and Everett A Hinkley. Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use. Remote Sensing, 4(6):1671–1692, 2012.

[78] Jingjun Yu, Kang Wu, Guanghua Zong, and Xianwen Kong. A comparative study on motion characteristics of three two-degree-of-freedom pointing mechanisms. Journal of mechanisms and robotics, 8(2):021027, 2016.

[79] Brian Zhu, Jiawei Xu, Andrew Charway, and David Saldaña. Pogodrone: Design, model, and control of a jumping quadrotor. In 2022 International Conference on Robotics and Automation (ICRA), pages 2031–2037. IEEE, 2022.

[80] Enrique Zuazua. Switching control. Journal of the European Mathematical Society, 13(1):85–117, 2010.