# Planning, Control, and Estimation for Diverse Multi-UAS Missions

by

Matthew Romano

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in the University of Michigan
2022

Doctoral Committee:

Professor Ella M. Atkins, Chair
Assistant Professor Nima Fazeli
Associate Professor Dimitra Panagou
Assistant Professor Hossein Rastgoftar, University of Arizona

Matthew Romano

mmroma@umich.edu

ORCID iD: 0000-0001-5542-1273

# ACKNOWLEDGMENTS

I would love to thank my advisor Prof. Ella M Atkins. You have been a phenomenal advisor. You've supported me, pushed me, and made me the researcher and engineer that I am today. You have dreamt up the most daring applications of UAS technology (re: nailgun drone) and I have absolutely loved seeing them to fruition with your guidance. I loved having the luxury of enjoying Domino's pizza during lab meetings because of your sponsorship for the past five years. You truly made me and I'm sure everyone else feel like they are part of the A2Sys family. You are more than just a research advisor because I know that I could come to you for anything and you would be more than willing to help. Thank you to my committee members Prof. Panagou, Prof. Rastgoftar, and Prof. Fazeli for their valuable feedback. I appreciate each of your perspectives and guidance.

Thank you to the A2Sys lab members: Mia Stevens, Brian Yao, Cosme Ochoa, Jeremy Castagno, Prashin Sharma, Akshay Mathur, Joseph Kim, Mark Nail, Paul Flanigen, Brandon Apodaca, and Jianyang (Joe) Tang. Special thanks to Prince Kuevor who is absolutely amazing. We came into the program together and you were with me every step of the way, helping, pushing, and supporting me. Thank you to Michael Gonzalez for putting in so much time and effort for our game nights and random escapades. Thank you to my hackathon buddies Jeremy, Prince, and Prashin! I had an absolute blast solving complex problems with little sleep and caffeine raging through my bloodstream. I've had the pleasure of working with several master's, undergraduate, and high school students during the course of my PhD. Thank you to Derek Lukacs, Owen Marshall, Catherine Ferrey, Jack Getty, Yuxin Chen, N.M.T. Vijay, Vansh Amin, Anne Ye, and John Pye. Thank you Dr. Peter Gaskell. I will never forget the days going to your office in the basement of the SRB and the conversations to get me started with XBees and the kill switch. You provided invaluable help and are a genuinely kind person willing to give your time even when your workload is high.

I would love to thank my family. Dad, Joey, and Mia, even though we were a four hour drive apart, I always felt your presence and support to keep me focused on finishing. Finally, I want to thank my fiance, Pha. While I am grateful that you pushed my payload, I am even more grateful that you pushed and supported me to do my best. You dealt with me when I didn't feel like writing and were woken up for the hundreds of night time flight testing. But the journey we are on together is incredible and I am the luckiest person in the world to share it with you. I love you. Thank you.

# TABLE OF CONTENTS

# LIST OF FIGURES

FIGURE

# LIST OF TABLES

TABLE

# LIST OF ALGORITHMS

# LIST OF APPENDICES

# LIST OF ABBREVIATIONS

**AASLP** Altitude Aware Straight Line Planer
**ADC** Analog-To-Digital Converter
**AFRL** Air Force Research Lab
**AGL** Above Ground Level
**AI** Artificial Intelligence
**AMASE** Aerospace Multi-Agent Simulation Environment
**APM** ArduPilot
**BBb** BeagleBone Blue
**BVLOS** Beyond-Visual-Line-Of-Sight
**CAD** Computer-Aided Design
**CEM** Containment Exclusion Mode
**CCW** Counter-Clockwise
**CPP** Coverage Path Planning
**CW** Clockwise
**DC** Direct Current
**DOF** Degrees of Freedom
**DTED** Digital Terrain Elevation Data
**ESC** Electronic Speed Controller
**FPV** First Person View
**GCS** Ground Control Station
**GND** Ground
**GPIO** General-Purpose Input/Output
**GPS** Global Positioning System
**GPU** Graphics Processing Unit
**GUI** Graphical User Interface
**IMU** Inertial Measurement Unit
**IOU** Intersection over Union
**I2C** Inter-Integrated Circuit
**KS** Kill Switch
**LiPo** Lithium Polymer
**MCU** Microcontroller
**MILP** Mixed Integer Linear Programming

**MSL** Mean Sea Level
**MQS** Multi-Quadrotor System
**NED** North-East-Down
**OMPL** Open Motion Planning Library
**PCB** Printed Circuit Board
**PID** Proportional-Integral-Derivative
**PSD** Power Spectral Density
**PSO** Particle Swarm Optimization
**PWM** Pulse Width Modulation
**RRT** Rapidaly-Exploring Random Tree
**RTK** Real-Time Kinematic Positioning
**SVM** Support Vector Machine
**UART** Universal Asynchronous Receiver Transmitter
**UAS** Unmanned Aircraft System
**USGS** United States Geological Survey
**UTM** UAS Traffic Management
**W-LOG** Weighted Logistic Regression
**W-LS** Weighted Least Squares Regression
**W-SVM** Weighted Support Vector Machine

# LIST OF SYMBOLS

| | |
|---|---|
| $a$ | Rotor arm length |
| $a_f$ | Virtual cylindrical obstacle radius |
| $\boldsymbol{a}_b^A$ | Acceleration of body b in frame A |
| $b_i$ | Damping coefficient for tether $i$ |
| $c_{ac}$ | Admittance Controller Virtual Damping Coefficient |
| $c_1, c_2$ | Motor constants |
| $d$ | Payload Diameter |
| $d_{des}$ | Line follower desired offset from local boundary |
| $d_i$ | Sensor $i$ offset term |
| $d_{line}$ | Actual offset from local boundary |
| $d_{min}$ | Minimum separation distance between agents |
| $\mathbf{d}$ | Displacement vector of deformable formation |
| $\mathbf{d}_g$ | Vector used in Chapter 3 derivation |
| $\mathbf{d}_l$ | Vector used in Chapter 3 derivation |
| $f$ | A scalar force |
| $f_c$ | Cutoff frequency |
| $f_s$ | Sampling frequency |
| $g$ | Acceleration due to gravity on Earth |
| $h_{max}$ | Maximum AGL for terrain avoidance |
| $h_{min}$ | Minimum AGL for terrain avoidance |
| $k_i$ | Spring constant for tether $i$ |

| | |
|---|---|
| $l$ | Side length of leading triangle |
| $l_i$ | Tether $i$ length |
| $m$ | Number of time steps to run CEM navigation function for |
| $m_{ac}$ | Admittance Controller Virtual Mass |
| $m_P$ | Payload Mass |
| $m_V$ | Vehicle Mass |
| $n$ | Number of dimensions |
| $o_j$ | Sensor measurement $j$ |
| $p_i, q_i, r_i$ | Attitude rates (about ($x$, $y$, and $z$ respectively) for body $i$) |
| $\mathbf{r}_i$ | Vehicle $i$ position |
| $\mathbf{r}_{d,i}$ | Local desired position of agent $i$ |
| $\mathbf{r}_{i,HT}$ | Global desired formation position of agent $i$ |
| $\mathbf{r}_s$ | AASLP start position |
| $\mathbf{r}_g$ | AASLP goal position |
| $\mathbf{r}_{i,s}$ | Starting position for vehicle $i$ |
| $\mathbf{r}_{i,g}$ | Goal position for vehicle $i$ |
| $\mathbf{r}_P$ | Payload position |
| $\mathbf{r}_{P,i}$ | Mounting point for tether $i$ |
| $s_i$ | Sensor $i$ scale term |
| $\mathbf{s}_i^P$ | Planned Formation Position of Vehicle $i$ in the Payload Frame |
| $\mathbf{t}_{P,i}^G$ | Tether vector $i$ connected between quadrotor $i$ and the payload |
| $t_x$ | $x$ minutes into the simulation |
| $u_i, v_i, w_i$ | Body frame velocities ($x$, $y$, and $z$ respectively) for body $i$ |
| $v$ | UAS scalar speed |
| $w_{i,j}$ | Communication weight from $j$ to $i$ |
| $C$ | Regularization parameter |
| $C_g$ | Constant to encapsulate gradient of $\phi$ |

| | |
|---|---|
| $D_s$ | Minimum separation distance between any pair of agents at $t = t_0$ |
| $D_b$ | Minimum distance from any follower to the leading triangle |
| $D_l$ | Distance between two parallel sides of the leading and bounding triangle |
| $\mathbf{D}_{loc}$ | Used for derivation in Chapter 3 |
| $\mathbf{D}_g$ | Used for derivation in Chapter 3 |
| $\mathcal{E}$ | Set of edges in graph |
| $\mathbf{F}_b^A$ | Force b in frame A |
| $FHS(t)$ | Fire Hack Score at time t |
| $G$ | Ground Coordinate Frame |
| $\mathcal{G}$ | Weighted directed graph that defines inter-agent communication |
| $H$ | Low Pass Filter transfer function |
| $J$ | Jacobian of potential and streamline functions |
| $J_{x,i}, J_{y,i}, J_{z,i}$ | Moment of inertia about $x$, $y$, and $z$ for body $i$ |
| $K$ | Gain Matrix |
| $\mathbf{K}$ | Used for derivation in Chapter 3 |
| $L_i$ | Tether $i$ unstretched length |
| $N_V$ | Number of Vehicles |
| $N_f$ | Number of Failed Vehicles |
| $N_F$ | Number of fires |
| $N_h$ | Number of Healthy Vehicles |
| $N_O$ | Number of obstacles |
| $N_K$ | Number of times to run CEM navigation function per time step |
| $N_i$ | Number of iterations in CalcXY |
| $P$ | Payload Coordinate Frame |
| $P_{sensor}$ | Power Spectral Density Height for Sensor Noise |
| $P_{dist}$ | Power Spectral Density Height for Disturbance Forces |
| $\mathbf{P}_0$ | Matrix of leader starting positions |

| | |
|---|---|
| $\mathbf{P}_d$ | Matrix of leader desired positions |
| $\mathbf{P}$ | Matrix of agent positions |
| $\mathbf{P}_{A,HT}$ | Matrix of agent global desired positions |
| $\mathbf{P}_{L,HT}$ | Matrix used for derivation in Chapter 3 |
| $\mathbf{Q}$ | Deformation matrix of deformable formation |
| $R_B^A$ | Rotation matrix from frame B to frame A |
| $S_i$ | Scenario score, a weighted sum of FHS(t) |
| $\mathcal{V}$ | Set of agents |
| $\mathcal{V}_L$ | Set of leader agents |
| $\mathcal{V}_F$ | Set of follower agents |
| $\mathbf{W}$ | Used for derivation in Chapter 3 |
| $\alpha$ | Low pass filter parameter |
| $\alpha_i$ | Elevation angle to describe the formation position of vehicle $i$ relative to the payload |
| $\alpha_{i,j}$ | Global communication weight from $j$ to $i$ |
| $\boldsymbol{\alpha}$ | Used for derivation in Chapter 3 |
| $\beta_i$ | Used for derivation in Chapter 3 |
| $\gamma_i$ | Azimuthal angle to describe the formation position of vehicle $i$ relative to the payload |
| $\delta_{j,i}$ | PWM signal to spin rotor $j$ on vehicle $i$ |
| $\delta_g$ | Global deviation bound |
| $\delta_l$ | Local deviation bound |
| $\delta_{max}$ | Largest of inter-agent and follower-boundary distance in starting formation |
| $\Delta x$ | AASLP Forward discretization distance |
| $\Delta z^+$ | Discretized climb rate limit |
| $\Delta z^-$ | Discretized descent rate limit |
| $\Delta t$ | AASLP time discretization |
| $\epsilon$ | Radius of enclosing ball around vehicles |

| | |
|---|---|
| $\eta_i$ | Used in Chapter 3 derivation |
| $\theta_r$ | Rotation angle of deformable formation (Chapter 6) |
| $\theta_d$ | Deformation angle of deformable formation (Chapter 6) |
| $\theta_i$ | Pitch angle for body $i$ |
| $\lambda_1$ | First eigenvalue of deformable formation |
| $\lambda_2$ | Second eigenvalue of deformable formation |
| $\lambda_{min}$ | Minimum safe eigenvalue of deformable formation |
| $\mu$ | Weighting factor to shift planned load distribution between center vehicle and others |
| $\sigma$ | Standard deviation |
| $\boldsymbol{\tau}_b^A$ | Torque b in frame A |
| $\phi_i$ | Roll angle for body $i$ |
| $\phi$ | Potential Function |
| $\psi_i$ | yaw angle for body $i$ |
| $\psi$ | Stream Function |

## ABSTRACT

Unmanned Aircraft Systems (UAS) are being used for a variety of single vehicle missions such as surveillance, inspection, and payload delivery. Teams of UAS can perform these same missions more efficiently and can pursue novel cooperative missions not possible with a single vehicle. However, this comes at the cost of increased system complexity and introduces the challenge of safe team coordination. This motivates our research to pursue four diverse multi-UAS mission configurations. We propose novel methods to command and control teams of UAS with the majority supported with full-scale experimental validation.

To support all experiments conducted in this thesis, an experimental test bed consisting of a custom, open-source quadrotor, flight controller, and supporting infrastructure is developed with designs and code shared publicly.

This thesis offers four specific contributions to enable the deployment of UAS teams in missions with potential to benefit society. First, an existing formation control method, continuum deformation, is experimentally validated with observed tracking errors and delays informing the theory. Required inter-vehicle separation constraints are defined and applied to the real system. A global minimum separation bound based on a local controller error bound is derived to guarantee safety during real-world flights.

Second, a novel heads-up haptic pushing interface is developed that enables a user to move a heavy payload carried by multiple small UAS through a crowded cluttered environment. Real-time estimation of user applied force via an instrumented payload updates virtual dynamics of an admittance controller to guide the system. This capability will support resilient and safe package delivery to untrained consumers and can assist in delivering medical and survival supplies in disaster relief scenarios.

Third, computationally efficient planning methods are developed to support wildfire mapping over a large area by a team of UAS. A state machine is used to handle multi-vehicle task allocation between exploration (coverage) and exploitation (line following). Efficiency gains are achieved by separating the 3D problem into 2D lateral and 1D terrain avoidance sub-problems. This work offers a first step towards mapping increasingly severe wildfire threats that cause significant damage and claim the lives of hundreds of people every year.

Fourth, a generalized path planner is developed to manage a deformable small UAS formation

capable of rotating, expanding, contracting, and shearing. Provably sufficient inter-agent collision avoidance constraints are leveraged to efficiently plan safe trajectories in large-scale complex but static environments. An integrated guidance and control module onboard each small UAS tracks the designed trajectory while avoiding pop-up obstacles and vehicle failures by following an ideal fluid flow field airspace template.

We are only beginning to imagine and prototype the missions made possible with teams of UAS. This thesis provides problem formulations, solution methods, and experimental realizations for four multi-UAS mission configurations. Countless other missions can be pursued that will operate safely and can improve quality of life. Continued research is needed to achieve this including full system integration with onboard sensing and collaboration with key stakeholders for each mission that deeply understand the problem spaces. Nonetheless, this thesis provides a solid foundation for future researchers to build upon.

# CHAPTER 1

# Introduction

## 1.1 Motivation

Unmanned Aircraft Systems (UAS) are becoming cheaper and more widely available. Single vehicle missions such as surveillance [3], inspection [4], and payload delivery [5] are being performed. Teams of multiple UAS might execute these same missions more efficiently and can pursue novel cooperative missions (Figures 1.1, 1.2, and 1.3). However, a multi-vehicle solution increases system complexity and introduces the challenge of safely coordinating the vehicle team.

In UAS Traffic Management (UTM), the goal is to safely fly many vehicles in a shared airspace. This is an active area of research and a wide array of solutions are being considered [1]. Formation flying is one method that can be used to group vehicles together to more easily manage traffic. Inter-agent collision avoidance, obstacle avoidance, and scalable communication are all challenges within formation control. Rigid formation control of UAS has been developed theoretically [6, 7] and validated experimentally [8]. Continuum deformation [9] theory provides versatility and supports deformable formations among other benefits described in Chapter 3. This dissertation is first to experimentally validate continuum deformation. Path planning for deformable formations is necessary to fully realize these benefits in the context of UTM. Plans developed offline with environment and airspace maps can prescribe formation motions and deformation plans through complex obstacle fields over extended route distances. In real-time, single-vehicle failures and pop-up obstacles need to be handled in a way that guarantees the entire formation will avoid obstacles by reactively deforming around them. This thesis investigates offline and reactive planning strategies needed to plan deformable formation flights and to modify them to avoid pop-up obstacles and failed vehicles.

Multiple small UAS can carry heavier payloads than can a single small vehicle. Counteracting tether forces provide stability to reduce or eliminate disturbances such as pendulum swinging and spinning about a single tether. Existing techniques consider either pre-planned guidance or remotely piloting the system [10, 11]. However, in dynamic, cluttered, and unknown environments representative of scenarios such as disaster relief, workers might be untrained in UAS flight, and


Figure 1.1: Multi-UAS mission #1: Cooperative control for UAS traffic management[1]

existing maps will likely be outdated. For package delivery an untrained consumer could be ready to receive their package but neither qualified nor authorized to take control of a UAS or UAS team. This dissertation presents a novel and intuitive haptic guidance interface that enables an untrained user to safely guide a slung payload through a cluttered environment. The user applies force in the direction they want the payload to move with a heads-up interface that allows the user to maintain continuous situational awareness. The close proximity and physical touch allows the user to have greater precision as compared with alternative guidance schemes. Ref. [12] conceived of this idea and laid a theoretical foundation. Simulation results however rely on simplifying assumptions for user interaction, sensors, and state estimation. This dissertation realizes and evaluates this haptically-guided small UAS payload transport system with consideration of all the necessary user interface, sensing, and state estimation details.

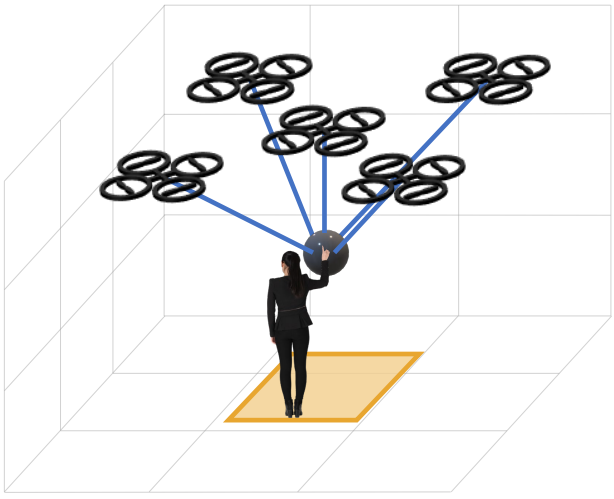


Figure 1.2: Multi-UAS mission #2: Payload transportation with direct haptic guidance

Cooperative UAS teams can support wide area surveillance and mapping when deployed

individually as well as in close formations. For wildfire mapping, UAS teams have the potential to support emergency response personnel by providing accurate real-time map updates of moving and growing wildfire regions. Teams of UAS equipped with fire sensors have the capability to gather local observations and acquire real-time maps of fire boundaries. Each vehicle must gather pertinent data while avoiding potentially complex terrain, remaining within sensor range to terrain, and rapidly exploring large spaces efficiently. Given computational resource constraints, 3D multi-vehicle path planning can be effective for offline planning but too slow for dynamic, real-time replanning [13]. This dissertation presents a computationally efficient multi-UAS planning strategy to cover a large area and follow/map wildfire boundaries once fire(s) are detected.

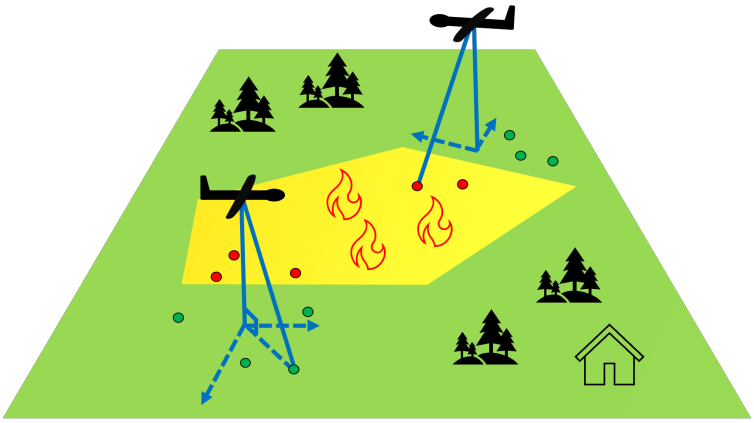


Figure 1.3: Multi-UAS mission #3: Wildfire detection and boundary mapping

The three missions considered in this dissertation are diverse even beyond the specific posed applications. The environments considered vary from obstacle-free to cluttered with rough terrain to being unknown and dynamically-varying. Communication is assumed perfect in wildfire mapping, but with limited network topology in formation control. Centralized command and control is modeled for wildifre mapping and payload transportation, but distributed networking is explored for formation control in UTM. State estimation is assumed accurate for most missions, but user force must be estimated with low-cost sensors in haptic guidance payload transportation. Due to the diversity within these multi-UAS missions, it is unlikely that a single method will be universally effective. Therefore, for these multi-UAS missions (and in general), it is very important to carefully develop solutions for each that properly respect mission constraints and truly satisfy the requirements.

## 1.2 Problem Statement

This dissertation aims to address the following challenges:

1. Continuum deformation offers deformable and safe formation control developed only in

theory and simulation prior to this dissertation. This work seeks to experimentally validate continuum deformation theory with a multiple quadcopter system. We define assumptions and address practical related challenges in real-world communication and control.

2. Multi-UAS payload transportation allows for heavy loads to be carried by many relatively smaller vehicles. In dynamic, cluttered, and unknown environments pre-planned or human piloted guidance is ineffective. A novel, intuitive, heads-up haptic guidance scheme has been proposed in theory but has relied on simplifying assumptions for user interaction, sensors, and state estimation. This dissertation realizes the haptic guidance scheme and experimentally validates its use to direct a team of UAS carrying a common payload.
3. UAS teams offer a low-cost method to assist emergency response teams during wildfires. Under computational constraints, existing methods struggle to plan safe trajectories in real-time to find fires while avoiding complex 3D terrain. This thesis investigates an integrated planning approach to develop safe, computationally efficient trajectories in real-time for a team of UAS to find and map wildfires.
4. Continuum deformation supports deformable formation control with safety constraint guarantees. Existing work [14] shows its effectiveness in traversing between user-specified formation leader waypoints. This dissertation proposes a generalized deformable formation path planner. Additionally, a scheme is developed to guarantee team-wide collision avoidance given failed vehicle(s) or other pop-up obstacle(s).

### 1.3 Research Approach and Thesis Outline

Figure 1.4 shows an outline of the proposed research centered around the guidance, estimation, and communication and control of multiple UAS teams. The organization of this dissertation is described below with challenge problems 1-4 addressed as defined below in corresponding items 1-4.

1. Guarantee safety for distributed and deformable UAS formations. (Ch. 3)
2. Demonstrate and explore haptic guidance of a cooperative payload transportation system. (Ch. 4)
3. Manage a team of UAS to map wildfires via computationally efficient exploration planning and 3D terrain avoidance. (Ch. 5)
4. Plan paths to optimize and manage deformable formations offline as well as safely circumvent pop-up obstacles in real-time. (Ch. 6)

The rest of Chapter 1 describes the contributions and innovations from this dissertation. Chapter 2 of the dissertation describes the affordable custom quadrotor design, instrumentation, and automa-

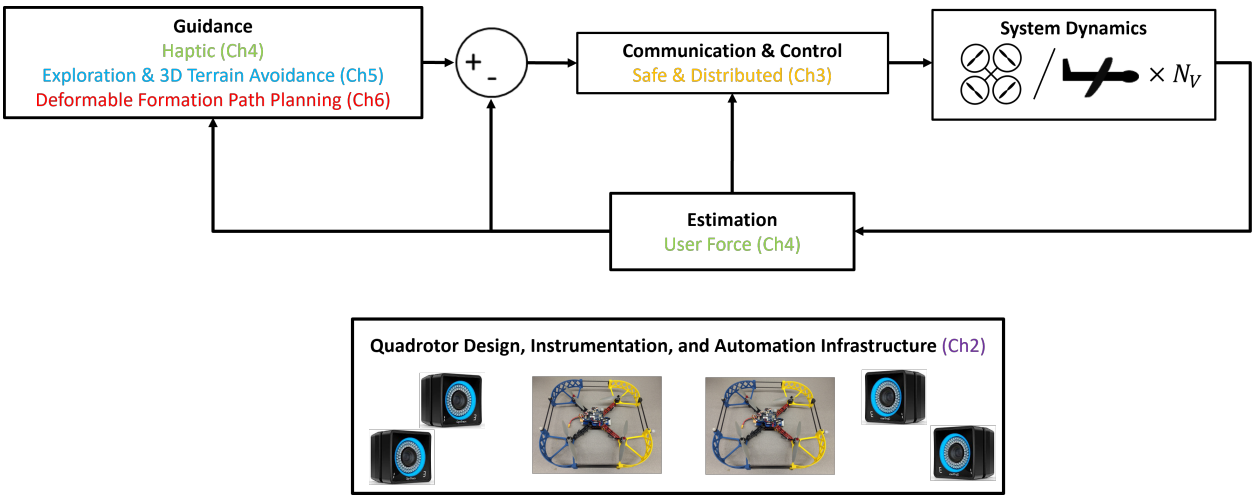



Figure 1.4: Research approach and dissertation outline

tion infrastructure developed and utilized in this work. This quadrotor system has been refined, replicated, and flight tested in cooperative control and cooperative transport experiments as well as modeled for supporting simulation studies. In Chapter 3, continuum deformation is experimentally validated and extended via five quadrotor team formation flights. The existing safety theorem relied on global deviation bounds being met to guarantee inter-agent collision avoidance and containment. However, only local controller error bounds can be obtained experimentally. This chapter derives a global deviation bound given communication weights and local bounds to guarantee a safe trajectory can be planned under continuum deformation using local communication. Chapter 4 explores haptic guidance of a multilift slung load transportation system. An experimental system consisting of an instrumented payload, user, and five quadrotor team is used to demonstrate the concept. Online user force estimation updates virtual dynamics of an admittance controller which is tracked by a payload controller in the direction of user applied force. Chapter 5 proposes computationally efficient planning methods for a team of UAS to map wildfires. A state machine handles multi-UAS task planning to distribute UAS between finding new fires (exploration mode) and mapping known fires (boundary follower mode). A 1D terrain avoidance technique works in tandem with these 2D planning modes to support efficient 3D path planning. Lastly, Chapter 6 considers the path planning problem for deformable formations by splitting it into two modules. A high level, global planner considers a static, known environment and generates a reference trajectory for the system that can translate, rotate, and deform through space while avoiding obstacles and inter-agent collision. The low-level guidance module follows planned trajectories but additionally must deal with sudden vehicle failures and pop-up obstacles. Our solution uses a fluid flow field to direct UAS traffic along streamlines and wrap each failed agent with a repulsive ”bubble” that efficiently redirects UAS traffic streamlines around the bubble without collision.

## 1.4 Contributions and Innovations

Specific contributions of this dissertation include:

- M330 Quadrotor and `rc_pilot_a2sys` flight controller open source software project including the integration of a reliable, low latency motion capture feedback system for multi-vehicle systems.
- First experimental evaluation of continuum deformation using a team of five UAS.
- Instrumented payload open source software project for cooperative payload transport with haptic guidance.
- Simulation and experimental validation of deformable formation planning and collision avoidance.

Specific innovations of this dissertation include:

- Definition of required inter-vehicle separation constraints applied to a real world experimental system for deformable formations under continuum deformation. Derivation of a global minimum separation bound based on a local controller error bound.
- Instrumentation, algorithms, and software enabling a heads-up haptic pushing interface in which a novice user can direct a heavy payload carried by multiple UAS through a crowded or cluttered environment.
- System integration of a neural network fire boundary characterization with multi-UAS path planning to support wildfire mapping over a large area.
- Hierarchical deformable formation path planning with UTM application. To-date UTM has considered only considered single vehicle routing solutions and vehicle pair detect-and-avoid solutions.

# CHAPTER 2

# Quadrotor Design, Instrumentation, and Automation Infrastructure

## 2.1 Introduction

This chapter presents the hardware and software systems developed to enable experimental realizations of the multi-UAS missions considered in this dissertation. This includes a reliable quadrotor platform (Sec. 2.2), flight controller (Sec. 2.3), low-latency motion capture integration (Sec. 2.4), and an instrumented payload (Sec. 2.5). Figure 2.1 shows a high level view of this system. The full system was used for Chapter 4 which included the slung instrumented payload and user. Chapters 3 and 6 used a subset of the system with just quadrotor vehicles.

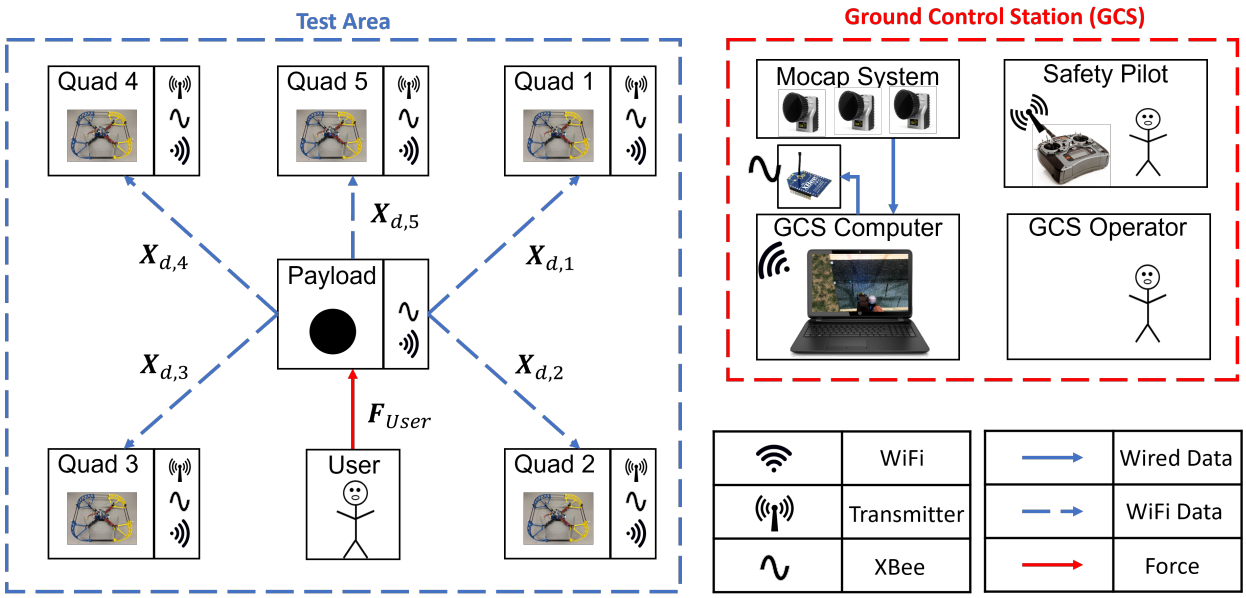



Figure 2.1: High level hardware block diagram of the experimental test bed.

## 2.2 M330-Quadrotor

The M330-Quadrotor (”M”ichigan, ”330”mm diagonal motor-to-motor distance, ”Quadrotor”) is a low-cost (≈$300) quadrotor designed using off-the-shelf components and 3D printed parts

(Figure 2.2). It was specifically designed to support multi-vehicle research and a graduate level experimental UAS course. The mass of the vehicle is 1.08kg with an enclosing circle radius of 28cm. Four 920kV brushless DC motors spin 8" x 4.5" Nylon propellers and a 4S 2650mAh LiPo battery powers the vehicle. A BeagleBone Blue single-board Linux computer runs the `rc_pilot_a2sys` flight controller to enable autonomous flight capabilities. This autopilot is available as an open source project with a bill of materials, build and operation manuals, CAD files, and software [15]. This platform's low-cost, replaceable parts, and propeller guard design facilitate multi-vehicle research. With vehicles operating close to their neighbors at theoretical safety limits, likelihood of collisions are increased, but propeller guards allow adjacent vehicles to bump off each other instead of crashing. Individual parts can be replaced, most commonly propeller guards, allowing the researcher to pursue bold new ideas in safe facilities such as M-Air.

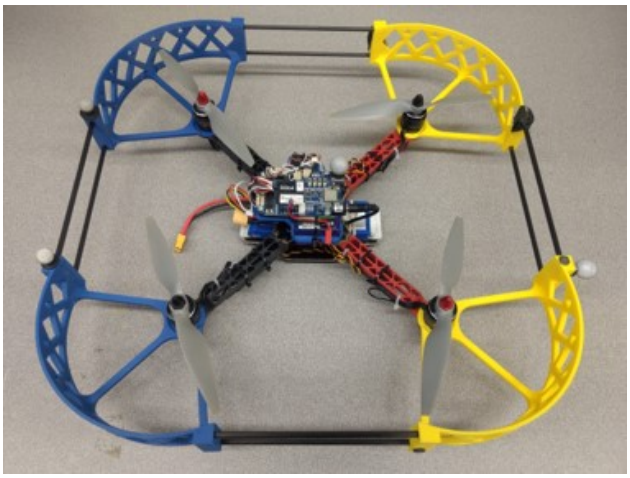


Figure 2.2: Custom M330-Quadrotor

### 2.2.1 Hardware Description

An electrical block diagram can be seen in Figure 2.3. It is mostly a standard configuration of hardware components. The main LiPo battery powers four electronic speed controllers (ESCs) and the Beaglebone Blue (BBb) through a power distribution board. The ESCs spin the brushless motors and propellers based on the pulse width modulation (PWM) signal they receive from the BBb. A DSMX receiver communicates with the BBb to allow for pilot control signals to be received. The Beaglebone Blue provides all of the necessary low level hardware interfaces including UART, I2C, PWM, DSMX, and GPIO if needed.

There are three design decisions that differ from a standard configuration. First, most quadrotors would use a microcontroller (MCU) as opposed to a full single-board Linux computer like the BBb. This gives the advantage of additional computing power and the convenience of a Linux system.

However, it gives up certain guarantees of real-time performance that MCUs can have. This is due to low-level Linux kernel operations and other scheduling details but it is mitigated by using a ”real-time” image that only contains necessary functionality. Second and partially related, another smaller battery powers the BBb to allow for hot swapping of the main battery. This is useful since each restart of the BBb would take several minutes and require additional steps otherwise.

Third, an optional kill switch (”KS”) is included that can either pass through the PWM signals from the BBb to the ESCs or pass through a GND signal to effectively stop the motors. This allows a ground control station operator to safely disable the motors in an emergency even if the BBb fails. This will be discussed further in the motion capture integration section below.

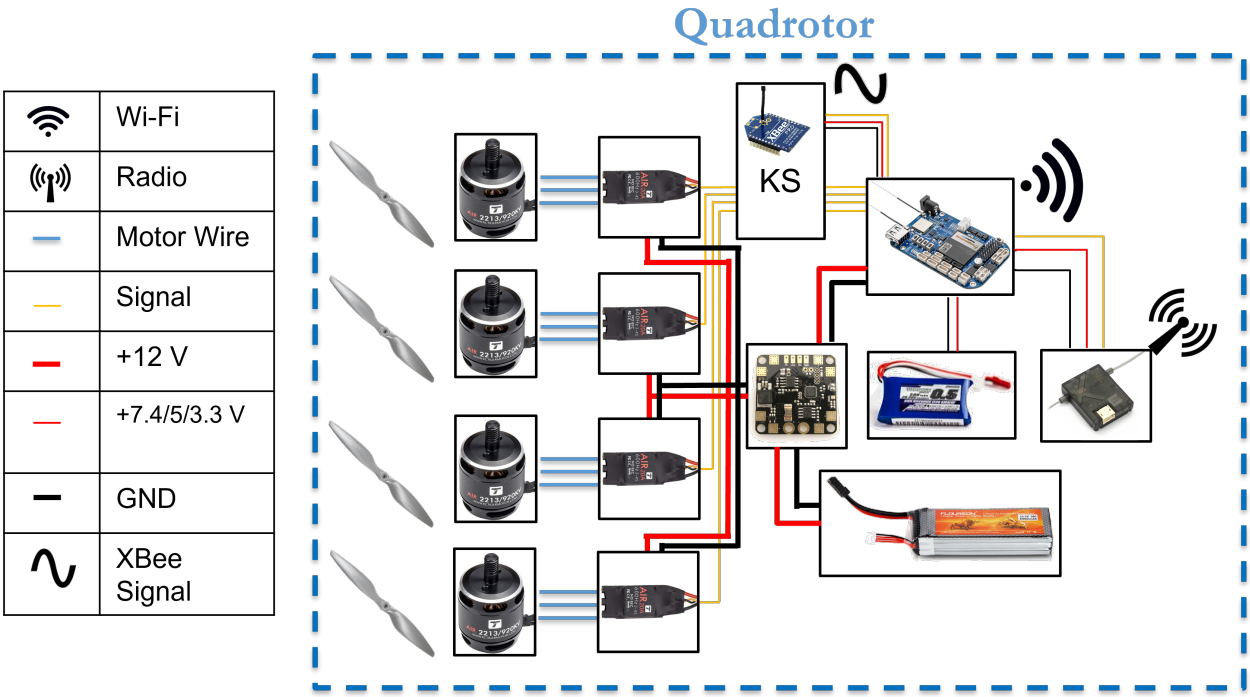


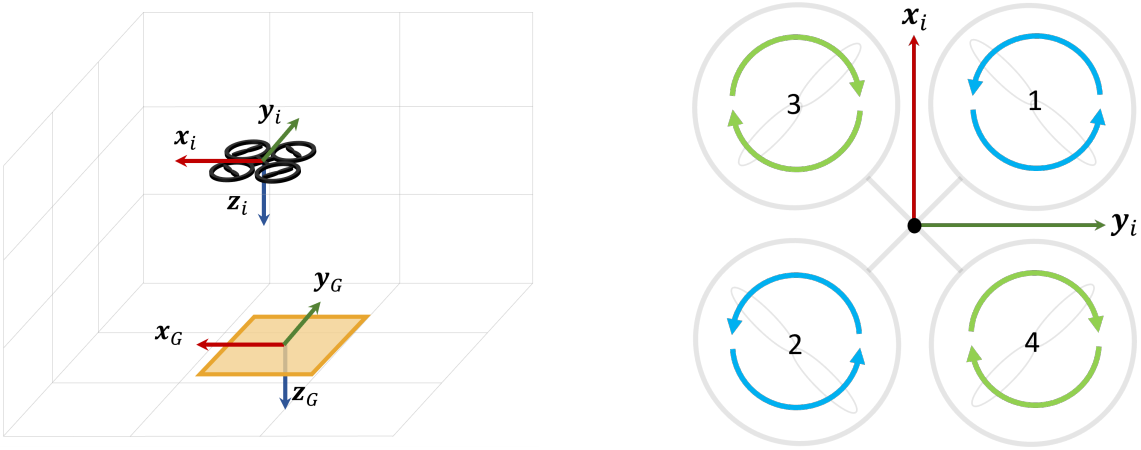

Figure 2.3: Electrical block diagram of M330-Quadrotor

### 2.2.2 Quadrotor Model

Considering $N_V$ vehicles, define an inertial North-East-Down (NED) ground reference coordinate frame $G$ with a local origin, and a coordinate frame for vehicle $i \in \{1, ..., N_V\}$ (Figure 2.4a with one vehicle shown). Define vehicle $i$ position as $\mathbf{r}_i, (i \in \{1, ..., N_V\})$. Gravity and total propeller thrust force act on each vehicle $i$ as given by

$$\mathbf{F}^i_{net,i} = \mathbf{F}^i_{Gravity,i} + \mathbf{F}^i_{Thrust,i}. \tag{2.1}$$

Propeller forces and torques apply a net torque vector on vehicle $i$ as given by




(**a**) Vehicle and Ground Reference Frames

(**b**) Quadrotor motor spin directions

Figure 2.4: System overview and motor model

$$\boldsymbol{\tau}^i_{net,i} = \begin{bmatrix} \tau_{\phi,i} \\ \tau_{\theta,i} \\ \tau_{\psi,i} \end{bmatrix}. \tag{2.2}$$

A pulse width modulation (PWM) signal value ($\delta_{j,i}$) is sent to each motor that spins it such that $F_{j,i} = c_1\delta_{j,i}$ and $\tau_{j,i} = c_2\delta_{j,i}$ are the generated forces and torques from each rotor $j \in \{1, 2, 3, 4\}$ on each vehicle $i \in \{1, ..., N_V\}$ (per Figure 2.4b). With vehicle arm length $a$ the total force and torques exerted onto the vehicle by the propellers are given by

$$\begin{bmatrix} f_{thrust,i} \\ \tau_{\phi,i} \\ \tau_{\theta,i} \\ \tau_{\psi,i} \end{bmatrix} = \begin{bmatrix} c_1 & c_1 & c_1 & c_1 \\ -\frac{\sqrt{2}}{2}ac_1 & \frac{\sqrt{2}}{2}ac_1 & \frac{\sqrt{2}}{2}ac_1 & -\frac{\sqrt{2}}{2}ac_1 \\ \frac{\sqrt{2}}{2}ac_1 & -\frac{\sqrt{2}}{2}ac_1 & \frac{\sqrt{2}}{2}ac_1 & -\frac{\sqrt{2}}{2}ac_1 \\ c_2 & c_2 & -c_2 & -c_2 \end{bmatrix} \begin{bmatrix} \delta_{1,i} \\ \delta_{2,i} \\ \delta_{3,i} \\ \delta_{4,i} \end{bmatrix}, \tag{2.3}$$

and thrust and gravity forces are

$$\mathbf{F}^i_{Thrust,i} = -f_{thrust,i}\mathbf{e}_3, \;\; \mathbf{F}^i_{Gravity,i} = R^i_G m_i g \mathbf{e}_3. \tag{2.4}$$

where $m_i$ is the mass of vehicle $i$, $R^i_G$ is the rotation matrix from frame $G$ to vehicle frame $i$, and $\mathbf{e}_3 = [0; 0; 1]$. Vehicle position is defined in the ground frame as $\mathbf{r}^G_i$. The vehicle $i$ body frame is used to define velocity vector $[u_i, v_i, w_i]$, roll/pitch/yaw attitude vector $[\phi_i, \theta_i, \psi_i]$, and angular rate vector $[p_i, q_i, r_i]$. Mass moments of inertia ($J_{x,i}, J_{y,i}, J_{z,i}$) are also defined for each vehicle $i$.

Theses forces and torques act on the quadrotor according to standard rigid body dynamics as given by

$$\dot{\mathbf{r}}_B^G = R_B^G \begin{bmatrix} u_B \\ v_B \\ w_B \end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix} \dot{u}_B \\ \dot{v}_B \\ \dot{w}_B \end{bmatrix} = \begin{bmatrix} r_B v_B - q_B w_B \\ p_B w_B - r_B u_B \\ q_B u_B - p_B v_B \end{bmatrix} + \frac{1}{m_B} \mathbf{F}_{net,B}^B, \tag{2.6}$$

$$\begin{bmatrix} \dot{\phi}_B \\ \dot{\theta}_B \\ \dot{\psi}_B \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi_B \tan\theta_B & \cos\phi_B \tan\theta_B \\ 0 & \cos\phi_B & -\sin\phi_B \\ 0 & \frac{\sin\phi_B}{\cos\theta_B} & \frac{\cos\phi_B}{\cos\theta_B} \end{bmatrix} \begin{bmatrix} p_B \\ q_B \\ r_B \end{bmatrix} \tag{2.7}$$

$$\begin{bmatrix} \dot{p}_B \\ \dot{q}_B \\ \dot{r}_B \end{bmatrix} = \begin{bmatrix} \frac{J_{y,B}-J_{z,B}}{J_{x,B}} q_B r_B \\ \frac{J_{z,B}-J_{x,B}}{J_{y,B}} p_B r_B \\ \frac{J_{x,B}-J_{y,B}}{J_{z,B}} p_B q_B \end{bmatrix} + \begin{bmatrix} \frac{1}{J_{x,B}} \\ \frac{1}{J_{y,B}} \\ \frac{1}{J_{z,B}} \end{bmatrix} \cdot \boldsymbol{\tau}_{net,B}^B, \tag{2.8}$$

where $B \in \{1, ..., N_V, P\}$ and $P$ is the payload as described later.

### 2.3 Open Source Autopilot `rc_pilot_a2sys`

The M330-Quadrotor flight controller is coded in the customized open source software package `rc_pilot_a2sys`. This software is intentionally simple, efficient, and effective in supporting research via autonomous trajectory tracking and ease of modification. It was created as a fork from `rc_pilot`, which was an incomplete project developed by James Strawson that leveraged the `librobotcontrol` library. This library provides access to the low level robotics peripherals as well as convenient math and filter functionality so that the flight controller simply has to use these functions to implement the core logic needed [16]. The original project wasn't intended to be fully functional nor for public use and it had some critical software bugs. We fixed these bugs and added some additional functionality for use in our research vehicles and are now calling the flight control software, `rc_pilot_a2sys`.

For the first two years of my PhD I used `Ardupilot` (APM) as the flight controller (also running on a BeagleBone Blue). APM supports different hardware platforms (Pixhawk, Beaglebone Blue, etc), different vehicle types (multirotor, plane, rover, sub), and many, many extra features. To enable this rich feature set, the full APM project has over 4,000 files and over 1,000,000 lines of code (300,000 in .cpp/.h files). However, the majority of these features weren't directly needed for my use case. Moreover, I needed to develop additional features to support my research. I developed a library to integrate motion capture data using XBees, added custom logging, command

line telemetry, and an operation state machine, among other things. After development, the system worked and was used in Chapter 3. However, the large codebase made it challenging to develop additional features for research and educational use cases. Specifically, bringing a new developer up to speed took too long and often discouraged those with limited development experience.

Conversely, `rc_pilot_a2sys` has twelve core (.c/.h) files and only about 10,000 lines of .c/.h code in the entire project. This is because `rc_pilot_a2sys` only supports the Beaglebone Blue and multirotors, although it has also been adapted for a quadplane which confirms ease of adaptation. It was successfully used in an experimental UAS course and by several researchers across the University of Michigan and around the country. The relative simplicity with core functionality allowed new developers to quickly learn and use the system.

Specific improvements that I made to `rc_pilot` to arrive at `rc_pilot_a2sys` include: a general overhaul of flight modes (and introduction of several new ones), a new controller (inspired by APM), a bug fix to allow for consistent control loops without dangerous delays, motion capture integration over XBee, and setpoint commands using MAVLink messages over WiFi. Some of the flight modes available include manual commanding of attitude and total thrust through an RC transmitter, a loiter mode that tracks a transmitter-commanded position, and autonomous flight that tracks pre-loaded or real-time computed trajectories. A cascaded proportional-integral-derivative (PID) controller enables setpoint tracking. State estimation is accomplished with a combination of the onboard IMU and motion capture feedback.

A high level view of the controller can be seen in Figure 2.5.

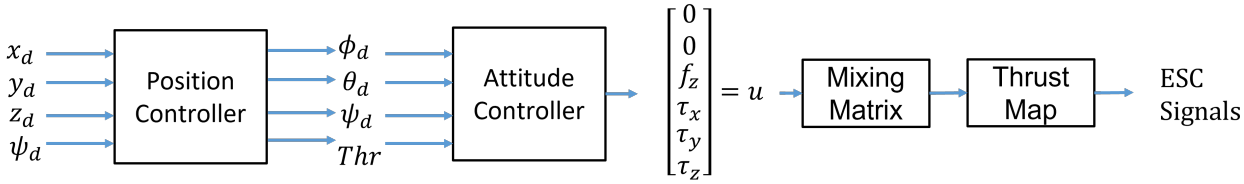



Figure 2.5: `rc_pilot_a2sys` controller block diagram

### 2.3.1 Cascaded PID Controller

The cascaded PID controller encompasses both the position and attitude controller from Figure 2.5. This results in a 4-stage cascaded PID controller to track the desired state. Tracking error in the outer loops feed as setpoints to the inner PID control loop to compute desired body frame forces and torques. Position ($\mathbf{r}^G_{d,i}$), velocity ($\dot{\mathbf{r}}^G_{d,i}$), feedforward thrust ($f_{hover,i}$), and attitudes ($\phi_{FF,i}$,$\theta_{FF,i}$, $\psi_{d,i}$) setpoints are commanded by a guidance module (pre-loaded or in real-time over WiFi). The controller is

$$\dot{\tilde{\mathbf{r}}}_{d,i} = K_{1,p}\left(\mathbf{r}^G_{d,i} - \mathbf{r}^G_i\right) + K_{1,int}\int_0^t \left(\mathbf{r}^G_{d,i} - \mathbf{r}^G_i\right) d\tau + K_{1,d}\frac{d}{dt}\left(\mathbf{r}^G_{d,i} - \mathbf{r}^G_i\right) + \dot{\mathbf{r}}^G_{d,i} \tag{2.9}$$

$$\ddot{\mathbf{r}}_{d,i} = K_{2,p}\left(\dot{\tilde{\mathbf{r}}}_{d,i} - \dot{\mathbf{r}}^G_i\right) + K_{2,int}\int_0^t \left(\dot{\tilde{\mathbf{r}}}_{d,i} - \dot{\mathbf{r}}^G_i\right) d\tau + K_{2,d}\frac{d}{dt}\left(\dot{\tilde{\mathbf{r}}}_{d,i} - \dot{\mathbf{r}}^G_i\right) \tag{2.10}$$

$$\phi_{d,i} = \frac{-s\psi_i\ddot{x}_{d,i} + c\psi_i\ddot{y}_{d,i}}{g} + \phi_{FF,i} \tag{2.11}$$

$$\theta_{d,i} = \frac{-c\psi_i\ddot{x}_{d,i} - s\psi_i\ddot{y}_{d,i}}{g} + \theta_{FF,i} \tag{2.12}$$

$$f_{thrust,i} = \frac{f_{hover,i} + \ddot{z}_d}{c\phi_i c\theta_i} \tag{2.13}$$

$$\omega_{d,i} = K_{3,p}\left(\mathbf{\Omega}_{d,i} - \mathbf{\Omega}_i\right) + \omega_{FF,i} \tag{2.14}$$

$$\tau_i = K_{4,p}\left(\omega_{d,i} - \omega_i\right) + K_{4,int}\int_0^t \left(\omega_{d,i} - \omega_i\right) d\tau + K_{4,d}\frac{d}{dt}\left(\omega_{d,i} - \omega_i\right), \tag{2.15}$$

where $\dot{\tilde{\mathbf{r}}}_{d,i}$ is an intermediate control desired velocity and $(\phi_{d,i}, \theta_{d,i}, \psi_{d,i})$ is the desired attitude.

This controller itself (Eqs. 2.9-2.15) is not novel, but rather represents a common open source control approach that many off-the-shelf vehicles utilize. The inner loop controller uses linear Euler angle error dynamics. However, this remains valid and effective when aggressive angles are not used. However, quaternion-based error dynamics and others that properly represent the entire space of $SO(3)$ could be used to allow the system to perform well at all angles as opposed to just near hover [17, 18, 19].

### 2.3.2 Motor Mapping and Saturation

The output of the cascaded PID controller is a desired thrust force ($f_{thrust,i}$) and torques ($\tau_{\phi,i}, \tau_{\theta,i}, \tau_{\psi,i}$). The goal of the motor mapping module is to determine what ESC signals to send to achieve that (per Eq. 2.3).

Since $\delta_i$ is normalized from 0 to 1 we can calculate the maximum and minimum values for the torques and force, and then normalize the mixing matrix to arrive at

$$\begin{bmatrix} f_{z,norm} \\ \tau_{x,norm} \\ \tau_{y,norm} \\ \tau_{z,norm} \end{bmatrix} = \begin{bmatrix} -0.25 & -0.25 & -0.25 & -0.25 \\ -0.5 & 0.5 & 0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix} \tag{2.16}$$

And inverting we have

$$\begin{bmatrix}\delta_1\\ \delta_2\\ \delta_3\\ \delta_4\end{bmatrix} = M_{4x2}\begin{bmatrix}0\\ 0\\ f_z\\ \tau_x\\ \tau_y\\ \tau_z\end{bmatrix} \tag{2.17}$$

$$M_{4x2} = \begin{bmatrix}0 & 0 & -1 & -0.5 & 0.5 & 0.5\\ 0 & 0 & -1 & 0.5 & -0.5 & 0.5\\ 0 & 0 & -1 & 0.5 & 0.5 & -0.5\\ 0 & 0 & -1 & -0.5 & -0.5 & -0.5\end{bmatrix} \tag{2.18}$$

Note that this mixing matrix is specific to the M330-Quadrotor platform but for arbitrary multirotor platforms a different matrix can be derived and used interchangeably.

In most cases, Eq. 2.17 can be used directly without issue. However, motor commands are constrained to be between 0 and 1 and so therefore a more careful approach needs to be taken.

The approach taken in `rc_pilot_a2sys` is to pass through desired forces and torques one by one, keeping track of the cumulative motor command values and making sure not to saturate them at any point as

$$\begin{bmatrix}\delta_1\\ \delta_2\\ \delta_3\\ \delta_4\end{bmatrix} = M_{4x2}\left(\begin{bmatrix}0\\ 0\\ f_{z,1}\\ 0\\ 0\\ 0\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ \tau_x\\ 0\\ 0\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 0\\ \tau_y\\ 0\end{bmatrix} + \begin{bmatrix}0\\ 0\\ f_{z,2}\\ 0\\ 0\\ 0\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 0\\ 0\\ \tau_z\end{bmatrix}\right). \tag{2.19}$$

This is a mathematical representation of the algorithm. Throttle force is separated into two components $f_{z,1}$ and $f_{z,2}$ where $f_{z,1} = min(f_z, 0.5)$ and $f_z = f_{z,1} + f_{z,2}$. First, up to half of the throttle is passed through ($f_{z,1}$). Second, roll torque is passed through ($\tau_x$ saturated to 0.2 magnitude in motor space). Third, pitch torque is passed through ($\tau_y$ saturated to 0.2 magnitude in motor space). Fourth, any remaining throttle force is passed through ($f_{z,2}$). Fifth, yaw torque is passed through ($\tau_z$ saturated to 0.2 magnitude in motor space). Due to the structure of the mixing matrix, all three torques saturate at 0.4 magnitude (but this could be different for differing vehicle models). This means that 0.5 throttle, roll, and pitch are all guaranteed to run without saturation issues (this adds to 0.9 out of 1.0 max). However, the full throttle and yaw are not guaranteed to run without being saturated out. This is a design choice that allows the vehicle to maintain level flight and give up

some throttle and all yaw control authority when faced with saturation. It was chosen this way particularly for the payload transportation mission (Chapter 4) but in general saturation could be designed to prioritize these control objectives in different ways if the mission necessitated it. For example, in drone racing, pilot feedback is provided via first-person-view (FPV) goggles and so yaw may also be important to prioritize over throttle such that a reasonable view could be maintained.

## 2.4 Motion Capture Integration

Real-time motion capture supports accurate state estimation allowing research to focus on guidance, control, and higher-level automation. An eight Optitrack camera system with a 3m x 4m trackable area was used for indoor testing. Outdoors, a 32 Qualisys camera system with a 10m x 10m trackable area in M-Air, Michigan's netted flight facility, was used. Each rigid body of interest, quadrotors and instrumented payload, was equipped with motion capture markers that are tracked in real-time by the motion capture system. This data is then relayed through 900MHz, low-latency XBee wireless serial modems to the quadrotors and instrumented payload. This method supports low-latency pose feedback for up to six entities (vehicles/payload) simultaneously.

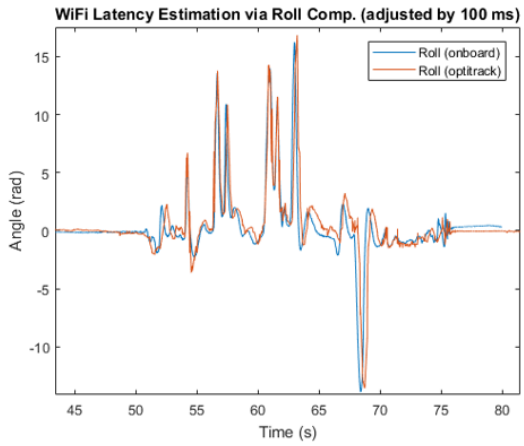
### 2.4.1 XBee vs. WiFi

Both motion capture applications (Qualisys Tracker Manager and Optitrack Motive) run on a ground control station computer which is connected to a network of cameras via ethernet. 2D camera data is received and the application uses multiple observations of markers to estimate their 3D positions. Additionally, groups of markers are assigned to rigid bodies which can have their positions and orientations estimated. This information can be shared in real-time using provided sample real-time client code via UDP multicast.

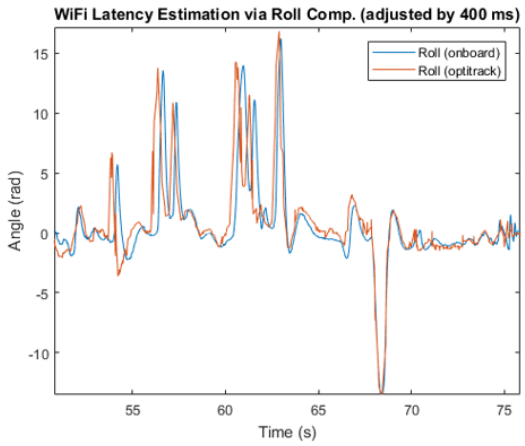
As a first pass, we connected a quadrotor via UDP multicast over WiFi to the data stream coming from Optitrack Motive. We performed a very smooth roll motion that was sensed by both the onboard IMU and motion capture system. Figure 2.6 shows the results. Data came in near a 100ms latency for most of the test, but went up to 400ms for later portions. This is quite high and would be unacceptable for any reasonable feedback control system.

Next, we considered using 900MHz wireless serial XBee modules (XBeePRO900HP). This required another system in between the motion capture application and the receiving XBee to "relay" the data and transform it from the UDP packet into a minimal serial packet. The same test was performed and the results are shown in Figure 2.7. Motion capture data came in at a consistent 40ms delay as compared with the onboard IMU. This value is very reasonable to support aggressive feedback control systems and so XBees were chosen for this purpose.

Separately, a comparison was done between 2.4GHz and 900MHz XBees. It was determined



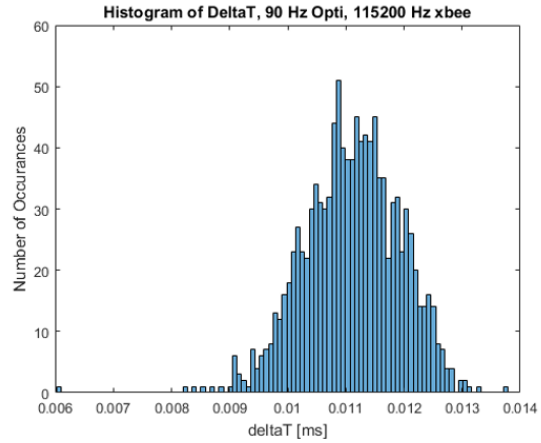

(**a**) Motion capture data is delayed by at least 100ms



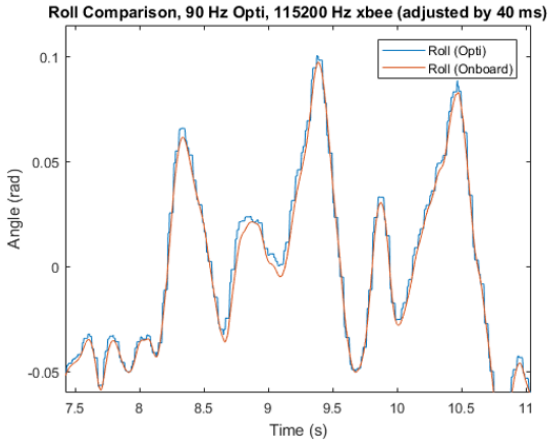

(**b**) Motion capture data is delayed by up to 400ms

Figure 2.6: Motion capture data sent over a WiFi link with latency between 100ms and 400ms.

that the 2.4GHz XBees experienced intermittent interference to cause some drop packets, likely due to many devices operating on that frequency (WiFi, bluetooth, etc). However, the 900MHz XBees experienced near zero drop packets.




(**a**) XBee packet receipt histogram (1/90Hz peak).




(**b**) A 40ms shift aligns the data very well.

Figure 2.7: Motion capture data sent over an XBee link with latency consistently at 40ms.

### 2.4.2 Six Simultaneous Senders

Reliable motion capture data feedback was achieved for a single vehicle in the previous section. However, for the multi-UAS missions considered in this dissertation it was required to develop a reliable system for six receivers (five vehicles and one payload). Additionally, an optional ”kill switch” was needed such that the ground control station operator could flip a switch and disable the motors at any time.

Figure 2.8 shows the system developed to achieve this. The Optitrack Relay Cape attaches to a Beaglebone Black which receives motion capture data packets over ethernet and sends out serial (UART) data packets to the custom PCB. Six XBees connect (five through hardware UART ports, one through a USB UART dongle) and receive the UART data over separate pins. A physical switch on the GCS PCB sets the state of a pin on the sending XBee which is mirrored through a feature of the XBees on the receiving XBee. When the switch is toggled, the state of the pin on the quadrotor side toggles, which is logically ANDed (via hardware logic gates) alongside the ESC signals. This means that even if the BBb fails and commands the motors erroneously for any reason, a completely separate system can safely shut them down. Additionally, a timeout is set such that if communication is lost between the XBees, the default state is to kill the motors. This was essential during development and testing of an autonomous roofing concept using a nailgun-equipped octocopter [20]. Additional details of these PCBs are included in Sec. A.1.1 and Sec. A.1.2.

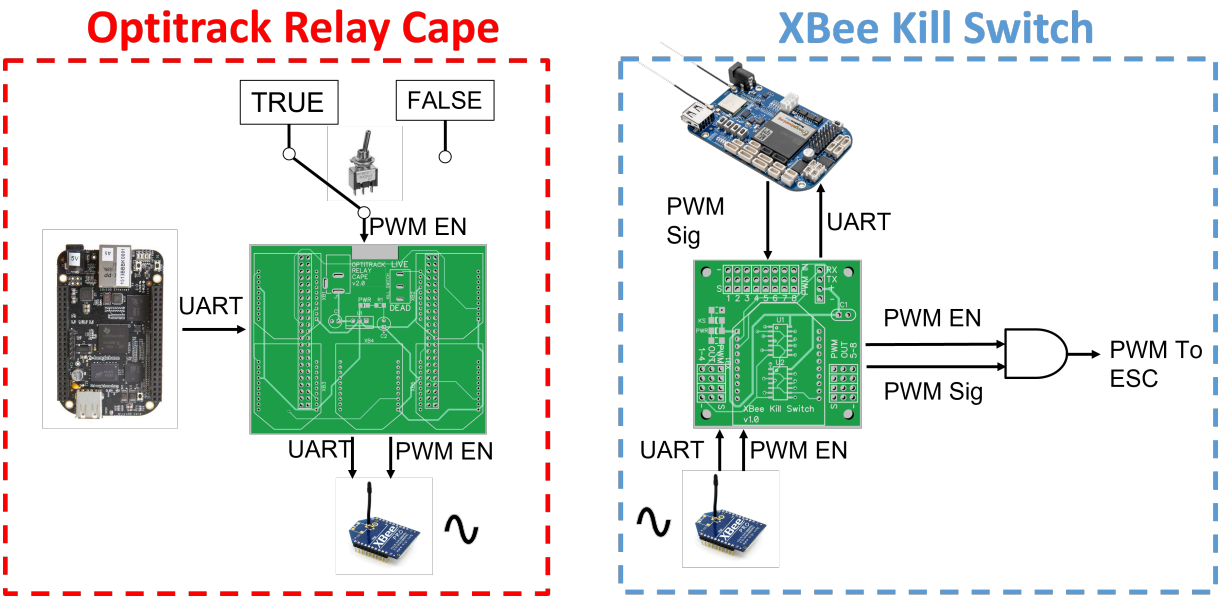



Figure 2.8: XBee system that simultaneously sends motion capture data and kill switch signals.

The specific serial packet used is shown in Figure 2.9. The onboard IMU estimates roll and pitch effectively, but does a poorer job with yaw and has no direct way of measuring global position. Therefore, yaw and 3D position are sent along with a timestamp. Two start Bytes and two checksum Bytes are also sent to allow for explicit error handling during transmission. The XBee modules themselves were configured to be transparent serial devices with minimal error handling features enabled. Specific settings can be found in the M330-Quadrotor project [15].

While 900MHz XBees were chosen to avoid interference with over devices, consideration was needed to avoid interference between the XBees themselves. This was performed via time and

| FRAMING BYTES | | Data | | | | | Checksum |
|---|---|---|---|---|---|---|---|
| 0x1B | 0xFE | time_us | yaw | x | y | z | Fletcher16 |
| 1 | 1 | 4 | 4 | 4 | 4 | 4 | 2 |

Figure 2.9: XBee serial packet with number of Bytes listed per entry (24 Bytes total).

frequency separation simultaneously as shown in Figure 2.10. The 900MHz band has 63 channels and each XBee requires a minimum of 25 enabled channels to operate. Therefore, two independent network of XBees were setup, 31 channels each with an unused channel to separate them. In time, three sending periods were allocated such that no two XBees sent on the same channel at the same time resulting in a total of six reliable XBee transmissions per sequence. Additionally, 57,600 baud was the value experimentally determined to be the most reliable while also giving a high data rate. Therefore, the theoretical throughput limit of this transmission strategy is

$$\frac{1Sequence}{3Packets} \times \frac{1Packet}{24Bytes} \times \frac{1Byte}{8bits} \times \frac{57,600bits}{1s} = 100Hz. \tag{2.20}$$

However, in practice 60 Hz was the value that we were able to use while achieving reliable performance.

Lastly, another benefit of this setup was the ability to send ”fake” motion capture data to facilitate software testing. Instead of sending real motion capture data, the serial relay application can send specific data to help test and debug the flight controller. Testing in M-Air with a full system is very time consuming and requires additional personnel. However, testing via fake motion capture data doing software in the loop tests is much faster and efficient to find bugs, not to mention the added benefit of not crashing the vehicle.

## 2.5 Instrumented Payload

In support of the cooperative payload transport haptic guidance mission, an instrumented payload capable of real-time sensor data acquisition was developed (Figure 2.11). This device was custom-built for this project and the bill of materials, build guide, and software for the system can be found in an open source project [21]. The rest of this section describes the hardware, construction, and core software while Chapter 4 details the research use case and operation.

### 2.5.1 Overview

Five tether lines with inline load cell tension sensors are wrapped around a 24.6cm Styrofoam ball in guide channels cut to prevent slippage (Figure 2.11d). Using Styrofoam as the main structural component allowed the payload to be both lightweight and easy to manufacture through the use of channels and inserts cut with a hot knife. This allowed us to rapidly iterate on the design by

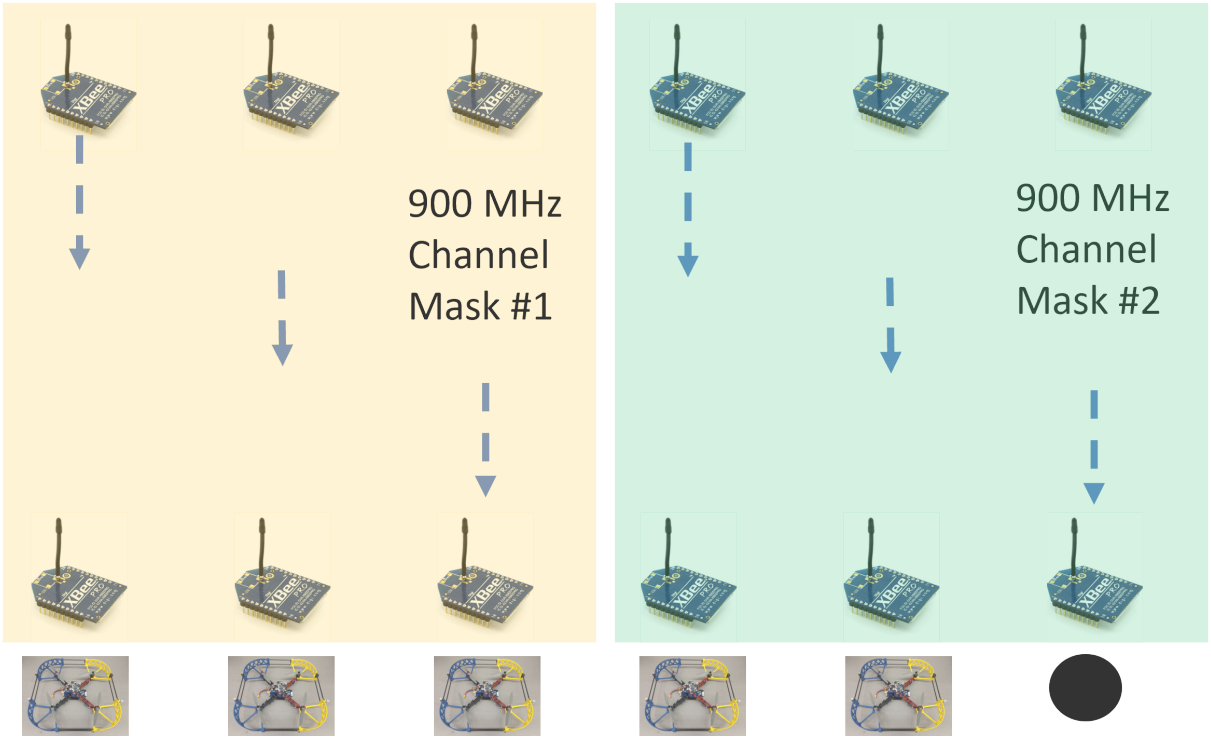
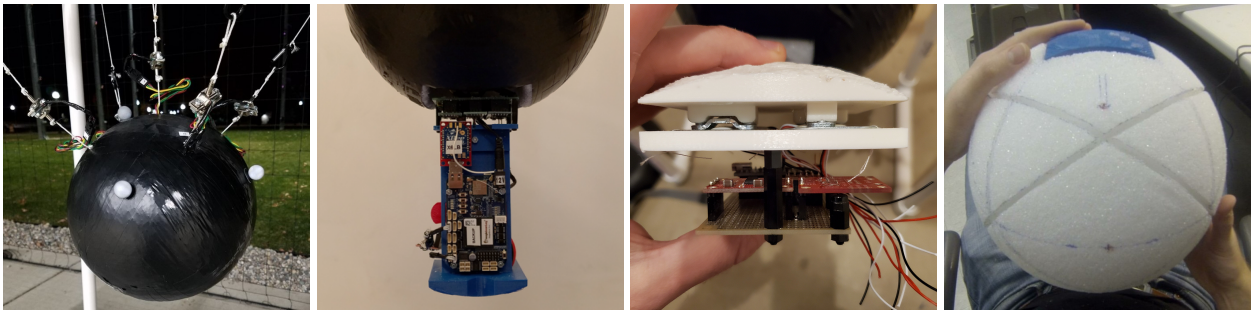



Figure 2.10: XBee strategy using six XBees

adding new inserts and wire channels as they were required without rebuilding the entire payload. The outer surface of the ball is a weak point where the Styrofoam could break into smaller pieces, however, this was rectified by wrapping the ball in electrical tape. Not only does this maintain structural integrity for the payload in the face of collisions and external forces, this also provides a smooth surface for lower aerodynamic drag. On the bottom, the Styrofoam has an 8cm square extrusion cut to an 18cm depth to host the main payload insert (Figure 2.11b). The end of the main payload insert houses a custom printed circuit board (PCB) that receives sensor cables from the load cells. The PCB contains five HX711, 24-bit ADCs (one for each sensor) that are sampled at 80Hz by an Arduino Nano. This PCB is described in further detail in Sec. A.1.3. On the front, the Styrofoam has another square extrusion cut to host the push sensor insert (Figure 2.11c). The push sensor insert holds a push force sensor (four compression load cells with a combinator) and one HX711 that is sampled with another Arduino Nano at 10Hz. A BeagleBone Blue single-board Linux computer receives tension sensor and push sensor data from the Arduino Nanos serially, receives motion capture data over wireless serial (XBee), and samples its onboard 9-axis IMU. The BeagleBone Blue WiFi communicates with the ground control station and all quadrotors. Power is provided by a 3s, 3000mAh Lithium Polymer (LiPo) battery. All of the electronics in both inserts are mounted to custom 3D-printed frames with spherical end caps to fit flush with the payload

surface. Once inserted, the end caps are covered in electrical tape to smoothly secure the payload. The instrumented payload weighs 1.211kg. A hardware diagram of the payload can be seen in Figure 2.12. The tethers are arranged symmetrically at a $45°$ cone angle with an additional tether on top.



(**a**) In PVC test rig (**b**) Main Insert (bottom) (**c**) Validation Sensor (**d**) Construction

Figure 2.11: Instrumented payload constructed with embedded sensing and computation.

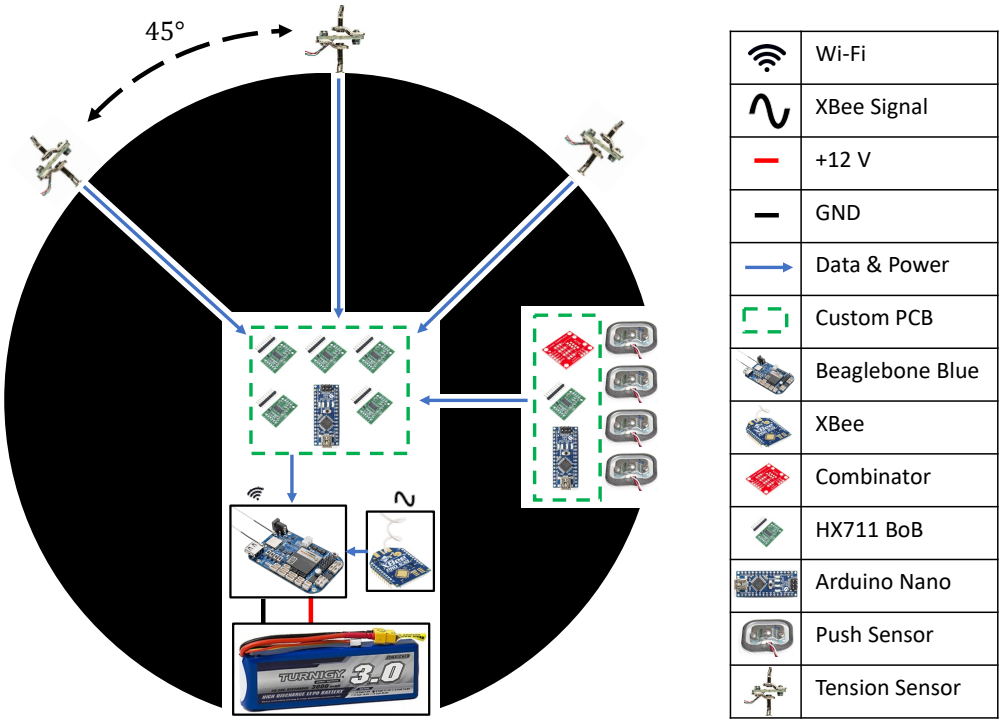



Figure 2.12: Payload hardware block diagram. Real-time sensor data is collected for use in our control algorithms.

### 2.5.2 Force Sensor Calibration

The force sensors used in the payload are load cells. They change resistance based on the applied force; these resistances are measured by ADCs. We assume the sensors are linear such that

$$o_j = s_i m_j + d_i, \tag{2.21}$$

with measurement $o_j$ coming from mass $m_j$ with a scale of $s_i$ and offset of $d_i$ for sensor $i \in \{1, ..., N_V, P\}$. Ideally, these parameters should be constant; however, these are sensitive to temperature and other factors and must be calibrated. We used a two-step calibration routine for all five tension sensors and the push force sensor. Given two measurements ($o_1$ of mass $m_1$ and $o_2$ of mass $m_2$) we can solve for the scale and offset according to

$$s_i = \frac{o_2 - o_1}{g(m_2 - m_1)}, \quad d_i = m_1 - (s_i m_1 g). \tag{2.22}$$

For the tension sensor we used two masses of 0.410kg and 1.372kg to cover the range of expected forces (0 to payload mass of 1.211kg and beyond). For the push force sensor we used the payload mass and no mass. To obtain the force applied after calibration we rearrange the equation as

$$f_j = \frac{o_j - d_i}{s_i}. \tag{2.23}$$

### 2.5.3 Tether Tension Sensor Data Processing

After converting the raw ADC reading, $o_j$, for tension sensor $i$, into a force reading in Newtons, $f_j$, it is still potentially noisy. To account for this we constrain the force reading to be between 0N and 20N, defaulting to the previously obtained reading if out of those bounds to obtain $\bar{f}_i$. Additionally, this is still only a 1D reading. However, we require a 3D force vector. We circumvent this problem by assuming that the vehicles are in their desired, known formation to obtain the direction vectors and combine this with our force data from the load cell tension sensors such that

$$\mathbf{F}^P_{Tether,i} = R_{z,\gamma_i} R_{y,\alpha_i} \bar{f}_i. \tag{2.24}$$

### 2.5.4 Push Force Sensor Data Processing

The push force sensor also only provides a 1D force reading. Since it is rigidly attached to the payload, we make the assumption that pushes will always be in the direction of the sensor, i.e. $\mathbf{F}^P_{User} = \begin{bmatrix} f_{Push} & 0 & 0 \end{bmatrix}^T$, where $f_{Push}$ is the measured 1D force, thus

$$\mathbf{F}^G_{User} = R^G_P \begin{bmatrix} f_{Push} & 0 & 0 \end{bmatrix}^T. \tag{2.25}$$

## 2.6 Conclusions and Future Work

This chapter presented the hardware and embedded software developed to support the research in the rest of the dissertation. Key elements include a quadrotor platform, real-time flight controller, low-latency motion capture integration, and an instrumented payload.

These systems were effective in supporting the research in this dissertation. However, there is room for future work to explore additional state-of-the-art research. The `rc_pilot_a2sys` flight controller could have additional features (more flight modes, more sensor support). Larger variants of the M330-Quadrotor could be pursued to enable carrying larger payloads. This is very important for supporting state estimation via onboard sensing alone. A companion computer and additional sensors could be mounted to achieve this. Lastly, the core principles of simplicity, low-cost, and open-source should be upheld for all of these future developments so that they are accessible by everyone.

# CHAPTER 3

# Distributed Quadrotor Deformable Formation Control

## 3.1 Introduction

Existing formation control methods such as containment control [22, 6], virtual structure [7], and consensus [23] support rigid formations of agents that can translate and rotate. However, rigid formations have certain limitations including fitting through narrow passages. Continuum deformation is a cooperative control technique that can be used for versatile, deformable formation control. A theoretical foundation has been laid that supports coordination and provides certain safety guarantees but results have been restricted to simulations. This chapter improves upon the existing continuum deformation theory considering formation control of a team of five quadrotors using a three leader, two follower continuum deformation specification.

Continuum deformation is a distributed leader-follower algorithm and treats all agents as points in a deformable rigid body (in two dimensions this is called the ”leading triangle”). Based on an initial configuration of the agents, the leading triangle is deformed according to a desired homogeneous transformation. The leaders are commanded directly, whereas the followers update their desired positions based on a weighted sum of their neighbors' actual positions, which are acquired via local communication. Based on a global deviation bound, constraints are placed on the homogeneous transformation that guarantee inter-agent collision avoidance and containment of all the agents within the leading triangle [9, 24]. In this chapter, a local deviation bound is determined empirically for a single quadrotor (Fig 3.1A). This is used as the global deviation bound to generate leader agent trajectories that put the system at the theoretical safety limit (assuming local deviation is equal to global deviation). The five-quadrotor team using motion capture feedback is deployed in tests where leaders execute the prescribed trajectories. Followers either use globally desired positions (guarantees safety) or update their desired positions based on the communication topology in Fig. 3.1C (pushes system beyond safety limit). The results are analyzed and the actual global deviation bound is derived.

Sec. 3.2 compares related work. Sec. 3.3 summarizes continuum deformation theory, while Sec. 3.4 describes leader flight planning and presents test trajectories. Sec. 3.5 summarizes the

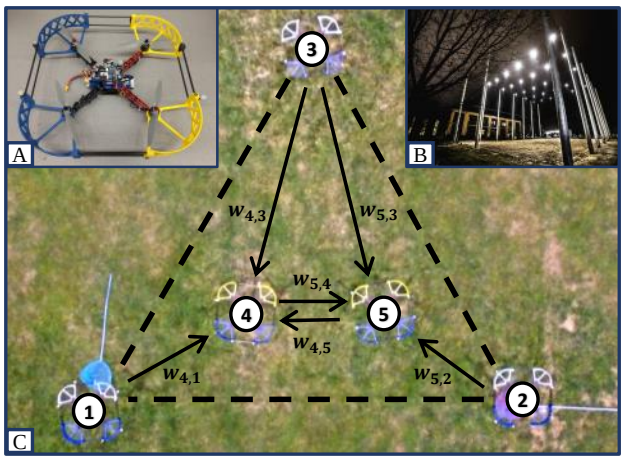



Figure 3.1: A) Test quadrotor, B) M-Air netted facility, C) Mid-flight, overhead snapshot in M-Air with three leaders (1,2,3) and two followers (4,5) flying a continuum deformation.

experimental apparatus including quadrotors, electronics, sensors, and software. Sec. 3.6 presents flight test results, a global deviation bound is derived in Sec. 3.7 and Sec. 3.8 concludes the chapter.

## 3.2 Related Work

Commonly, there are two approaches in cooperative control: centralized (a single station controls every vehicle) and distributed (there is no central station and computation and control happen onboard of each vehicle, with information being shared between neighbors) [25]. Consensus [23, 26], containment control [22, 6], and continuum deformation [9, 27] are examples of the distributed approach. Virtual structure [28] is an example of the centralized approach. Consensus is the most commonly-applied decentralized cooperative control technique [29, 25, 26, 30]. Distributed consensus was applied for agent coordination in [31, 32] and flight tested in [33, 34]. Consensus guided by a single leader is studied in [35, 36] and flight tested in [37, 38]. Cooperative control has been applied to unmanned aircraft system (UAS) teams for tasks such as surveillance [39], area surveys [40], and payload delivery [12].

Continuum deformation is a multi-agent control technique that allows translation, rotation, and shearing of a bounding envelope while ensuring agents remain within the bounding envelope and avoid collisions. To do this, continuum deformation treats agents as particles of a body that deforms under a homogeneous transformation [9, 24]. A desired $n$-D homogeneous transformation ($n = 1, 2, 3$) is defined by $n + 1$ leaders in $\mathbb{R}^n$ and acquired by followers through local communication. Shared state data is weighted based on a reference configuration. Continuum deformation stability is analyzed in [9], while coordination under switching communication topologies is studied in [27]. Characterization of the homogeneous transformation and safety guarantees are key features of continuum deformation.

## 3.3 Continuum Deformation Review

### 3.3.1 Continuum Deformation Definition

An $n-D$ continuum deformation ($n \in \{2,3\}$) is defined by homogeneous transformation $t \geq t_0, \mathbf{r}_{i,HT}(t) = \mathbf{Q}(t,t_0)\mathbf{r}_{i,0} + \mathbf{d}(t,t_0)$, where the initial time is $t_0$, $\mathbf{Q}(t,t_0) \in \mathbb{R}^{n\times n}$ is a non-singular deformation matrix, $\mathbf{d}(t,t_0) \in \mathbb{R}^{n\times 1}$ is a displacement vector, and $\mathbf{r}_{i,HT}(t) \in \mathbb{R}^{n\times 1}$ is the *global* desired formation position of agent $i$. Note that $\mathbf{d}(t_0,t_0) = \mathbf{0} \in \mathbb{R}^{n\times 1}$ and $\mathbf{Q}(t_0,t_0) = \mathbf{I}_n \in \mathbb{R}^{n\times n}$, so $\mathbf{r}_{i,0} = \mathbf{r}_{i,HT}(t_0)$ is the initial position of quadrotor $i \in \mathcal{V}$. For this chapter, a 2D continuum deformation was considered ($n = 2$).

Assume an $N_V$-agent quadrotor team $\mathcal{V} = \{1, \cdots, N_V\}$ with leaders $\mathcal{V}_L = \{1,2,3\}$ and followers $\mathcal{V}_F = \{4, \cdots, N_V\}$. Leaders form a *leading triangle* for all $t \geq t_0$ such that all followers are contained within it. $\mathbf{Q}$ and $\mathbf{d}$ uniquely relate to leader positions as in [9] by $t \geq t_0$, $\begin{bmatrix} \text{vec}\left(\mathbf{Q}^T\right) \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_2 \otimes \mathbf{P}_0 & \mathbf{I}_2 \otimes \mathbf{1}_3 \end{bmatrix}^{-1} \text{vec}\left(\mathbf{P}_d\right)$, where "$\otimes$" is the Kronecker product symbol, $\mathbf{I}_2 \in \mathbb{R}^{2\times 2}$ is the identity matrix, and $\mathbf{1}_3 \in \mathbb{R}^{3\times 1}$ is a vector of ones, $\mathbf{P}_0 = \begin{bmatrix} \mathbf{r}_{1,0}^T \\ \mathbf{r}_{2,0}^T \\ \mathbf{r}_{3,0}^T \end{bmatrix} \in \mathbb{R}^{3\times 2}$, $\mathbf{P}_d = \begin{bmatrix} \mathbf{r}_{1,d}^T \\ \mathbf{r}_{2,d}^T \\ \mathbf{r}_{3,d}^T \end{bmatrix} \in \mathbb{R}^{3\times 2}$.

### 3.3.2 Continuum Deformation Acquisition

A weighted directed graph $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$ defines inter-agent communication as shown in Fig. 3.1C. Given edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, the in-neighbor agents of follower $i \in \mathcal{V}_F$ are defined by

$$\forall i \in \mathcal{V}_F, \qquad \mathcal{N}_i = \{j | i \in \mathcal{V}_F,\ j \in \mathcal{N}_i\}.$$

For an $n$-D continuum deformation, $\left|N_i\right| = n+1,\ \forall i \in \mathcal{V}_L$ and in-neighbor agents of every follower form an $n$-D polytope. Therefore, each follower communicates with three in-neighbor agents, forming a triangle in a 2D continuum deformation. Let $i_1$, $i_2$, and $i_3$ denote index numbers of follower $i$'s in-neighbor agents, initially positioned at $\mathbf{r}_{i_1,0}$, $\mathbf{r}_{i_2,0}$, and $\mathbf{r}_{i_3,0}$. Then, communication weights of follower $i \in \mathcal{V}_F$ denoted $w_{i,i_1}$, $w_{i,i_2}$, and $w_{i,i_3}$ are given by [9]:

$$\begin{bmatrix} w_{i,i_1} \\ w_{i,i_2} \\ w_{i,i_3} \end{bmatrix} = \begin{bmatrix} x_{i_1,0} & x_{i_2,0} & x_{i_3,0} \\ y_{i_1,0} & y_{i_2,0} & y_{i_3,0} \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_{i,0} \\ y_{i,0} \\ 1 \end{bmatrix}. \tag{3.1}$$

Let $\mathbf{r}_i$ denote actual position and $\mathbf{r}_{d,i}$ be the *local desired* position of each agent $i \in \mathcal{V}$.

**Local Communication Guidance** Under "Local Communication" guidance, leader agents have direct access to their global desired position, $\mathbf{r}_{i,HT}$, but followers acquire this via a weighted sum of neighbors' actual positions via local communication.

$$\mathbf{r}_{d,i} = \begin{cases} \mathbf{r}_{i,HT} & i \in \mathcal{V}_L \\ \sum_{j \in \mathcal{N}_i} w_{i,j} \mathbf{r}_j & i \in \mathcal{V}_F \end{cases} \tag{3.2}$$

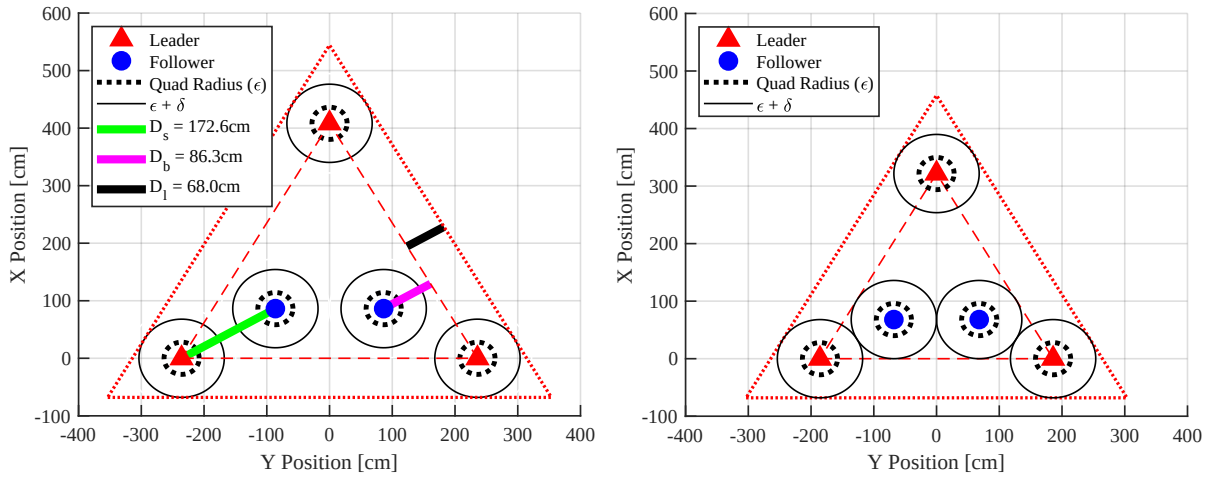
where $w_{i,j}$ is the weight on the directed edge of a communication graph from $j$ to $i$.

**Global Desired Guidance** Under "Global Desired" guidance all agents have direct access to their global desired positions.

$$\mathbf{r}_{d,i} = \mathbf{r}_{i,HT} \quad i \in \mathcal{V} \tag{3.3}$$

### 3.3.3 Continuum Deformation Coordination Safety

Given leader desired trajectories, we define a bounding triangle as a dilation of the original leading triangle with the following properties: (1) Both leading and bounding triangles have a common centroid at any time $t$, (2) Parallel sides of both triangles are separated by $D_l$ at any time $t$ (see Fig. 3.2).




(**a**) Initial configuration.

(**b**) Contracted configuration.

Figure 3.2: Leaders and followers shown in their global desired positions for initial and contracted configurations, respectively, with leading (smaller) and bounding (larger) triangles shown. $D_s$, $D_b$, and $D_l$ are shown in the initial configuration.

**Theorem 1.** *Assume each quadrotor is enclosed by a ball of radius $\epsilon$. $D_s$ denotes the minimum separation distance between any pair of agents at $t = t_0$, $D_b$ is the minimum distance from any*

*follower to the leading triangle boundary, and $D_l = \delta_g + \epsilon$ is the distance of two parallel sides of the leading and bounding triangles. Here, $\delta_g$ is the maximum deviation from the global desired position for every quadrotor: $\|\mathbf{r}_i - \mathbf{r}_{i,HT}\| \leq \delta_g, \forall i \in \mathcal{V}$. Additionally, define a local deviation bound as $\|\mathbf{r}_i - \mathbf{r}_{d,i}\| \leq \delta_l, \forall i \in \mathcal{V}$ where $\mathbf{r}_{d,i}$ is the local desired position being tracked for every agent. Let*

$$\delta_{\max} = \min\{\frac{D_s - 2\epsilon}{2}, D_b - \epsilon\}. \tag{3.4}$$

*Inter-agent collision avoidance is guaranteed, no follower leaves the leading triangle, and no agent leaves the bounding triangle if*

$$\lambda_{\min} \leq \inf_{\forall t} \left\{\lambda_1(t), \lambda_2(t)\right\}, \tag{3.5}$$

*where*

$$\lambda_{\min} = \frac{\delta_g + \epsilon}{\delta_{\max} + \epsilon}, \tag{3.6}$$

*$\lambda_1$ and $\lambda_2$ denote eigenvalues of matrix $\mathbf{U}_D = \left(\mathbf{Q}^T\mathbf{Q}\right)^{\frac{1}{2}}$.*

*Proof.* See the proof in [41]. □

### 3.4 Generating the Leading Triangle Path

Flight tests investigate if real-world continuum deformation will satisfy four constraints: follower containment within the leading triangle, collision avoidance between agents, bounding of deviation from local desired position, and bounding of deviation from global desired position. To evaluate, leaders are set as close together as they theoretically can be. Leader separation is given by scaling factor $\lambda_{\min}$ (Eq. 3.6). With $\epsilon$ = 28cm (Sec. 3.5) and $\delta_g$ = 40cm (Sec. 3.6), the leaders form an equilateral triangle with edge length $l$ = 3.72m at the contracted $\lambda_{\min}$ limit.

Fig. 3.3 shows the planned trajectory. The flight begins with the leaders contracting from an initial state to the $\lambda_{\min}$ limit, then continues with the leaders traversing a square with edge-length 1m. The flight concludes with leaders expanding back to the initial configuration; the pilot then commands simultaneous descent and landing. During the flight, the leaders fly to six waypoints. Leaders move from their initial equilateral triangle (pose 1) with edge length $l$ = 4.72m to their contracted configuration (pose 2) with $l$ = 3.72m. Leaders then traverse three of the four sides of the 1m edge-length square through poses 3, 4, and 5. From pose 5, the leaders complete the square by returning to pose 2 and finally expand to pose 1 to end in their initial configuration.

All leaders move in straight lines to reach their next waypoint. The following quintic spline

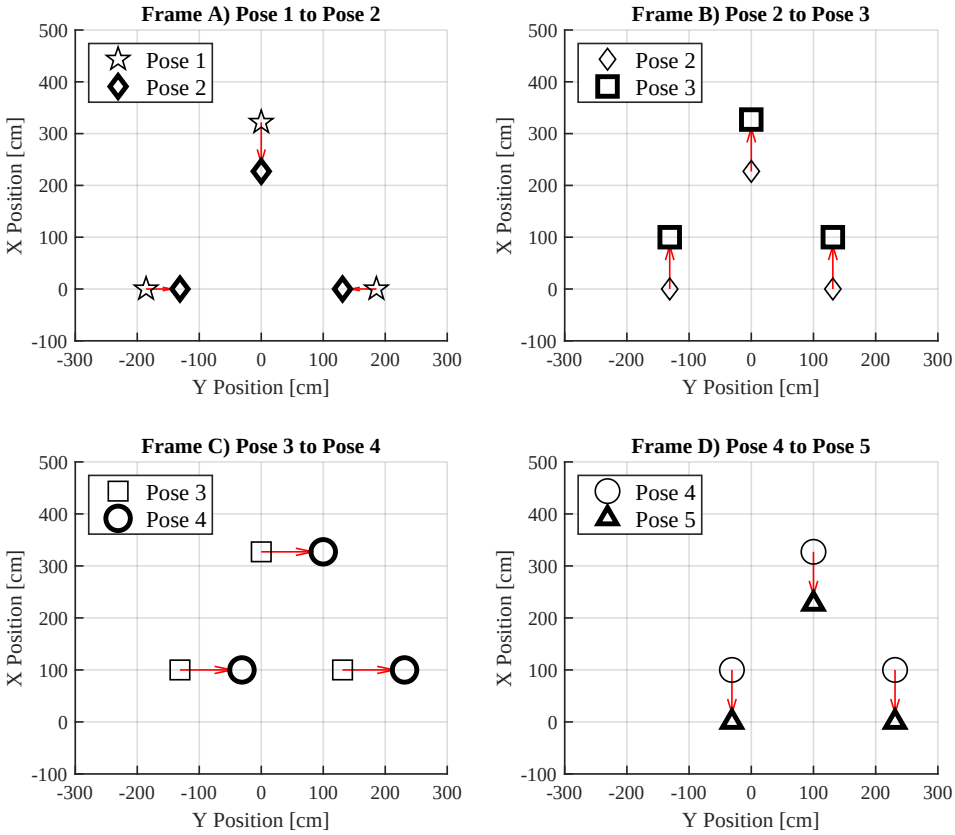



Figure 3.3: Waypoints traversed by leaders during flight. Poses 1-2 show motion to the contracted position. Poses 2-5 show leaders traversing three sides of the 1m edge-length square. From pose 5, leaders return to pose 2 then to pose 1.

guidance law is used for generating the desired trajectory for each leader:

$$
\begin{aligned}
i \in \mathcal{V}_L, \qquad & \mathbf{r}_{d,i}(t) = \mathbf{a}_i + \mathbf{b}_i t + \mathbf{c}_i t^2 + \mathbf{d}_i t^3 + \mathbf{e}_i t^4 + \mathbf{f}_i t^5 \\
& \dot{\mathbf{r}}_{d,i}(t) = \mathbf{b}_i + 2\mathbf{c}_i t + 3\mathbf{d}_i t^2 + 4\mathbf{e}_i t^3 + 5\mathbf{f}_i t^4 \\
& \ddot{\mathbf{r}}_{d,i}(t) = 2\mathbf{c}_i + 6\mathbf{d}_i t + 12\mathbf{e}_i t^2 + 20\mathbf{f}_i t^3,
\end{aligned}
\tag{3.7}
$$

where $\mathbf{r}_{d,i}(t)$, $\dot{\mathbf{r}}_{d,i}(t)$, and $\ddot{\mathbf{r}}_{d,i}(t)$ are the 2D desired position, velocity, and acceleration of the $i^{th}$ agent, respectively. With Eq. 3.7 we enforce six constraints: $\mathbf{r}_{d,i}(t_0) = \mathbf{r}_{0,i}$, $\mathbf{r}_{d,i}(t_f) = \mathbf{r}_{f,i}$, $\dot{\mathbf{r}}_{d,i}(t_0) = \dot{\mathbf{r}}_{d,i}(t_f) = 0$, and $\ddot{\mathbf{r}}_{d,i}(t_0) = \ddot{\mathbf{r}}_{d,i}(t_f) = 0$ to solve for coefficients $\{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}_i, \mathbf{e}_i, \mathbf{f}_i\}$ where $\mathbf{r}_{0,i}$ and $\mathbf{r}_{f,i}$ are the initial and final positions of the $i^{th}$ agent, respectively. From this guidance law, we define $v_{\max}$ as the largest desired velocity of any agent $\dot{\mathbf{r}}_{d,i}(t)$ for all flight times $t$. This desired velocity serves as a feed-forward term in the velocity-tracking portion of the cascaded PID controller (Sec. 3.5). Time of flight $(t_f - t_0)$ between each pair of poses is 3.75s resulting in a

translation of 1m when $v_{\max}$ = 50$\frac{cm}{s}$.

### 3.5 Experimental Setup

Five M330-Quadrotors were used running a modified version of `ArduPilot` (APM) open-source flight controller software (Chapter 2.2). Fig. 3.4 describes the experimental setup. Vehicle pose is provided by an Optitrack motion capture system with eight Prime13 cameras. A BeagleBone Black Ground Control Station receives pose estimates of all vehicles from the motion capture system and uses individual XBee radios to send the pose estimate, desired position (followers only), and a synchronization variable at 60Hz to each quadrotor. A pilot uses a 2.4GHz DSMX Transmitter for manual override in case of system anomalies and a separate computer receives telemetry data from each vehicle over WiFi. All multi-vehicle tests were conducted in M-Air, a 80'$\times$120'$\times$50' outdoor netted facility at the University of Michigan. A wind vane and cup anemometer were used to obtain wind data. Outdoor tests were conducted at night to improve motion capture performance.

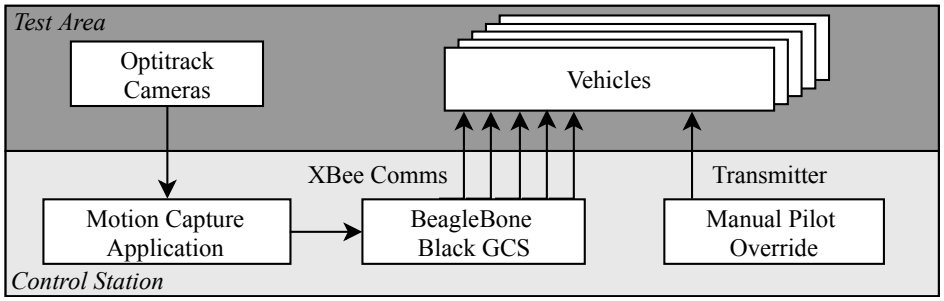



Figure 3.4: Experimental system block diagram

Fig. 3.5 outlines vehicle software. We modified APM's cascaded proportional-integral-derivative (PID) position control to use Optitrack position data. In cascaded PID, position tracking error is scaled by a proportional gain and added to the feed-forward velocity set by the desired velocity $\dot{\mathbf{r}}_d$. Velocity estimates are obtained from a derivative filter on the position estimate (Eq. 3.8) where $T$ is the sampling period [42]. Velocity tracking error becomes a desired acceleration and error from the acceleration tracking loop is fed into an attitude controller that relies on three-axis onboard inertial measurement unit (IMU) data. APM runs the control loop at 400Hz. Leaders use a synchronization variable and a pre-loaded flight path file to generate desired positions and velocities for the position controller. Followers receive a position setpoint via XBee that is the weighted sum of its neighbors' real-time positions set per continuum deformation.

$$\hat{\dot{\mathbf{r}}}(t) = \frac{2\left[\mathbf{r}(t-T) - \mathbf{r}(t-3T)\right] + \left[\mathbf{r}(t) - \mathbf{r}(t-4T)\right]}{8T} \tag{3.8}$$

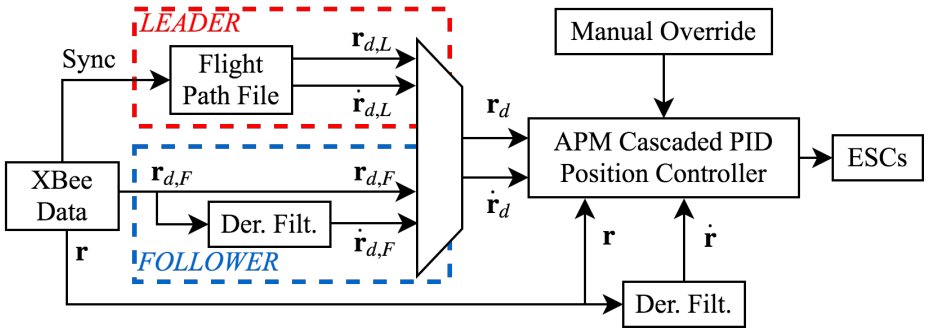



Figure 3.5: Vehicle software block diagram

## 3.6 Experimental Results

We conducted a series of flights with a single vehicle to empirically bound controller error, $\delta_l$ (Sec. 3.6.1). Results were used to set bounding parameter $\delta_g$ as $\delta_l$ in the computation for $\lambda_{\min}$ in Eq. 3.6. Given the set of prescribed leader trajectories, we command the follower agents in two ways. In the first test series, (Sec. 3.6.2) we set the desired position of each follower ($\mathbf{r}_{d,i}$) to its global desired position ($\mathbf{r}_{i,HT}$) $\forall i \in \mathcal{V}_F$ as depicted in Fig. 3.2. In effect, the desired position of the followers is a weighted sum of leaders' desired positions. With this approach, $\delta_l$ and $\delta_g$ are equivalent meaning collision avoidance and containment are guaranteed from Thm. 1. In the second approach (Sec. 3.6.3) we compute desired follower position as a weighted sum of the actual position of three neighbor agents as prescribed in Eq. 3.2 (using the weights in Eq. 3.9). This approach, although more practical for a large and spatially distributed system (due to shorter communication links), no longer guarantees that the controller error bound for each agent $\delta_l$ is a bound on the global deviation of the followers $\|\mathbf{r}_i - \mathbf{r}_{i,HT}\|$ $\forall i \in \mathcal{V}_F$.

$$\begin{bmatrix} w_{4,1} \\ w_{4,3} \\ w_{4,5} \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.134 \\ 0.366 \end{bmatrix}, \begin{bmatrix} w_{5,2} \\ w_{5,3} \\ w_{5,4} \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.134 \\ 0.366 \end{bmatrix} \tag{3.9}$$

### 3.6.1 Determining Controller Error

To determine controller error bound $\delta_l$, we flew a single vehicle in a 1m edge-length square while varying parameters to characterize error sources. Parameters varied include battery voltage and environment (indoor vs. outdoor with negligible wind), largest desired velocity from quintic spline guidance ($v_{\max}$), and altitude above ground level. We also considered the effect of disturbances induced by nearby vehicles by flying all five agents, with followers using global desired positions, to see if controller performance varied from the single-vehicle tests.

Results show that battery voltage, indoor vs. outdoor environment, and altitude above ground

(minimum 1m) have negligible impact on controller error $\delta_l$. Battery voltage was varied from 11.4V to 12.5V, a typical operating range, with less than 10% change in $\delta_l$. We varied flight altitude from 0.75m to 1.75m in increments of 0.25m, with a 15% larger standard deviation in error at 0.75m than at other altitudes. Since all altitudes above 0.75m had similar performance, we flew multi-agent tests at 1.5m above ground level.

To check dependence on $v_{\max}$, a single vehicle was flown using the quintic spline guidance law from Eq. 3.7. We relax the constraint on final desired position $\mathbf{r}_{d,i}(t_f) = \mathbf{r}_{f,i}$ and instead command a particular velocity halfway through each flight segment $\dot{\mathbf{r}}_{d,i}(\frac{t_f - t_0}{2}) = v_{max}$. For this test series, time of flight between each waypoint pair is fixed at $(t_f - t_0)$ = 3.75s which results in a translation of 1m when $v_{\max}$ = 50$\frac{cm}{s}$.

Table 3.1: Statistical parameters of controller error increase as $v_{\max}$ increases. Tests performed indoors at 150cm altitude.

| $v_{\max}$ (cm/s) | Mean (cm) | Std. Dev (cm) | Max (cm) |
|---|---|---|---|
| 0 | 4.80 | 1.90 | 12.44 |
| 25 | 6.64 | 3.04 | 15.95 |
| 50 | 7.94 | 4.27 | 22.65 |
| 75 | 11.64 | 6.65 | 29.77 |
| 100 | 14.53 | 9.23 | 36.44 |

Table 3.1 shows that tracking error mean, max, and standard deviation increase as $v_{\max}$ increases. We believe a large source of error is introduced in state estimate delay. Each agent receives a position estimate from Optitrack with about a 40ms delay. Then, a derivative filter (Eq. 3.8) is used on position data to estimate the actual velocity. The filter adds an additional delay to the velocity estimate, but characterization of the derivatives filter's delay and its effects on position error was not explored for this chapter.

For data shown in Table 3.1, the $v_{\max} = 0\frac{cm}{s}$ hover case is computed over 40 flights. The other datapoints, $v_{\max} = \{25, 50, 75, 100\}\frac{cm}{s}$ in Table 3.1, have ten flights for each $v_{\max}$. All flights in this dataset were conducted indoors at an altitude of 150 cm above the floor.

To investigate downwash impact on neighboring agents, we flew all five quadrotors with followers using global desired positions and compared results to those in Table 3.1. Leaders flew the trajectory from Sec. 3.4 with $\delta_g$ = 40cm, $(t_f - t_0)$ = 3.75s and $v_{max}$ = 50$\frac{cm}{s}$ as before. Two full flights with five vehicles were flown resulting in ten datasets overall.

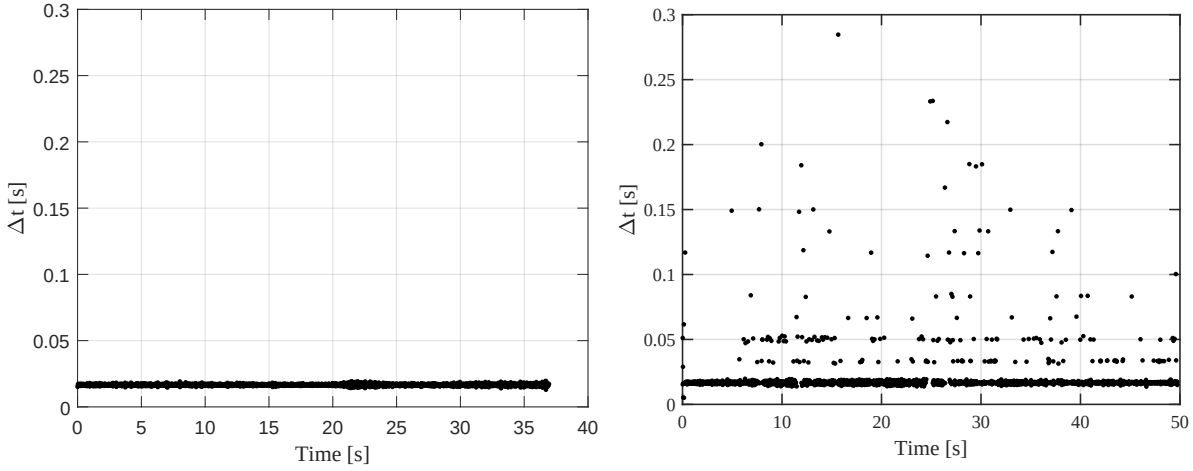
Table 3.2 shows the results of the five-agent flights alongside the single-agent flight data copied from the 50$\frac{cm}{s}$ case in Table 3.1. For the five-agent flights, agent 4 had an anomaly where it did not run any controller-related computations for 0.3s resulting a maximum error of 41.68cm. which is treated as an outlier and excluded from error characterization.

During the five-agent flights, the XBees occasionally drop packets. Fig. 3.6 shows the change in

Table 3.2: Statistical parameters of inter-agent disturbance characterization. Tests performed outdoors at 150cm altitude

| | Mean (cm) | Std. Dev (cm) | Max (cm) |
|---|---|---|---|
| Five-Agent Flights | 9.69 | 5.16 | 30.33 |
| Single-Agent Flights | 7.94 | 4.27 | 22.65 |

time $\Delta t$ between two consecutive Optitrack pose estimates received by an agent over Xbee. Fig. 3.6a shows a typical single-agent flight while Fig. 3.6b is from a five-agent flight where the larger $\Delta t$ corresponds to an integer number of packets lost. Five-agent flights may have larger error in part because of Optitrack data packet drop. Results from Tables 3.1 and 3.2 show that an error bound of $\delta_l$ = 40cm is sufficient. The first five-agent flights sent followers global desired positions to support analysis of continuum deformation constraints in Sec. 3.6.2.




(**a**) Single-agent flight.

(**b**) Five-agent flight.

Figure 3.6: Data is received consistently over Xbee radios at 60Hz for single-agent flights. Packets dropped in five-agent flights.

### 3.6.2 Five-Agent Flight: Followers with Pre-set Waypoints

Fig. 3.7 shows the results for a five-agent test with follower positions based on a weighted sum of the leader desired positions. The wind speed was measured to be 0mph. $\delta_g$ = 40cm was used as a conservative error bound. This test was run to distinguish experimental setup error sources (downwash, communication interference, etc) from coordination-induced error. The four plots in Fig. 3.7 show team performance with respect to the four constraints from Sec. 3.4: follower containment within the leading triangle, collision avoidance between agents, bounding of deviation from the local desired position, and bounding of deviation from the global desired position. The

black vertical lines on each plot use the same symbols from Fig. 3.3 to show the desired waypoint of the leaders in time. The unlabeled vertical lines between Star ($\star$) and Diamond ($\Diamond$) represent an intermediate waypoint added to fit all agents within our motion capture workspace.

Fig. 3.7A shows the distance from both followers to the leader boundary with a flat horizontal line at $\epsilon$ = 28cm and another horizontal line at $-\delta_g$ = -40cm. Crossing the $\epsilon$ line means the follower left the leader boundary while crossing the $-\delta_g$ line means the follower left the outer bounding triangle. Thus, the followers never leave the leading triangle since their lines never go below the horizontal $\epsilon$ line. Fig. 3.7B shows each agent's distance to its nearest neighbor along with a $2\epsilon$ = 56cm line. There are no inter-agent collisions because no agent has a distance that goes below the $2\epsilon$ line. Fig. 3.7C and Fig. 3.7D show the deviation of each agent from its local and global desired position. No agent has a local or global deviation exceeding $\delta_g$ = 40cm (represented by the horizontal line). Figs. 3.7C and 3.7D are identical since followers are being commanded to global desired positions.

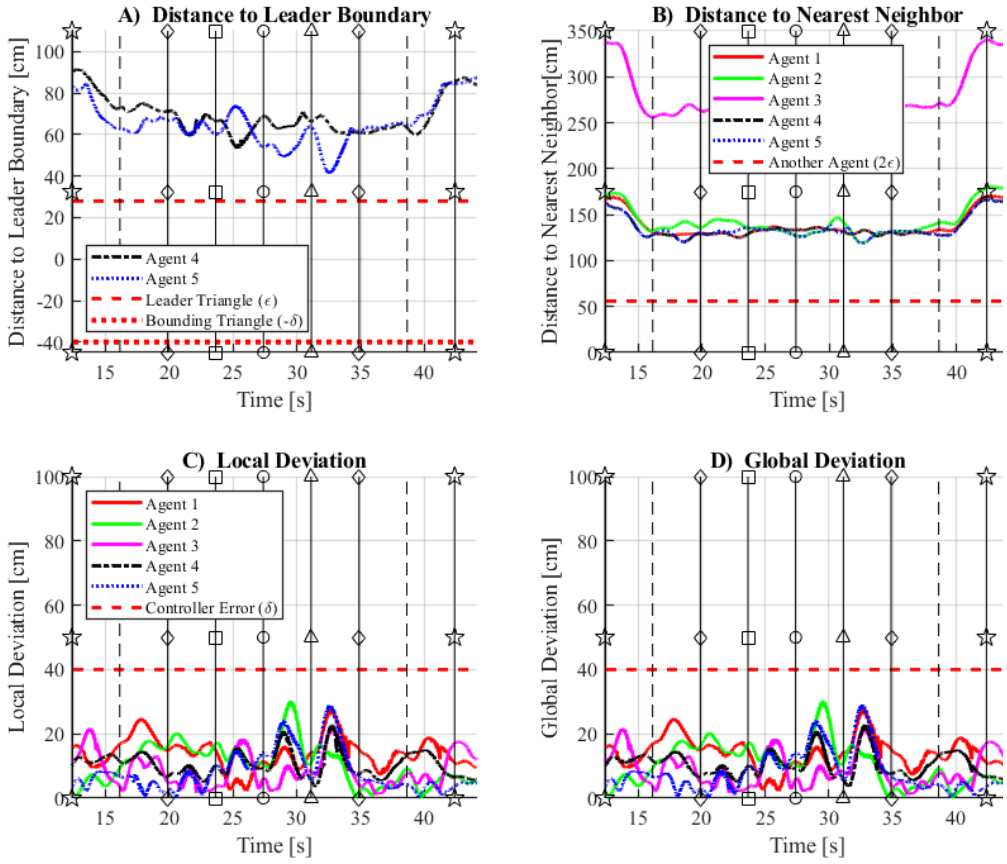



Figure 3.7: In-flight constraint values over time for the five-quadrotor continuum deformation trajectory. Follower desired positions are computed from desired leader positions.

### 3.6.3 Five-Agent Flight: Followers with Local Communication

Fig. 3.8 shows results for the five quadrotor system running continuum deformation under local communication. The average wind speed was 2.1mph at 75° relative to +X (effectively the direction of motion in Fig. 3.3C: □ to ◯).

Figs. 3.8B and Figs. 3.8C show there are no inter-agent collisions and that no agent exceeds the local deviation bound. Fig. 3.8D shows that followers violate the global deviation constraint in all segments except when the formation moves in the direction of the wind:(□ to ◯) and (◊ to the intermediate waypoint). In segments where the global deviation constraint is satisfied, neither follower leaves the leader triangle as predicted by Thm. 1. However, there is a segment (intermediate waypoint to ◊) where followers violate global deviation while within the leader boundary.

The large follower deviation in Fig. 3.8D is caused in part by steady-state error in agent y-components, most likely due to wind which effectively shifted the entire formation in +Y. Follower local desired positions ($\mathbf{r}_{d,i}$) shift with the leaders while their global desired positions ($\mathbf{r}_{i,HT}$) are unaffected, resulting in a compounding error effect with respect to the follower global desired position reference.

## 3.7 Global Deviation Bound from Local

Continuum deformation guarantees collision avoidance and containment when a global deviation bound used for trajectory generation is met. In tests where the local deviation bound is met and also equal to the global deviation bound (Fig. 3.7), we observed the system meeting all four constraints.

However, under strictly local communication of actual positions, the actual global deviation bound is unknown and constraints cannot be guaranteed using a local deviation bound (Fig. 3.8). The collision and local deviation constraints were met while containment and global deviation were violated. Importantly, all constraints were satisfied in segments of the flight where the global deviation constraint was met (Thm. 1).

A very large global deviation bound can be prescribed to satisfy all constraints, but such a conservative approach would require the quadrotors to be substantially separated which may not be practical.

This section derives the actual global deviation bound for each agent as a function of the communication weights and individual local deviation bounds. Definitions from Sec. 3.3 are used along with additional notation when necessary.

Let

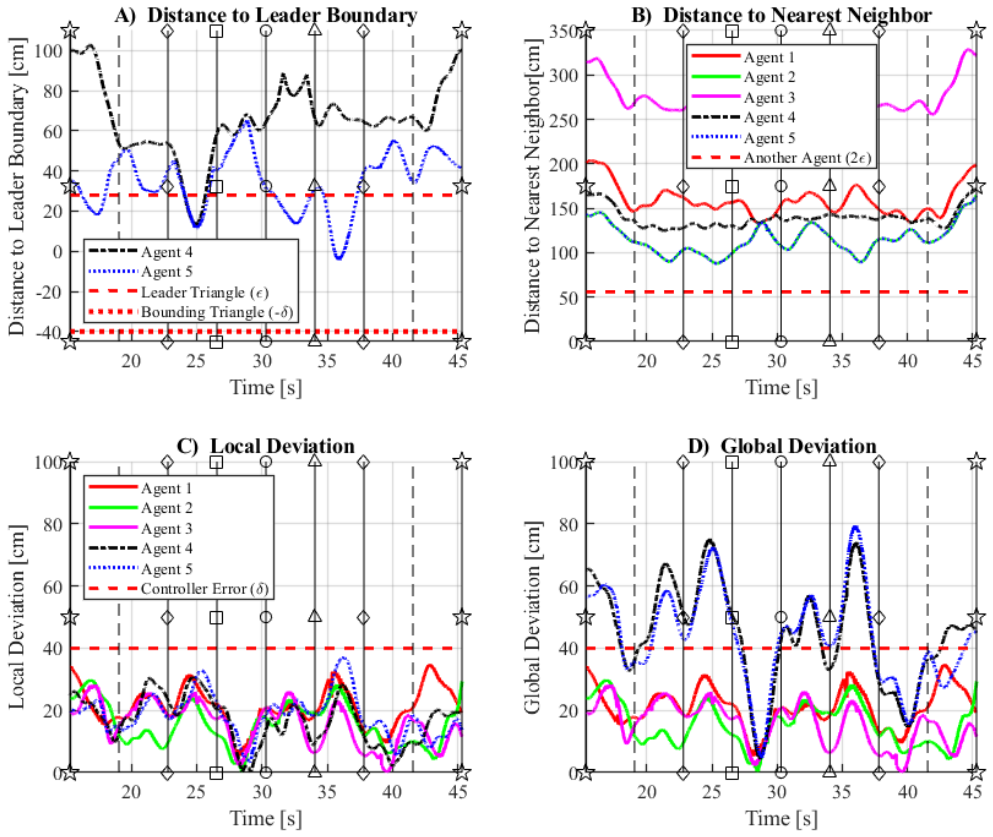



Figure 3.8: In-flight constraint values over time for the five-quadrotor continuum deformation trajectory. Follower desired positions are computed from local communication of actual neighbor positions.

$$\mathbf{P} = \begin{bmatrix} \mathbf{r}_1^T \\ \vdots \\ \mathbf{r}_n^T \end{bmatrix} \in \mathbb{R}^{nx2}, \mathbf{P}_d = \begin{bmatrix} \mathbf{r}_{d,1}^T \\ \vdots \\ \mathbf{r}_{d,n}^T \end{bmatrix} \in \mathbb{R}^{nx2}$$

$$\mathbf{P}_{A,HT} = \begin{bmatrix} \mathbf{r}_{1,HT}^T \\ \vdots \\ \mathbf{r}_{n,HT}^T \end{bmatrix} \in \mathbb{R}^{nx2}, \mathbf{P}_{L,HT} = \begin{bmatrix} \mathbf{r}_{1,HT}^T \\ \mathbf{r}_{2,HT}^T \\ \mathbf{r}_{3,HT}^T \\ \mathbf{0}_{(n-3)\times 2} \end{bmatrix} \in \mathbb{R}^{nx2}$$

These are matrices of stacked position vectors (actual, desired, global desired, and leader global desired respectively). Note, these are different from $\mathbf{P}_0, \mathbf{P}_{HT} \in \mathbb{R}^{3\times 2}$ (Sec. 3.3) which will not be referenced.

Let $\mathbf{D}_{loc} = \mathbf{P} - \mathbf{P}_d$ and $\mathbf{D}_g = \mathbf{P} - \mathbf{P}_{A,HT}$ be the local and global position offsets, respectively.

Define a matrix of the communication weights (Eq. 3.1) as

$$\mathbf{W} = \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \dots & \mathbf{0}_{3\times1} \\ w_{4,1} & w_{4,2} & \dots & w_{4,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \dots & w_{n,n} \end{bmatrix} \tag{3.10}$$

Define "Global communication weights" $\alpha_{i,j}$ such that

$$\mathbf{r}_{i,HT} = \sum_{j=1}^{3} \alpha_{i,j} \mathbf{r}_{j,0} \tag{3.11}$$

$$\begin{bmatrix} \alpha_{i,1} \\ \alpha_{i,2} \\ \alpha_{i,3} \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ y_{1,0} & y_{2,0} & y_{3,0} \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_{i,0} \\ y_{i,0} \\ 1 \end{bmatrix} \tag{3.12}$$

Combining these weights into a single matrix, also define

$$\boldsymbol{\alpha} = \begin{bmatrix} & \mathbf{I}_3 & & \mathbf{0}_{3\times(n-3)} \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \mathbf{0}_{1\times(n-3)} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{n,1} & \alpha_{n,2} & \alpha_{n,3} & \mathbf{0}_{1\times(n-3)} \end{bmatrix} \tag{3.13}$$

Let $\|\mathbf{r}_{d,i} - \mathbf{r}_i\| \leq \delta_{i,l}$ and $\delta_l = \max_{1\leq i\leq n} \delta_{i,l}$ be the individual local deviation bounds and the local deviation bound, respectively. Let $\|\mathbf{r}_{i,HT} - \mathbf{r}_i\| \leq \delta_{g,i}$ and $\delta_g = \max_{1\leq i\leq n} \delta_{g,i}$ be the individual global deviation bounds and the global deviation bound, respectively.

Define $\mathbf{d}_l = \begin{bmatrix} \delta_{1,l} & \dots & \delta_{n,l} \end{bmatrix}^T$ and $\mathbf{d}_g = \begin{bmatrix} \delta_{g,1} & \dots & \delta_{g,n} \end{bmatrix}^T$. Next define a relationship between individual local deviation bounds ($\mathbf{d}_l$) and individual global deviation bounds ($\mathbf{d}_g$).

**Theorem 2.**

$$\mathbf{D}_g = (\mathbf{I} - \mathbf{W})^{-1} \mathbf{D}_{loc} \tag{3.14}$$

*Proof.* We can express the desired agent positions as

$$\mathbf{P}_d = \mathbf{P}_{L,HT} + \mathbf{W}\mathbf{P} \tag{3.15}$$

We can also write the position of each agent as

$$\mathbf{P} = \mathbf{P}_d + \mathbf{D}_{loc} \tag{3.16}$$

Solving for $\mathbf{P}$ yields

$$\begin{aligned}\mathbf{P} &= \mathbf{P}_{L,HT} + \mathbf{W}\mathbf{P} + \mathbf{D}_{loc} \\ (\mathbf{I} - \mathbf{W})\,\mathbf{P} &= \mathbf{P}_{L,HT} + \mathbf{D}_{loc} \\ \mathbf{P} &= (\mathbf{I} - \mathbf{W})^{-1}\,(\mathbf{P}_{L,HT} + \mathbf{D}_{loc})\end{aligned}$$

The global position offset can then be written as

$$\begin{aligned}\mathbf{D}_g &= \mathbf{P} - \mathbf{P}_{A,HT} \\ &= (\mathbf{I} - \mathbf{W})^{-1}\,(\mathbf{P}_{L,HT} + \mathbf{D}_{loc}) - \boldsymbol{\alpha}\mathbf{P}_{L,HT} \\ &= (\mathbf{I} - \mathbf{W})^{-1}\,\mathbf{D}_{loc} + \left[(\mathbf{I} - \mathbf{W})^{-1} - \boldsymbol{\alpha}\right]\mathbf{P}_{L,HT}\end{aligned}$$

The second term in the above equation is zero as shown in detail in Appendix B. Thus,

$$\mathbf{D}_g = (\mathbf{I} - \mathbf{W})^{-1}\,\mathbf{D}_{loc}$$

□

**Theorem 3.**

$$\mathbf{d}_g = (\mathbf{I} - \mathbf{W})^{-1}\,\mathbf{d}_l \tag{3.17}$$

*Proof.* Because $(\mathbf{I} - \mathbf{W})$ is a non-singular M-matrix, all entries of its matrix inverse are greater than or equal to 0. Note that $\mathbf{D}$ derived in Appendix B is inverse positive [27]. Therefore, $(\mathbf{I} - \mathbf{W})$ is inverse-positive and an M-matrix.

Let $\mathbf{K} = (\mathbf{I} - \mathbf{W})^{-1}$ and define

$$\mathbf{D}_{loc} = \begin{bmatrix} \eta_1 \cos\beta_1 & \eta_1 \sin\beta_1 \\ \vdots & \vdots \\ \eta_n \cos\beta_n & \eta_n \sin\beta_n \end{bmatrix},\ 0 \le \eta_i \le \delta_{i,l},\ 0 \le \beta_i \le 2\pi$$

as a general local deviation that satisfies the local deviation bound. Then,

$$\begin{aligned}&\|\mathbf{r}_{i,HT} - \mathbf{r}_i\| \\ &= \sqrt{\left[\sum_{m=1}^{n} \mathbf{K}_{i,m}\eta_m \cos\beta_m\right]^2 + \left[\sum_{m=1}^{n} \mathbf{K}_{i,m}\eta_m \sin\beta_m\right]^2}\end{aligned}$$

This expression is maximized when $\beta_i = \beta_j \forall i, j$ and $\eta_i = \delta_{i,c} \forall i$ as shown in Appendix B.1, yielding

$$\mathbf{d}_g = (\mathbf{I} - \mathbf{W})^{-1} \mathbf{d}_l$$

□

Eq. 3.17 and communication weights from Eq. 3.9 yields

$$\delta_g = 2.5773 \, \delta_l \tag{3.18}$$

Using Eq. 3.18 and our experimentally determined local deviation bound of 40 cm results in a global deviation bound of 103.092 cm. The methods from Sec. 3.4 can then be repeated with this new $\delta_g$ to actually design a safe trajectory under continuum deformation as opposed to using the local deviation bound which does not.

## 3.8 Conclusion

This research conducted the first experimental evaluation of continuum deformation. Local deviation bounds were obtained experimentally and used to plan $\lambda_{min}$ trajectories to test the theory. Under "global desired" guidance, continuum deformation constraints were satisfied and safety guarantees were provided. Under "local communication" guidance, constraints were violated for most of the flight. To address this issue, $\delta_g$ bounds were updated theoretically given a local controller error. Therefore, a trajectory can now be planned using local controller errors for each agent that guarantees collision avoidance and containment.

In the future, this result will allow inter-agent collision avoidance and containment to be guaranteed by satisfying a local deviation bound alone. We plan to conduct further experiments that both validate this new global deviation bound as well as directly compare continuum deformation to other cooperative control techniques.

# CHAPTER 4

# Cooperative Payload Transportation with Haptic Guidance

## 4.1 Introduction

Payloads that do not fit conveniently within an aircraft or rotorcraft cargo bay can be transported short distances as a tethered slung load. Single vehicle slung load transportation is well studied, especially in military applications with large transport helicopters [5]. However, as the payload weight increases so must the carrying capacity of the single vehicle. Alternatively, instead of using one large vehicle, many comparatively smaller vehicles can be used to carry the same heavy payload. Cooperative slung load transportation is focused on this problem; carrying a single load by a group of air vehicles all tethered to it [43, 44, 11, 45, 46, 47, 10, 48, 49]. It offers the advantage of redundancy: if a single vehicle fails the system level mission might continue. Dynamics modes that are problematic with a single tether such as spinning about the tether and pendulum swinging can be more easily damped out with multiple vehicles pulling in multiple directions. As the payload weight scales up (multi-ton and higher) each of these factors become even more important. Total system cost (development and production) of a single utility vehicle that can be reconfigured for multi-lift missions when needed will likely be much cheaper than a single vehicle designed specifically for massive loads that is infrequently used. However, this comes at the cost of system complexity and the need to avoid collisions between vehicles, avoid tether entanglement, and consistently cooperate as a team to maintain control system efficiency and stability.

A typical slung load system would be guided by directly piloting the vehicle(s) or following pre-planned trajectories. However, when a payload needs to be guided through an unknown cluttered, dynamic environment neither existing option might be effective. Consider a disaster relief mission where a cooperative slung load system needs to transport supplies into a truck bed that is beyond visual line of sight. A remote pilot might use a first-person camera mounted on the payload to guide it into the truck bed, but the operator would then not maintain situational awareness of the quadrotors or tethers. Further, pilot projection into the first-person payload viewpoint will reduce the pilot's awareness of their own surroundings. As an alternative, suppose the slung payload was cooperatively flown to a safe location near the truck bed. On approach, suppose there was a

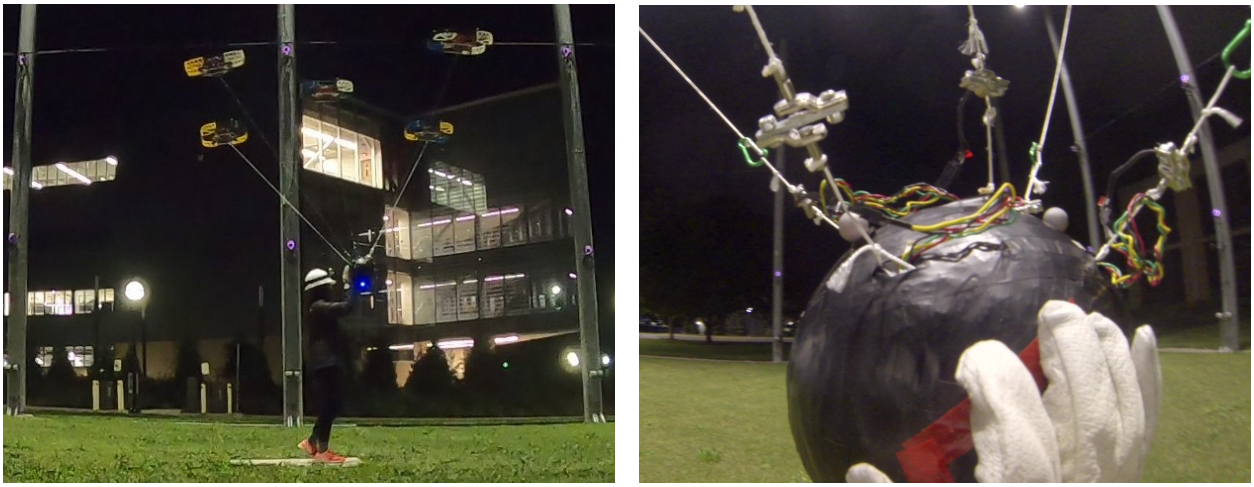
person ready to receive the payload by physically guiding it into place. This end user might not be trained as a remote pilot and so the interface would need to be simple and intuitive. This chapter proposes a haptic feedback system enabling such an end user to apply force onto the slung payload to accurately guide it in the direction they want it to move. This type of system isn't limited to this disaster relief scenario and has wide reaching applications. For military applications, rapid construction in hostile environments is a possible use of the system as a versatile crane where a ground operator could precisely position heavy materials via the haptic guidance. Likewise for civilian applications, material or equipment delivery onto a rooftop could be precisely placed by a worker using this system.

This chapter describes a haptically guided slung load cooperative transportation system (Figure 4.1). A centralized controller runs on an instrumented payload equipped with force sensors. Real-time estimation of user applied force is performed on the payload. This force estimate is input into an admittance controller. This type of controller is commonly used for human-robot interaction and uses virtual dynamics to allow for external force inputs into a system to modify a reference trajectory. This reference trajectory is then tracked by a payload controller which commands individual quadrotor reference states in real time. Individual quadrotor controllers track this trajectory, which in turn causes the tethers to move the payload in the direction of the user applied force or hover when no force is applied.

Ref. [12] first conceived of the haptic guidance concept. Simulation results however relied on simplifying assumptions for user interaction, sensors, and state estimation. This chapter realizes and evaluates this system in an experimental platform equipped with sensors, state estimation, and control laws necessary to accurately measure user applied forces and accurately translate them into the desired payload movements. Particular elements from Ref. [12] that were used in this chapter include the haptic guidance concept as a whole as well as a starting point for the force estimation and guidance modules. Force estimation was improved by considering real sensors, real-time filtering, and including an offset term and guidance was improved from an ad-hoc set of discrete waypoints traversed at a constant velocity to a general admittance controller formulation with inclusion of an activation force.

### 4.1.1 Related Work

The multi-vehicle slung load problem is well studied [43, 44, 11, 45, 46, 47, 10, 48, 49, 12]. Cooperative control among the vehicles can be centralized or distributed [50]. Centralized methods have a single agent that computes and communicates guidance and control commands for all vehicles [51, 52, 10]. Distributed methods are designed such that each agent can compute its own guidance and control solution with minimal inter-agent communication [53, 54]. The multi-vehicle slung load problem is typically managed with centralized control [10, 49, 55, 56]. For example,



(**a**) Third person view (**b**) First person view

Figure 4.1: Haptically guided slung load system. The payload travels in the user applied force direction.

Ref. [10] applies a centralized, hierarchical approach that optimizes cable angles and validates controllability with flight testing. However, Ref. [57] achieves communication-less distributed control via a leader-follower system with admittance controllers. In this chapter, a centralized control scheme was chosen. This is a practical choice since for the application a fully connected network can be assumed and safety for the user is a top priority (e.g. distributed communication delays could introduce system instability and be dangerous). Various models and control schemes have been studied for the multi-vehicle slung load problem in theory and simulation [44, 11, 45]. In Ref. [11] a geometric controller was constructed to track a rigid body payload trajectory and was validated in simulation. Flight test validation has also been performed [10, 49, 56, 55]. For example, Ref. [49] focuses on a military application with a dual-lift helicopter system. A 6DOF force/torque sensor measured tether tension and was used in the controller to adjust the height of the secondary vehicle to balance the 40 lb load between them. In this work, flight experiments are also used as a means to validate the haptic guidance system methodology proposed herein. Tether force feedback is also used but is distinct in its determination of user guidance inputs. An important element of the haptic guidance system is the motion tracking control of which there are many methods available. Each of these methods provide different benefits in terms of motion tracking of a desired payload trajectory with stability and conditions of stability being the most important factor. Ref. [11] shows asymptotic stability with a geometric nonlinear PID controller in a global formulation in theory and simulation. Ref. [10] demonstrated stability and disturbance rejection through flight testing of their hierarchical approach. Ref. [58] used a model predictive controller and also validates its performance via flight test results. In this work, the motion tracking controller was not the focus, but was used to support the haptic guidance scheme. A hierarchical approach (inspired by Ref.

[10]) was used. The 3D position tracking PI controller was validated to be stable for the ranges of states and inputs considered in the simulation and experimental results. Guidance for this complex system is typically pre-planned or transmitter-based [10, 11, 55]. However, Ref. [12] introduced the haptic guidance concept in theory and simulation. In this work, we also employ the haptic guidance concept but add several new elements (described later) to make it into a realizable form.

Haptic guidance can be achieved via admittance control. This allows for applied forces to pass through virtual dynamics which then prescribes a reference velocity command with magnitude and direction based on measured force application. Admittance control is well studied and has applications in industrial robot arms, exoskeletons, and multirotors [59, 60, 61, 62, 63, 64, 65]. For example, in Ref. [65], an admittance controller ran on a quadrotor and a user applied force to modify its trajectory. Ref. [57] used admittance control in a multi-vehicle slung load application where all vehicles are tethered directly above their mounting points. In hover, they had zero roll and pitch and could thus apply a thrust upwards to carry their portion of the load, with no 2D (lateral) external forces experienced. During transportation, a leader vehicle was directly piloted by a user to move in a desired 2D direction, which also pulled on the tether in this direction. The tether force pulled the payload and consequently pulled on the follower vehicles in the same direction. Follower vehicles ran admittance controllers which updated their 2D guidance based on the estimated force experienced, therefore they moved in the direction they were pulled and allowed the leader to guide the system without explicit communication. This was experimentally demonstrated on wide payloads that allowed every vehicle to be directly above their mounting point. However, this method would not work if the vehicles were not directly over their mounting points because they would experience horizontal tether forces during hover causing the vehicles to move over their mounting points away from the desired formation. The scope of vehicle configurations and payload dimensions are severely limited because of this. In this chapter, an admittance controller is developed for the payload itself rather than for one or more multirotors. This offers the advantage of arbitrary placements of the vehicles relative to their mounting points on the payload and allows them to have non-zero feedforward roll and pitch values.

Force estimation of the input applied force is required for admittance control to function. Two main approaches exist for estimating the total applied external force when it cannot be directly measured. The first approach applies a low pass filter on sensor data to compute the force of interest [12, 66]. Ref. [12] used an averaging filter on the tether forces before adding the remaining forces in the multi-vehicle slung load application. The drawback of this first approach is the need for sensors to directly measure all forces except the one being estimated. The second approach assumes that direct force measurements of external forces are not available and instead relies on an inertial sensor and more complex filtering techniques [67, 68, 69]. For example, Ref. [57] used an augmented state extended Kalman filter to estimate external forces and torques on follower vehicles. The drawback

of this approach is the requirement of accurate models of the system and well tuned observers to achieve good results. In this chapter, the first approach is used because it is expected the payload mass and drag properties will differ between flights, and also the multirotor vehicles will not be accurately modeled due to their nonlinear dynamics.

### 4.1.2 Contributions and Outline

This work offers four contributions. First, it develops and validates the first user haptic guidance system applying admittance control for cooperative airborne payload transport. Second, it develops and validates the first user applied force estimation capability for cooperative airborne payload transportation. The system estimates real-time applied force on the payload from tether tension force. Third, it presents the first experimental validation of an integrated cooperative airborne payload system haptically guided by a user. Fourth, it analyzes the viability of a haptically guided cooperative airborne payload transportation system.

Section 4.2 defines the haptically guided slung load cooperative transportation problem that the proposed system is solving. Section 4.3 presents the system dynamics model, and the control strategy is described in Section 4.4. User force estimation (Section 4.5) and payload guidance (Section 4.6) approaches are described. Experimental test set-up, simulation results, and experimental results are presented in Sections 4.7, 4.8, and 4.9, respectively, followed by conclusions in Section 4.10.

## 4.2 Problem Statement

An $N_V$ agent multirotor team is attached to a common payload by $N_V$ individual tethers, one per vehicle. The team must lift and carry the payload along a path that follows user applied force commands over time. User haptic guidance with 3D translation without rotation is supported, however, the payload is carried at constant shoulder level above ground (2D translation) to make applying force onto the payload as simple as possible. Accurate, real-time 6DOF pose updates are available to each of the multirotors as well as the payload through motion capture. The payload hosts an embedded computer connected to tether inline tension sensors to provide force estimates that augment motion capture data. All vehicle and payload computers are connected to a low-latency network enabling data to be shared in real-time across the team.

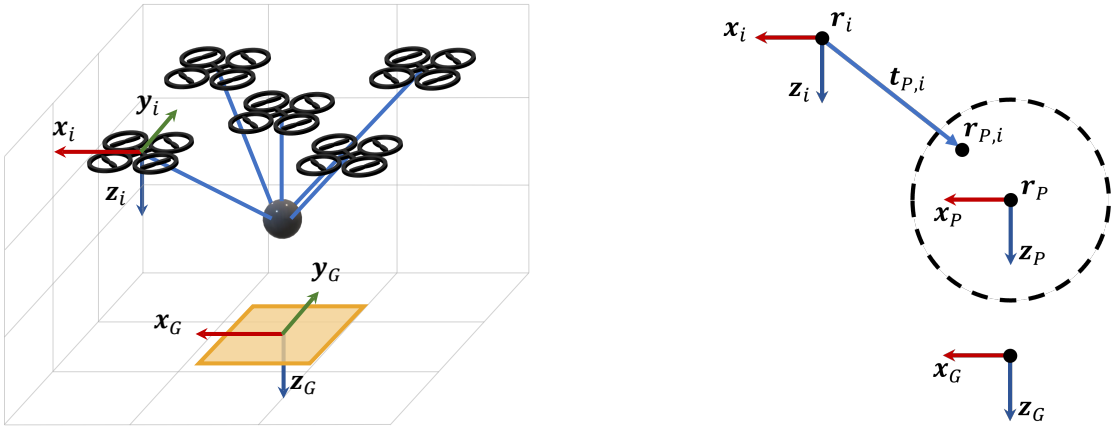
One multirotor is placed directly above the payload while the $N_V - 1$ remaining multirotors are spread equally around a virtual circle of given radius in a fixed formation. We define a virtual inverted cone with the slung payload at the vertex and multirotors positioned above on the cone's base. The angle between the cone's central axis and the multirotor circle is $45^\circ$ for this work, offering a balance between stability due to applied side forces and payload weight carrying capacity [10]. The placement of the overhead multirotor is atypical, but provides a powerful benefit. The

overhead multirotor carries weight more efficiently than the others because its tether force vector directly counters gravity, while each other multirotor applies a tether force to the payload at an angle. The multirotor team is homogeneous and the system must be controlled so that there is approximately equal tension along each tether during operation to balance the overall load.

This chapter focuses on design, implementation, and evaluation of the multi-vehicle payload control system enabling the multirotor team to stabilize and carry the payload in the direction of user applied force. Given raw sensor data, an estimate of user applied force must be computed. Next, desired payload state (position and velocity) must be updated based on the estimated user applied force input. Payload desired state must then be transformed to a desired state or setpoint for each multirotor such that the vehicles apply the correct tether force to move the payload and stabilize it per user inputs. System performance is evaluated with respect to stability and payload trajectory tracking.

## 4.3 Dynamics Model

Figure 4.2 shows the cooperative payload transportation system with $N_V = 5$ vehicles, a spherical payload, and $N_V = 5$ tethers that connect quadrotor centroids to mounting points on the payload surface. Define a North-East-Down (NED) ground reference coordinate frame $G$ with a local origin, a body coordinate frame for each vehicle $i \in \{1, ..., N_V\}$, and a payload coordinate frame $P$.




(**a**) 3D System Overview

(**b**) 2D Tether and Payload Diagram

Figure 4.2: Cooperative payload transportation system

### 4.3.1 Vehicle Model

Consider the quadrotor model from chapter 2.2.2. Define $N_V$ mounting points on the payload as $\mathbf{r}_{P,i}, (i \in \{1, ..., N_V\})$. Each vehicle $i$ is tethered from its center of mass ($\mathbf{r}_i$) to payload mounting point $i$ given by $\mathbf{r}_{P,i}$. Gravity, aerodynamic drag, tether force, and total propeller thrust force act on each vehicle $i$ as given by

$$\mathbf{F}^i_{net,i} = \mathbf{F}^i_{Gravity,i} + \mathbf{F}^i_{Drag,i} - \mathbf{F}^i_{Tether,i} + \mathbf{F}^i_{Thrust,i}. \tag{4.1}$$

### 4.3.2 Payload Model

A payload of mass $m_P$ and diameter $d$ is attached via tethers between each vehicle $i$ center of mass ($\mathbf{r}_i$) and their respective mounting points on the payload ($\mathbf{r}_{P,i}$). This work assumes tethers are massless and do not disturb the flow around them. The tether for each vehicle $i$ has unstretched length $L_i$ and is modelled as a spring-damper with spring constant $k_i$ and damping constant $b_i$. Define payload position in the ground frame as $\mathbf{r}^G_P$, the tether vector $\mathbf{t}^G_{P,i} = \mathbf{r}^G_{P,i} - \mathbf{r}^G_i$ with direction $\hat{\mathbf{t}}^G_{P,i} = \frac{\mathbf{t}^G_{P,i}}{\left\|\mathbf{t}^G_{P,i}\right\|}$, and tether length $l_i = \left\|\mathbf{t}^G_{P,i}\right\|$. The tether length rate of change is then $\dot{l}_i = \dot{\mathbf{t}}^G_{P,i} \cdot \hat{\mathbf{t}}^G_{P,i}$, where $\dot{\mathbf{t}}^G_{P,i} = \begin{bmatrix} p_P \\ q_P \\ r_P \end{bmatrix} \times \left(R^G_P \mathbf{r}^P_{P,i}\right) + \dot{\mathbf{r}}^G_P - \dot{\mathbf{r}}^G_i$. Gravity, aerodynamic drag, and tether forces act on the payload as

$$\mathbf{F}^P_{net,P} = \mathbf{F}^P_{Gravity,P} + \mathbf{F}^P_{Drag,P} + \sum_{i=1}^{n} \mathbf{F}^P_{Tether,i} + \mathbf{F}^P_{User}. \tag{4.2}$$

Tether forces induce a net moment

$$\boldsymbol{\tau}^P_{net,P} = \sum_{i=1}^{n} \mathbf{r}^P_{P,i} \times \mathbf{F}^P_{Tether,i}. \tag{4.3}$$

A spring-damper model governs each ground frame tether force per

$$\mathbf{F}^G_{Tether,i} = \left(k_i \left(l_i - L_i\right) + b_i \dot{l}_i\right) \left(-\hat{\mathbf{t}}^G_{P,i}\right). \tag{4.4}$$

Tether and applied gravity forces in the payload frame are thus

$$\mathbf{F}^P_{Tether,i} = R^P_G \mathbf{F}^G_{Tether,i}, \ \mathbf{F}^i_{Tether,i} = R^i_G \mathbf{F}^G_{Tether,i}, \ \mathbf{F}^P_{Gravity,P} = R^P_G m_P g \mathbf{e}_3. \tag{4.5}$$

where $m_P$ is payload mass, $R^P_G$ is the rotation matrix from frame $G$ to payload frame $P$, and $\mathbf{e}_3 = [0; 0; 1]$. The payload frame defines velocity vector ($u_P$, $v_P$, $w_P$), attitude vector ($\phi_P$, $\theta_P$, $\psi_P$),

and angular rate vector ($p_P$, $q_P$, $r_P$). Payload moments of inertia are defined as $J_{x,P}$, $J_{y,P}$, $J_{z,P}$.

## 4.4 Control

Figure 4.3 shows the controller block diagram. Each component will be discussed in the following sections.

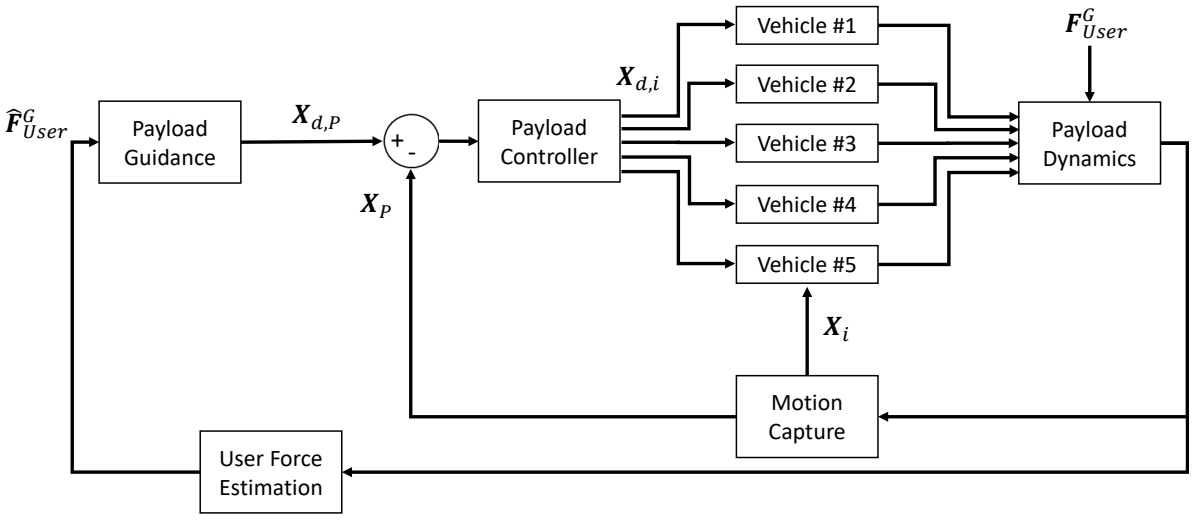



Figure 4.3: Controller block diagram

### 4.4.1 Payload Controller

Actual payload state is defined as $\mathbf{X}_P = \begin{bmatrix} \mathbf{r}_P^T & \dot{\mathbf{r}}_P^T \end{bmatrix}^T$, desired payload state as $\mathbf{X}_{d,P} = \begin{bmatrix} \mathbf{r}_{d,P}^T & \dot{\mathbf{r}}_{d,P}^T \end{bmatrix}^T$, and payload error state as $\mathbf{e}_{load} = \mathbf{X}_{d,P}^G - \mathbf{X}_P^G$. The goal of payload control is to track the desired payload state by commanding desired vehicle states $\mathbf{X}_{d,i} = \begin{bmatrix} \mathbf{r}_{d,i}^T & \dot{\mathbf{r}}_{d,i}^T \end{bmatrix}^T$. The implemented proportional-integral (PI) controller is

$$\begin{bmatrix} \mathbf{r}_{d,i}^G \\ \dot{\mathbf{r}}_{d,i}^G \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{d,P}^G + \mathbf{s}_i^P \\ \dot{\mathbf{r}}_{d,P}^G + K_{Load,p}\mathbf{e}_{load} + K_{Load,i}\int_0^t \mathbf{e}_{load}\, d\tau \end{bmatrix}, \tag{4.6}$$

where $K_{Load,p} \in \mathbb{R}^{3\times 6}$ is the proportional gain matrix, $K_{Load,i} \in \mathbb{R}^{3\times 6}$ is the integral gain matrix, and $s_i^P$ is the planned position for vehicle $i$ in the payload frame. Note that this formulation assumes the vehicles are in a rigid formation that only translates and doesn't rotate. This outer loop payload PID controller commands a fixed vehicle position offset regardless of the payload error but adds correction commands to vehicle velocities in the direction of decreasing error. The vehicle controllers (described in Section 4.4.2) then track these commands to bring the payload towards its

desired setpoint. This strategy can be seen as an extension of the four-stage cascaded PID vehicle controller from which it was inspired.

Define the rigid multirotor formation through a set of azimuthal ($\gamma_i$) and elevation angles ($\alpha_i$) relative to the payload for each vehicle $i$. The planned vehicle positions are:

$$\mathbf{s}_i^P = R_{z,\gamma_i} R_{y,\alpha_i} \begin{bmatrix} l_{d,i} \\ 0 \\ 0 \end{bmatrix} + \mathbf{r}_{P,i}^P, \ \hat{\mathbf{s}}_i^P = \frac{\mathbf{s}_i^P}{\|\mathbf{s}_i^P\|}, \tag{4.7}$$

where $\hat{\mathbf{s}}_i^P$ is the planned vehicle direction relative to the payload and $l_{d,i}$ is the desired tether $i$ length. There are a variety of considerations when choosing the formation ($\gamma_i$,$\alpha_i$). Modifying $\alpha$, the "cone angle", affects vehicle separation, controllability, and efficiency [10]. A wider formation has larger vehicle separation and better controllability but worse energy efficiency since the vehicles waste effort pulling against each other. A tighter formation will have less vehicle separation, be less controllable, and more efficient. For this work, a balanced tradeoff between end cases was chosen. For the five vehicle ($N_V = 5$) system define $\gamma_i = \{\frac{\pi}{4}, \frac{3\pi}{4}, \frac{-3\pi}{4}, \frac{-\pi}{4}, 0\}$ and $\alpha_i = \{\frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{2}\}$ with one vehicle on top pulling directly against the payload weight and four vehicles pulling diagonally to balance lateral forces. Additionally, define payload mounting points

$$\mathbf{r}_{P,i}^P = R_{z,\gamma_i} R_{y,\alpha_i} \begin{bmatrix} \frac{d}{2} & 0 & 0 \end{bmatrix}^T. \tag{4.8}$$

In general, the tethers will apply desired forces along the tether vector, $-\hat{\mathbf{t}}_{P,i}^P$. However, for the chosen formation and mounting points, the planned tether vectors are collinear with the mounting point vectors such that the tethers will apply desired forces along $\hat{\mathbf{s}}_i^P$ (also collinear).

### 4.4.2 Quadrotor Controller

The goal of the quadrotor controller is to track the desired vehicle states $\mathbf{X}_{d,i} = \begin{bmatrix} \mathbf{r}_{d,i}^T & \dot{\mathbf{r}}_{d,i}^T \end{bmatrix}^T$ as commanded by the payload controller in real-time (Section 4.4.2.1). However, this may require the vehicles to apply horizontal forces which thus requires non-zero equilibrium attitudes and thrusts based on the formation (Section 4.4.2.2).

#### 4.4.2.1 Quadrotor Controller

The experimental system deployed five M330 quadrotors running the *rc_pilot_a2sys* flight controller. A 4-stage cascaded PID controller runs on each quadrotor to track the desired state (Chapter 2.3.1). This same controller was implemented for the simulation results. Position ($\mathbf{r}_{d,i}^G$) and velocity ($\dot{\mathbf{r}}_{d,i}^G$) setpoints are commanded by the payload controller (Eq. 4.6), while the feedforward

thrust ($f_{hover,i}$) and attitudes ($\phi_{FF,i}$,$\theta_{FF,i}$) are obtained via Section 4.4.2.2. Desired yaw ($\psi_{d,i}$) is set to a zero (North) local reference in this work without loss of generality.

#### 4.4.2.2 Feedforward Thrust and Attitude Calculation

Next, the feedforward thrust ($f_{hover,i}$) and attitudes ($\phi_{FF,i}$,$\theta_{FF,i}$) required to maintain hover equilibrium while carrying the payload are calculated. Given a rigid formation ($\gamma_i$,$\alpha_i$) solve for a static payload hover equilibrium considering tether forces and gravity as $\sum_{i=1}^{n} \mathbf{F}^P_{Tether,i} = -\mathbf{F}^P_{Gravity,P} = \begin{bmatrix} 0 & 0 & -m_p g \end{bmatrix}^T$. Defining a weighting factor $0 \leq \mu \leq 1$ allows shifting the planned load distribution between the center vehicle and the other four vehicles such that $\mathbf{F}^P_{Tether,5} = -\mu \mathbf{F}^P_{Gravity,P}$ and $\sum_{i=1}^{4} \mathbf{F}^P_{Tether,i} \cdot \mathbf{e}_3 = -(1-\mu)\mathbf{F}^P_{Gravity,P} \cdot \mathbf{e}_3$. The remaining components of $\mathbf{F}^P_{Tether,i}$ are given by $\left\|\mathbf{F}^P_{Tether,i}\right\| = \frac{-(1-\mu)\mathbf{F}^P_{Gravity,P}\cdot\mathbf{e}_3}{4\hat{\mathbf{s}}^P_i \cdot \mathbf{e}_3}$, where $\mathbf{F}^P_{Tether,i} = \left\|\mathbf{F}^P_{Tether,i}\right\| \hat{\mathbf{s}}^P_i$. Desired tether lengths are then

$$l_{d,i} = \frac{\left\|\mathbf{F}^P_{Tether,i}\right\|}{k_i} + L_i. \tag{4.9}$$

Because the vehicles are identical, define $\mu$ such that every tether force pulling on the payload is equal, i.e., $\left\|\mathbf{F}^P_{Tether,i}\right\| = \left\|\mathbf{F}^P_{Tether,j}\right\|, \forall i, \forall j$. Note that if a larger, more powerful center vehicle were deployed, $\mu$ could increase so that the center vehicle would do the majority of the carrying and the other vehicles primarily serve to stabilize the payload. Quadrotors force magnitudes are then

$$\left\|\mathbf{F}^P_{Tether,i}\right\| = \begin{cases} \mu m_P g & i = 5 \\ \frac{(1-\mu)m_P g}{2\sqrt{2}} & 1 \leq i \leq 4 \end{cases}. \tag{4.10}$$

With all five force magnitudes equal, $\mu = \frac{1}{2\sqrt{2}+1} \approx 0.26$

Applied gravity, tether, and thrust forces are balanced when $\mathbf{F}^G_{Thrust,i} = \mathbf{F}^G_{Tether,i} - \mathbf{F}^G_{Gravity,i}$. Substituting and expanding terms:

$$\begin{bmatrix} -\cos\phi_i \sin\theta_i f_{thrust,i} \\ \sin\phi_i f_{thrust,i} \\ -\cos\phi_i \cos\theta_i f_{thrust,i} \end{bmatrix} = R^G_P \mathbf{F}^P_{Tether,i} - \begin{bmatrix} 0 \\ 0 \\ m_V g \end{bmatrix} = \begin{bmatrix} f_{d,i,1} \\ f_{d,i,2} \\ f_{d,i,3} \end{bmatrix},$$

where $f_{d,i,1}, f_{d,i,2}, f_{d,i,3}$ are desired force components. Solving this equation provides feedforward thrust and attitude commands for each vehicle $i \in \{1, ..., N_V\}$

$$f_{hover,i} = \left\| \begin{bmatrix} f_{d,i,1} \\ f_{d,i,2} \\ f_{d,i,3} \end{bmatrix} \right\|, \ \phi_{FF,i} = \arcsin\left(\frac{f_{d,i,2}}{f_{thrust,i}}\right), \ \theta_{FF,i} = \arctan\left(\frac{f_{d,i,1}}{f_{d,i,3}}\right). \tag{4.11}$$

## 4.5 User Force Estimation

Newton's Second Law for the payload in the ground frame is

$$\mathbf{F}^G_{net,P} = \mathbf{F}^G_{Gravity,P} + \mathbf{F}^G_{Drag,P} + \sum_{i=1}^{N_V} \mathbf{F}^G_{Tether,i} + \mathbf{F}^G_{User} = m_P \mathbf{a}^G_P. \tag{4.12}$$

User force, tether forces, drag, and gravity act on the payload and the net force produces a payload acceleration. The payload will move slowly, therefore a quasi-static assumption is made ($\mathbf{a}^G_P \approx \mathbf{0}_3$). Also, for estimation purposes drag will be assumed to be negligible ($\mathbf{F}^G_{Drag,P} \approx \mathbf{0}_3$). Solving Eq. 4.12 for user force with these assumptions yields

$$\mathbf{F}^G_{User} = -\mathbf{F}^G_{Gravity,P} - \sum_{i=1}^{N_V} \mathbf{F}^G_{Tether,i}, \tag{4.13}$$

where the constant gravity force $\mathbf{F}^G_{Gravity,P} = \begin{bmatrix} 0 & 0 & m_P g \end{bmatrix}^T$. The instrumentation measures individual tether force magnitudes $f_{Tether,i}$ and directions $\hat{\mathbf{t}}^G_{P,i}$ which can be combined as:

$$\mathbf{F}^G_{Tether,i} = f_{Tether,i} \left( -\hat{\mathbf{t}}^G_{P,i} \right). \tag{4.14}$$

To reduce noise, this estimate (Eq. 4.13) is processed with a low pass filter digital transfer function H with cutoff frequency $f_c$ and sampling frequency $f_s$ in Hz. An exponential averaging law of the following form was chosen:

$$y[k] = \alpha y[k-1] + (1-\alpha)x[k] \tag{4.15}$$

$$H(z) = \frac{1-\alpha}{1-\alpha z^{-1}} \tag{4.16}$$

where $y[k]$ is the filter output at time step $k$, $x[k]$ is the input, and $\alpha$ is the filter parameter. It performs an average of inputs, with exponentially decreasing weights for older measurements. However, it achieves this with only one computation of Eq. 4.15 per time step. Thus, this filter is computationally inexpensive and suitable for real-time implementation on a lightweight embedded processor. $f_s = 200Hz$ and through experimentation $f_c = 1.6Hz$ was chosen which translates to $\alpha = 0.95$ as sufficient for noise reduction without adding too much filter delay [70]. The haptic guidance application requires relatively low latency for the user to effectively guide the system. To account for unmodelled effects, a constant offset $\mathbf{F}_{Zero}$ was introduced. The full user force estimation equation is then:

$$\hat{\mathbf{F}}^G_{User} = H(s)\left(-\mathbf{F}^G_{Gravity,P} - \sum_{i=1}^{N_V} \mathbf{F}^G_{Tether,i}\right) - \mathbf{F}_{Zero}. \tag{4.17}$$

Offset $\mathbf{F}_{Zero}$ is set/calibrated at the beginning of each experiment while the payload hovers in the air with no user applied force input per $\mathbf{F}_{Zero} = \hat{\mathbf{F}}^G_{User}$. This accounts for any constant offset in force estimate for the system. Without it, a constant offset is sensed as an applied force and would cause the system to drift. Thus, using it, the intent of the user's applied force can be correctly obtained.

## 4.6 Payload Guidance with Admittance Control

Payload guidance is based on user applied force inputs plus preset autonomous team behaviors such as liftoff to fixed-altitude hover and coordinated landing. The haptic guidance goal is for the payload to act like a neutrally buoyant object that is easily moved by the user but not impacted by other forces. An admittance controller can provide this capability. With admittance control, force applied to a system goes through virtual dynamics that control the position and velocity setpoint the system tracks over time. Since the dynamics are purely virtual, they can be designed to achieve a desired system behavior (e.g., hockey puck on ice, fly in molasses) [60]. The admittance controller is given user applied force as input. It calculates a desired payload position ($\mathbf{r}^G_{d,P}$) and velocity ($\dot{\mathbf{r}}^G_{d,P}$) to guide slung payload motion by user force inputs over time. A full 3D admittance controller is presented next, but only a 2D version with constant payload altitude was flight tested because the payload was required to remain fixed at shoulder height.

Virtual dynamics for the admittance controller are given by

$$m_{ac}\ddot{\mathbf{r}}^G_{d,P} + c_{ac}\dot{\mathbf{r}}^G_{d,P} = \hat{F}^G_{User}, \tag{4.18}$$

where $m_{ac}$ is virtual mass and $c_{ac}$ is virtual damping coefficient. Choosing $m_{ac} > 0$ and $c_{ac} > 0$ yields mass-damper virtual dynamics

$$\frac{R_d(s)}{F(s)} = \frac{1}{m_{ac}s^2 + c_{ac}s}. \tag{4.19}$$

While mass-damper dynamics for admittance control is not new ([57, 60]), its use in airborne cooperative payload guidance is new along with the tuning procedure and parameter definitions presented in this chapter.

Defining $m_{ac} = 0$, $c_{ac} > 0$ results in a massless-damper system in which desired velocity would be directly proportional to applied force ($c_{ac}sR_d(s) = F(s)$). A massless virtual system would present no inertia when a user applied force or stopped. Values $m_{ac} > 0$, $c_{ac} = 0$ would yield a system in which a moving payload would never stop unless the user applied force in the other

direction (e.g., a hockey puck on friction-less ice). Therefore virtual dynamics with positive mass and damping coefficient was selected to get the desired response; that is when the user applies force the payload moves and when they stop the payload stops.

To handle disturbance rejection when not being actively manipulated a threshold function was included as shown in Figure 4.4. When the norm of the input force exceeds the activation force, $F_{min}$, the input force is passed through. Otherwise, a zero force is passed through. Ref. [57] uses a four parameter method to capture similar logic to activate and deactivate an admittance controller running directly on each multirotor.

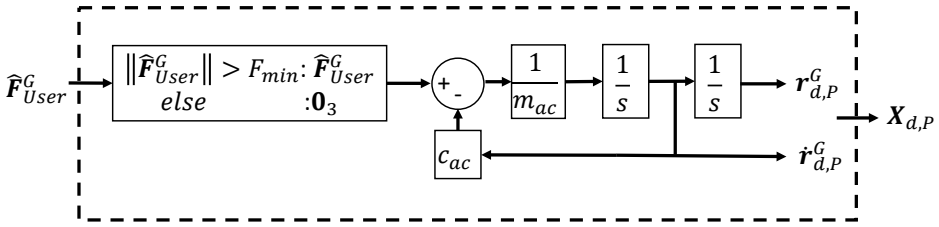



Figure 4.4: Mass-damper virtual dynamics

## 4.7 Experimental Setup

A snapshot of the experimental setup can be seen in Figure 4.5. It consists of five quadrotors, a custom-built instrumented payload, and a ground control station. Further details are provided in the following sections.

(**a**) Quadrotors and payload prepared for tethered flight.

(**b**) Ground control station with communication links.

Figure 4.5: Experimental setup. A ground control station communicates between the payload and quadrotors.

### 4.7.1 System Level Description

The system level experimental setup can be seen in Figure 2.1. Flight tests were conducted in M-Air, an outdoor netted flight facility equipped with 34 Qualisys motion capture cameras that provide real-time six degree-of-freedom pose data for the payload and quadrotors. A Ground Control Station (GCS) computer receives pose updates from the motion capture system over a local ethernet network and relays them to the quadrotors and payload over wireless serial connection.

A 2.4 GHz WiFi router connects the GCS computer, payload, and quadrotors. The GCS operator can view real-time telemetry data from each vehicle and the payload. Payload flight modes can be initiated via keyboard commands from the GCS operator. Quadrotor trajectory commands are sent directly from the payload to the vehicles over WiFi.

The "user" is tasked with operating the system using haptic guidance. While admittance control mode is enabled the user approaches the system and applies force on the payload to guide it. Additionally, a safety pilot stands by with a transmitter that connects to each vehicle and enables switching into a manual flight mode for safe landing (via throttle regulation) or activating a kill switch to turn off the motors in an emergency.

### 4.7.2 Instrumented Payload

An instrumented payload capable of real-time sensor data acquisition is shown in Figure 2.11 and described in detail in Chapter 2.5. Five tether lines with inline load cell tension sensors and a haptic force validation sensor are mounted to the 24.6 cm sphere. A BeagleBone Blue single-board Linux computer receives data from the tension sensors, haptic force validation sensor, motion capture system, and samples its onboard 9-axis IMU. Power is provided by a 3 s (3-cell), 3000 mAh Lithium Polymer (LiPo) battery. The instrumented payload weighs 1.211 kg.

The load cell force sensors were assumed to have an affine model (scale and offset) and were calibrated using two known masses. After calibration, 1D forces, $\bar{f}_i$, are computed from the raw ADC values. The 3D tension forces are obtained via $\mathbf{F}^P_{Tether,i} = R_{z,\gamma_i} R_{y,\alpha_i} \bar{f}_i$ which assumes that the vehicles are in their desired known formation. The 3D haptic validation force is computed as $\mathbf{F}^G_{User} = R^G_P \begin{bmatrix} f_P & 0 & 0 \end{bmatrix}^T$, which assumes the applied force is directly into the sensor.

### 4.7.3 System Operation

Figure 4.6 gives an overview of the system operation for the instrumented payload. All computation and commands are generated and sent from the BeagleBone Blue computer embedded in the instrumented payload. The GCS operator is able to initiate flight mode transitions via "Keyboard inputs" and the payload itself is able to transition via "Events" whenever the takeoff and landing

trajectories are completed. Additionally, the user force estimate can be zeroed out at anytime via keyboard input.

The instrumented payload begins in the Grounded state. Here, the grounded positions for each vehicle (Figure 4.5a) are given with zero velocity commands. After the GCS operator commands a takeoff, the flight mode transitions into Taking Off. Within Taking Off, a 2-stage trajectory is followed for each vehicle consisting of a quintic spline trajectory from the grounded positions to the tensioning point (Eq. 4.6 with payload at origin), and then another from the tensioning point to the hover point. Once the trajectory is finished, the instrumented payload enters into the Hovering flight mode. In Hover, all of the controllers begin to run as described in Section 4.4 except for the admittance controller. From this point forward, the GCS operator can transition into Landing mode via a keyboard input which follows the Taking Off trajectory identically except in reverse. Alternatively, they can enter the Admittance Control flight mode which enables the admittance controller (Section 4.6). Landing can also be initiated from Admittance Control.

In Admittance Control, a geofence boundary on the desired payload position is in place so that the vehicles remain within ideal view of the motion capture cameras (Figure 4.7). A conservative bound on the motion capture coverage area is 6 m x 6 m. Corner vehicles are offset from the payload by 1.075 m in both $\hat{\mathbf{x}}_G$ and $\hat{\mathbf{y}}_G$ and it is expected that disturbances and controller error may move the vehicle up to 1 m from its setpoint. This results in a 2 m x 2 m square geofence centered at the origin to ensure safety. When the payload is within it, the system behaves as described in the relevant guidance and control sections. At the limit, the system rejects all haptic guidance commands that would force it outside of this square and only accepts commands that push it along the boundary or back inside it.

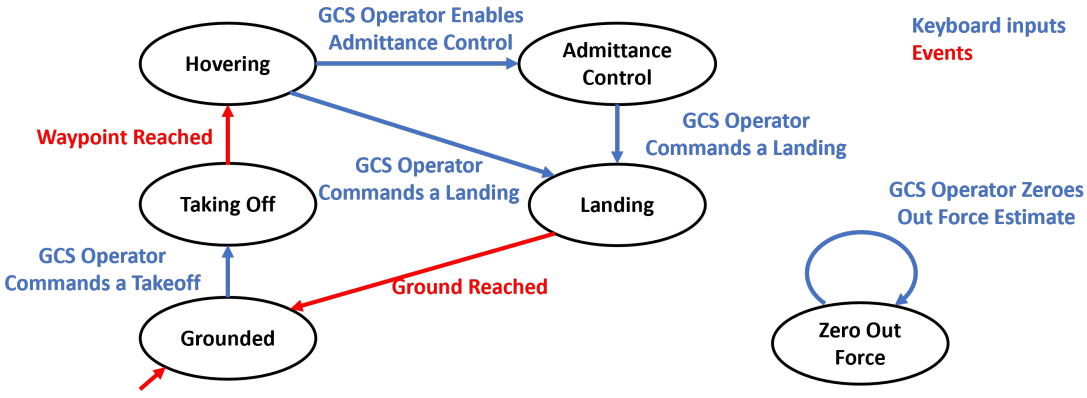



Figure 4.6: Instrumented payload flight modes state machine (left) and force estimate zeroing (right).

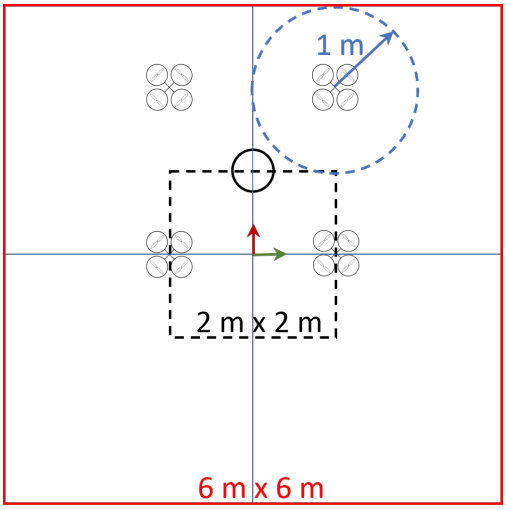



Figure 4.7: Flight test area including motion capture area (red), geofence (black), and vehicle error bound (blue).

### 4.7.4 Quadrotor Vehicles

Five M330 quadrotors were used for flight tests (Figure 4.5a with description in Chapter 2.2). Underneath, a tether mounting point is tied such that the tether pulls directly on the center of mass of the vehicle. A 4 s (four-cell), 2650 mAh battery powers the system and a Beaglebone Blue single-board Linux computer runs the `rc_pilot_a2sys` flight controller [15]. A series of cascaded PID control loops track desired setpoints sent over a WiFi link from the payload via MAVLink messages [71].

### 4.7.5 Virtual Dynamics Parameter Selection

The virtual mass was chosen as 1.211 kg, which is the actual payload mass. Through some initial experimentation an activation force of 1.5 N was chosen which was small enough to register comfortable applied forces but large enough to reject non-user disturbances. For safety, a maximum virtual system speed of 0.25 m/s was set. The damping coefficient is calculated using $c_{ac} = \frac{F_{ss}}{v_{ss}}$, where $F_{ss}$ is the steady state force and $v_{ss}$ is the steady state velocity. Assuming $F_{ss} = 2.5N$ and $v_{ss} = 0.25m/s$, then $c_{ac} = 10$. At the activation force of 1.5 N the steady state velocity is 0.15 m/s.

## 4.8 Simulation Results

MATLAB/Simulink was used to implement and evaluate the controller, force estimator, and guidance using the payload and vehicle models and standard rigid body dynamics [72]. The simulation code can be found in a publicly-accessible repository [73]. The main goal of simulation

was to validate the equations and methods before implementing them in experimental hardware.

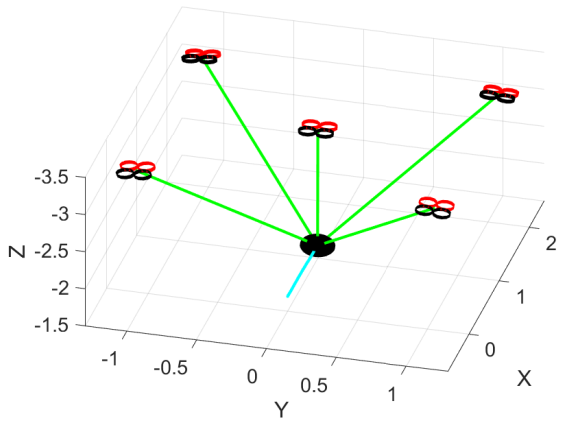
Table 4.1 shows the simulation parameters. Physical parameters ($m_P, m_V, L_i$) were selected to match the experimental system. Tether spring constant $k_i$ was obtained experimentally by hanging weights of different masses on the tether, measuring the extension, and then making a linear fit. Tether damping coefficient $b_i$ was selected in the simulation such that the dynamics matched the damped behavior. Virtual dynamics parameters ($m_{ca}, c_{ca}$) were chosen per Section 4.7.5. Zero mean, band-limited, white, Gaussian noise was used to simulate sensor noise and disturbance forces on the payload and all five vehicles. The magnitude of the power spectral density (PSD) function for this noise was selected to adjust the intensities. Define $P_{sensor}$ and $P_{dist}$ as the magnitude of the PSD for state feedback (position, velocity, attitude, attitude rate) and disturbance forces respectively. Noise values were chosen to match the experimental system. Aerodynamic drag forces were modeled assuming standard temperature and pressure. Spherical payload drag force was modeled based on Ref. [74] which is valid for relative wind speeds of 12.0 m/s or less. A lumped drag model was used for quadrotor drag [75, 76]. Wind velocity was set to 0 relative to the ground frame. Therefore, drag forces were only experienced while bodies were in motion relative to the ground frame.

Table 4.1: Simulation parameters

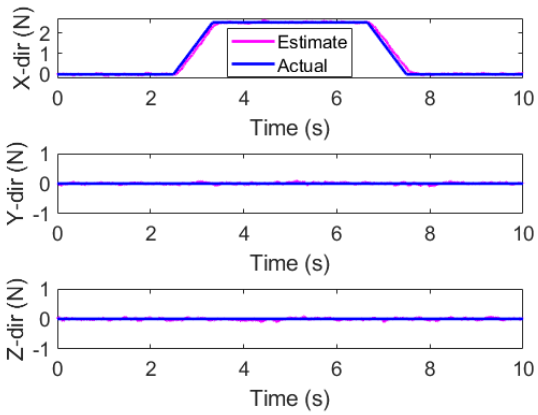
| $m_{ca}$ | $c_{ca}$ | $m_P$ | $m_V$ | $L_1$-$L_4$ | $L_5$ | $k_i$ | $b_i$ | $P_{sensor}$ | $P_{dist}$ |
|---|---|---|---|---|---|---|---|---|---|
| 1.21 kg | 10 | 1.21 kg | 1.05 kg | 2.02 m | 1.44 m | 750 N/m | 40 Ns/m | $10^{-8}$ | $10^{-5}$ |

$$K_{Load,p} = \begin{bmatrix} 0.7 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0.7 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0.7 & 0 & 0 & 0.2 \end{bmatrix}, K_{Load,i} = \begin{bmatrix} 0.05 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.05 & 0 & 0 & 0 \end{bmatrix}.$$

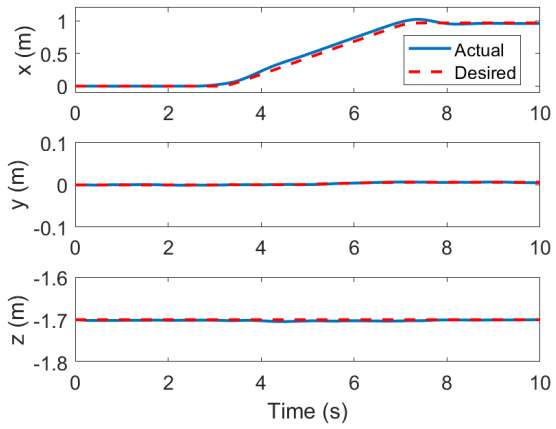
A forward applied force simulation was conducted to validate the system. Figure 4.8 shows the results for a forward applied force of 2.5 N. Within the first 2 seconds the payload controller stabilizes the system at hover with no applied force exerted on the payload. Next, the user force is applied in a ramp up, hold, and ramp down sequence (Figure 4.8b). This causes the virtual dynamics of the admittance controller to update between 3 seconds and 7 seconds with a positive $\hat{\mathbf{x}}_G$ desired velocity. During this applied force, the payload controller tracked the desired trajectory with an overshoot of less than 10 cm. The velocity was tracked as well with a 10 cm/s overshoot when manipulation began and a 15 cm/s overshoot when it ended. This overshoot behavior is expected to be seen in the experimental system, but is acceptable since the position (which is the focus of the controller) tracked effectively. The haptic force estimate followed the actual value within about 0.1 N for the majority of the test with slightly higher error during the ramp up and ramp down phases.



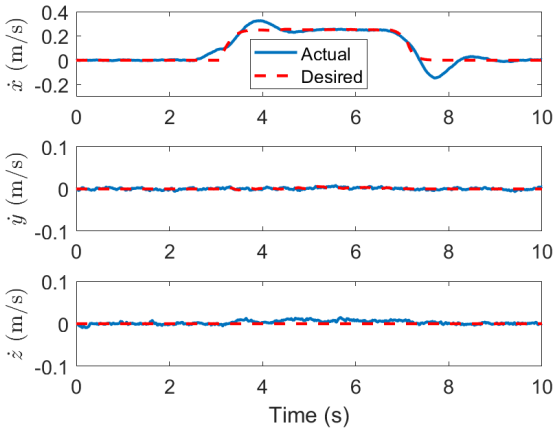

(**a**) 3D view at 10s (blue payload path, green tethers).




(**b**) Haptic Force Estimate




(**c**) Payload Position




(**d**) Payload Velocity

Figure 4.8: Haptic guidance simulation results with a 2.5N user force guiding the system forward.

## 4.9 Experimental Results

A flight validation was conducted using a progression from simple ground test runs to complex flying test runs. Table 4.2 summarizes the conditions of flight test runs performed. The force estimation method and associated sensor sensor data in Section 4.9.1 (Test A1) was validated first. Next, the haptic guidance was tested in both a software test (B2) as well as a remote, haptic guidance flight test (B3) shown in Section 4.9.2. Finally, the full system was tested with the user applying force directly on the payload to guide the team of vehicles as they carried it (test C2) in Section 4.9.3. An accompanying video of the flights can be found in Ref. [77].

Table 4.2: Flight test conditions

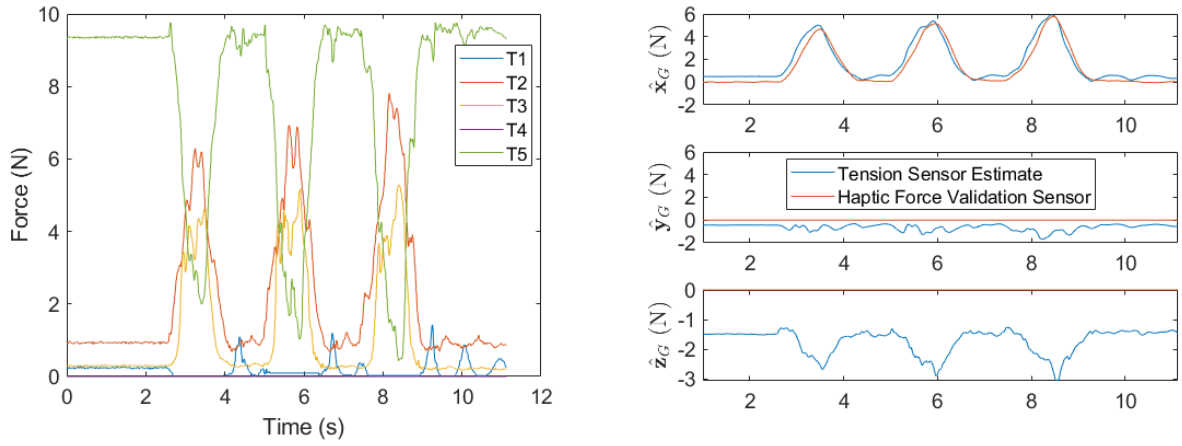
| Test | Payload | Vehicles | Haptic Guidance | Payload Controller |
|---|---|---|---|---|
| A1 | In Test Rig | Grounded | N/A | N/A |
| B2 | In Test Rig | Grounded | Enabled | N/A |
| B3 | In Test Rig | Flying | Enabled | Feedforward Only |
| C2 | In Flight | Flying | Enabled | Full |

### 4.9.1 User Force Estimation Validation

Figure 4.9 shows the test results that validate the user force estimation. The instrumented payload was tethered to the test rig as shown in Fig 2.11a. The user was instructed to give three light presses onto the payload where the haptic force validation sensor was placed. This provided ground truth and standard tension force data of user applied force. The tension force data was used to compute a user force estimate which was compared against the force data coming from the haptic force validation sensor. Both values agree in magnitude for $\hat{\mathbf{x}}_G$ and $\hat{\mathbf{y}}_G$. The estimate in $\hat{\mathbf{z}}_G$ does not. This is likely due to the applied force being slightly upwards (negative $\hat{\mathbf{z}}_G$) as the payload rotates (violating the positive $\hat{\mathbf{x}}_G$ force application assumption) as well as a steady state force offset resulting from the direction vectors being more vertical than assumed. Additionally, as the payload moves relative to its mounting points, the assumption that the tether vectors are fixed in their equilibrium positions (which was used to calculate the 3D tether forces from 1D sensors) is potentially violated, resulting in the upwards force estimate. Due to the experimental system using 2D lateral haptic guidance, the $\hat{\mathbf{z}}_G$ estimate was not needed. In time, the haptic force validation data lags by about 100 ms. This can be attributed to known differences in the sampling rates of the load cell ADCs (tension sensors at 80 Hz, haptic force validation sensor at 10 Hz) which caused a slightly higher delay for the haptic force validation sensor. However, the 10 Hz sampling mode provides half of the sampling noise as compared with the 80 Hz mode which makes it suitable to act as a ground truth for magnitude. Additionally, no zeroing was used for this test to obtain a baseline for system performance for future tests.

### 4.9.2 Haptic Guidance Validation

Haptic guidance was validated via an isolated test. The instrumented payload was attached via tethers to the test stand. User force estimation and payload guidance were active and run normally, while the payload and vehicle controllers were inactive. The user applied force to the instrumented payload while the payload guidance commands were recorded in software. This test was performed to validate the parameters from Section 4.7.5 for the admittance controller as a whole, and demonstrate 2D guidance commands. Figure 4.10 shows the results, where at 26 s a




(**a**) Tension Sensor Data

(**b**) User Force Estimate

Figure 4.9: User force estimation validation (Table 4.2: Test A1).

positive $\hat{\mathbf{x}}_G$ applied force of about 3.0 N was estimated. This commanded a positive $\hat{\mathbf{x}}_G$ velocity and increased the $\hat{\mathbf{x}}_G$ position command from 0.0 m to 0.6 m until the applied force was released at 28 s. At 31 s another positive $\hat{\mathbf{x}}_G$ applied force of about 2.5 N was estimated. However, at 32 s the 1.0 m geofence boundary was reached ($\times$ symbol). This caused the commanded velocity in $\hat{\mathbf{x}}_G$ to go to zero and the $\hat{\mathbf{x}}_G$ position command to be limited at 1.0 m. At 35 s a diagonal press (negative $\hat{\mathbf{x}}_G$ and negative $\hat{\mathbf{y}}_G$) was estimated. This caused both the $\hat{\mathbf{x}}_G$ and $\hat{\mathbf{y}}_G$ velocity commands to be negative and decreased the position commands accordingly. Using the activation force threshold on the input of the admittance controller was effective at distinguishing user applied force from model errors or other disturbance forces. Additionally, using the payload mass as the virtual mass worked well.

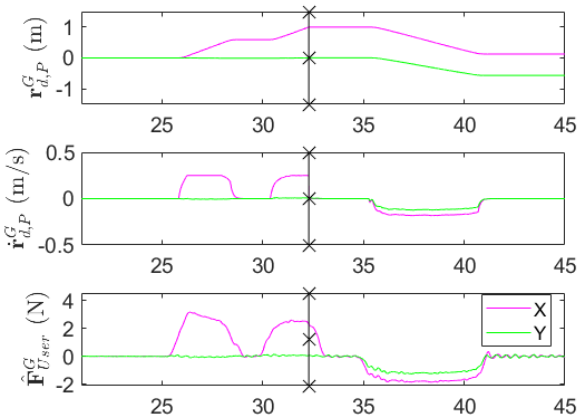
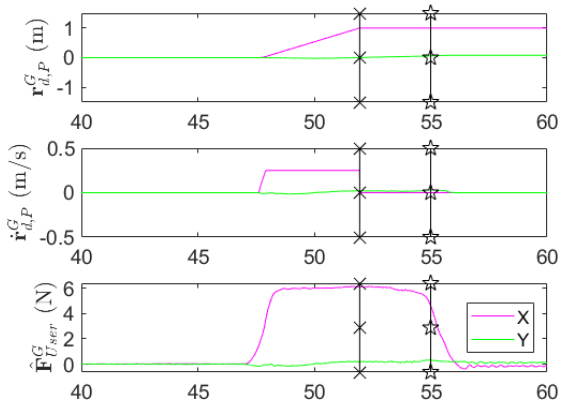



Figure 4.10: Admittance controller validation (Table 4.2: Test B2). User applied force updates payload guidance commands.

For the second test, a remote, haptic guidance flight was performed (test B3 shown in Figure

4.11). The user applied forces to the instrumented payload in the test rig (located outside of the flight area). However, the commanded trajectories were used as guidance for the quadrotor team in flight. This test began with the vehicles in their grounded positions. After following the takeoff trajectory, they assumed their formation positions and the user force estimate was zeroed out before enabling admittance control mode. The operator applied a positive $\hat{\mathbf{x}}_G$ force on the payload until reaching the 1 m limit in $\hat{\mathbf{x}}_G$ at 52 s ($\times$ symbol). The applied force was released at 55 s ($\star$ symbol). Afterwards, admittance control mode was disabled and the vehicles were commanded to land. The virtual dynamics were designed with a 2.5 N control force in mind, such that a steady state 2.5 N applied force results in a steady state 0.25 m/s commanded velocity. However, for this test a roughly 6 N force was applied (Figure 4.11b). The commanded velocity reached the limit of 0.25 m/s in 300 ms. However, without the limit, the 6.0 N force could have resulted in a 0.60 m/s steady state velocity. Even larger forces would command even larger steady state velocities, at which tracking performance would likely degrade. A user interacting with a system exhibiting these characteristics would be placed at risk and find haptic guidance difficult. Therefore, this velocity limit is important in maintaining a safe operating environment for the user even when commanded with larger than expected user forces.

(**a**) Third person view at roughly 50 seconds.




(**b**) Haptic Force Estimate

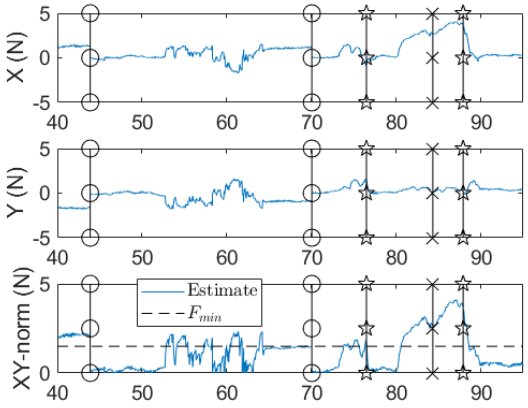
Figure 4.11: Admittance controller validation (Table 4.2: Test B3). An approximately 6 N user force guides the system forward.

### 4.9.3 Full System Validation

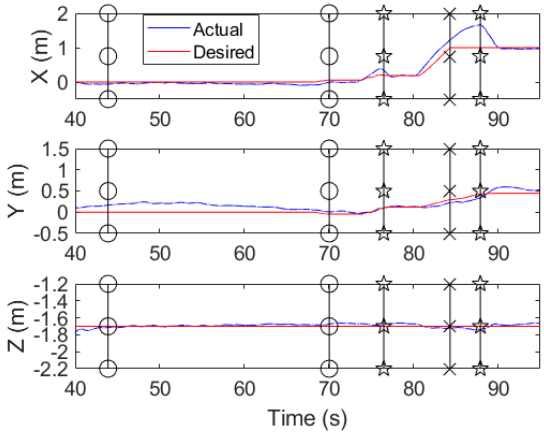
The full system test (C2) used the same setup as B2 except the instrumented payload was tethered to the vehicles, the full payload controller was enabled, and the user was inside the arena to provide direct haptic guidance. Figure 4.12 shows the results. The Circles ($\circ$) indicate when the

force estimation was zeroed, the Crosses ($\times$) indicate when the geofence boundary was hit, and the Stars ($\star$) indicate when the user applied forces were released.

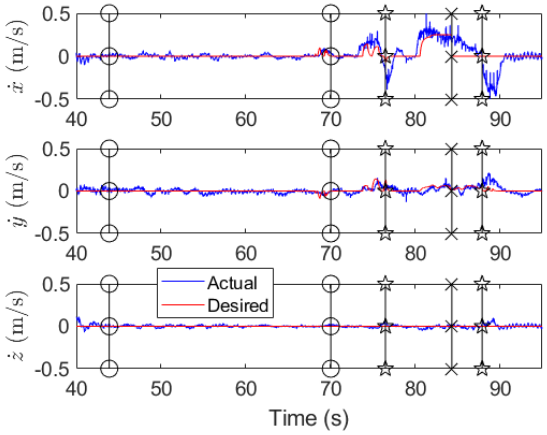
(**a**) Third person view at roughly 82 seconds.




(**b**) Tension Sensor Based Haptic Force Estimate




(**c**) Payload Position




(**d**) Payload Velocity

Figure 4.12: Full system test (Table 4.2: Test C2). A small 2N force followed by a larger 4N force guides the system forward.

After takeoff, the vehicles assumed their formation positions and the force estimate was zeroed (first $\circ$). Figure 4.12b displays, from top to bottom, the tension sensor based estimate in $\hat{\mathbf{x}}_G$, the estimate in $\hat{\mathbf{y}}_G$, and the Euclidean norm of those two quantities. The horizontal dotted line in the bottom plot indicates the activation force, $F_{min}$, which controls when forces are input into the admittance controller. When the first zeroing event occurs all three quantities reset to zero. At this point, a payload position error in $\hat{\mathbf{y}}_G$ of about 20 cm was present (Fig. 4.12c middle). Next, the system was given time to settle and the integral portion of the payload controller pushed the system towards its setpoint. Note that the gains chosen for the feedback term of the payload controller were conservative to increase safety for the user interacting with the system, yet it was still effective in

reducing the steady state tracking error.

At 68 s admittance control mode was enabled and the user approached the system. A second zeroing was triggered at 70 s (second $\circ$). The user was then instructed to apply force in the positive $\hat{\mathbf{x}}_G$ direction. The haptic force estimate norm was approximately 2 N for 2 seconds (Fig. 4.12b bottom near 75 s). Although, the user intended to apply force in purely the $\hat{\mathbf{x}}_G$ direction, both a positive $\hat{\mathbf{x}}_G$ and positive $\hat{\mathbf{y}}_G$ force were estimated. This is likely due to the true applied force not being solely in $\hat{\mathbf{x}}_G$ combined with small calibration and modelling errors. This haptic force exceeded the activation force and caused the admittance controller to update the setpoint (Figs. 4.12c and 4.12d). The commanded position in $\hat{\mathbf{x}}_G$ increased from 0.0 m to 0.2 m and also in $\hat{\mathbf{y}}_G$ from 0.0 m to 0.1 m. The payload controller tracked this setpoint with an overshoot error of 18 cm. When the user force is below the activation force it is no longer updating the setpoint via the admittance controller and therefore acts as a disturbance to the system, likely contributing to the overshoot. This happened between 75.6 s and 76.3 s before the user force was released at 76.5 s (first $\star$). After release, the payload converged to its setpoint in position.

At 80 s, the user applied a second force with an approximate 4 N magnitude. During this force application the admittance controller updated the setpoint in $\hat{\mathbf{x}}_G$ from 0.2 m to the 1.0 m limit and in $\hat{\mathbf{y}}_G$ from 0.1 m to 0.3 m. The actual payload position followed this guidance and led slightly by about 20 cm. At 84.3 s, the 1.0 m $\hat{\mathbf{x}}_G$ limit was reached ($\times$ symbol). After this, all haptic force inputs in the positive $\hat{\mathbf{x}}_G$ direction were set to zero so that the payload position setpoint remained within the geofence boundary. The $\hat{\mathbf{x}}_G$ haptic force estimate was positive therefore the $\hat{\mathbf{x}}_G$ position setpoint remained constant at 1.0 m and the $\hat{\mathbf{x}}_G$ velocity setpoint was zero. Therefore, between the $\times$ and second $\star$ symbols the user force acted like a disturbance on the payload since it was no longer updating the admittance controller. Because the conservatively tuned payload controller and user force were fighting in opposite directions, a steady state error of up to 67 cm was experienced before the user applied force was released (second $\star$) and the system converged back to its setpoint. Afterwards, the user backed away from the system before admittance control mode was disabled and the system landed. The conservatively tuned payload controller was effective with average 3D tracking error of 14.9 cm before the geofence boundary was hit (between 40s and 84.3 s) and 17.9 cm for the full time period.

While the admittance controller is enabled, the force of the user actually helps to guide it along the trajectory and work with the payload controller. When the admittance controller is disabled but there still exists a user force it acts as a disturbance and works against the payload controller. For the user this is like force feedback that can be used to determine where the virtual boundary is. At the geofence boundary, it becomes harder to push the payload beyond the limit because the system applies a greater control effort the further away it gets from its setpoint. This geofence system behavior was effective and should be used in further development.

## 4.10 Conclusion

This chapter has presented the first experimentally validated haptic guidance system for cooperative airborne payload transportation. A user force estimation method was developed that explicitly estimates the force via multiple, inline tension sensors. Payload guidance was updated using this force via virtual dynamics prescribed by the admittance controller running on the instrumented payload. Flight test results validated the system with a series of real user applied force inputs, where user force was successfully distinguished from other disturbances and used to intuitively guide the payload.

An interesting behavior that emerged from this haptic guidance concept is related to the interplay between applied user force, controller effort, and tracking performance. With the admittance controller off and no user force applied, the system was able to track its setpoint effectively. Acceptable tracking performance also occurred during manipulation with the admittance controller activated. The applied user force worked in tandem with the payload controller to move it along the desired trajectory. A reduced controller effort from the payload controller was needed as compared with non-haptic guidance. In fact, in the ideal case (no delays, noise, disturbances), since the virtual mass was selected as the payload mass there should be no controller effort as the applied force should perfectly move the payload along the generated trajectory. Lastly, with the admittance controller deactivated and user force applied, tracking error was non-trivial. This is because the applied force acted as a disturbance to the system, rather than as guidance. This happened when applied force was below the activation threshold, which resulted in minor overshoot. Additionally, when a geofence boundary was reached this also occurred, with significant overshoot. However, this can be seen as a feature of the geofence system, where the user is allowed to apply force and receive feedback from the system to indicate where the boundary is located without having to consult maps. This lends itself further to being an intuitive guidance scheme for an untrained user. In a deployed system, geofence boundaries will typically be much larger and encountered less frequently.

A full 3D haptic guidance system was presented, but only a constant shoulder height 2D guidance system was validated experimentally. This was due to the experimental estimate of user applied force in the z-direction being coupled to lateral pushes. This is likely due to the assumption of fixed tether directions used in force estimation being violated. To improve z-force estimation and force estimation in general, an estimate of tether directions based on the real-time relative positions of the vehicles and payload should be used to calculate a more accurate 3D tether force estimate. However, even with the data as-is, if a separate threshold on force input in the z direction was used and sized appropriately to reject these small forces (e.g. 3 N based on the data), this method could still work. This would require there to be two separate admittance controllers running a 2D lateral controller and a 1D vertical controller each with different force input thresholds. A vertical

geofence boundary could also be used to keep the payload within grasp of the user.

Future work is recommended to achieve haptic guidance in a deployed system. This chapter shows that the multilift haptic guidance concept is feasible. The instrumented payload facilitated centralized force estimation via inline tension sensors and force validation via the haptic force validation sensor. However, in a real world application, the payload would not be instrumented. Alternatively, the tension sensors could be placed inline with the tethers on the vehicle side. Vehicles would then be able to share information in either a distributed or centralized manner to arrive at a user force input estimate. Force estimation can be improved via automated force zeroing and real-time calibration (e.g. Kalman filter to estimate scale and biases). Replacing motion capture with a real-time kinematic positioning (RTK) GPS for position and velocity estimates and vehicle-mounted cameras for relative pose estimation is also recommended in future work. Lastly, consideration of resiliency to single vehicle and/or single tether failure should be explored to leverage the full benefits of a multilift system.

# CHAPTER 5

# Multi-UAS Wildfire Mapping

## 5.1 Introduction

Wildfires have the potential to inflict serious harm on people and property. In 2018, California suffered the worst wildfire season ever recorded; 1.8 million acres burned, 17,133 residences were destroyed, and more than 100 people died due to the fires [78]. Unmanned aircraft system (UAS) platforms have the potential to save lives and reduce damage through early fire detection and mapping. However, significant research and beyond-visual-line-of-sight (BVLOS) testing must take place before UAS can have appreciable fire surveillance impact.

Wildfire identification and boundary estimation requires a UAS team to offer large-scale area coverage. The UAS might need to map multiple, dynamic fires with range-limited onboard sensors to provide an accurate estimate of fire boundaries in real-time. Two distinct sub-problems are addressed in this work: multi-UAS path planning and fire boundary estimation. The planning problem requires commanding and controlling a team of UAS to safely traverse potentially mountainous terrain, avoid wildfire damage, and acquire data of wildfire exterior boundaries using the modest-cost sensors a UAS might reasonably carry. The boundary estimation problem requires fusing all incoming UAS sensor readings, accounting for data age as well as content, to produce a global estimate of dynamic fire boundaries.

The 2019 Air Force Research Lab (AFRL) Swarm and Search AI Competition was aimed at solving both the multi-UAS path-planning and wildfire boundary estimation problems. It asked competitors to develop algorithms for detecting and mapping a dynamic fire boundary in a simulation environment using a team of fixed-wing UAS [79]. AMASE (Aerospace Multi-Agent Simulation Environment) was used as the simulation environment; an AMASE GUI screenshot for one of the scenarios is shown in Fig. 5.1 [80, 81]. AMASE provides a testbed for multi-agent teams of UAS to observe wildfires, allowing for adjustable growth patterns of wildfires, handling UAS dynamics, and providing sensor models. Users interact with the system by sending waypoint commands and receiving state and sensor data. The main goal of the competition was to provide the best estimate of fire zones using only UAS onboard sensor data readings. The authors took 1st place

in the competition, demonstrating algorithms that could consistently find and map wildfires in mountainous terrain, heavy winds, and other anomalies introduced in the competition scenarios. This chapter describes mission and path planning research contributing to our team's first place competition result. Fig. 5.2 shows a high level block diagram of the approach that separated path planning and boundary estimation and Fig. 5.3 overviews the planning methods.

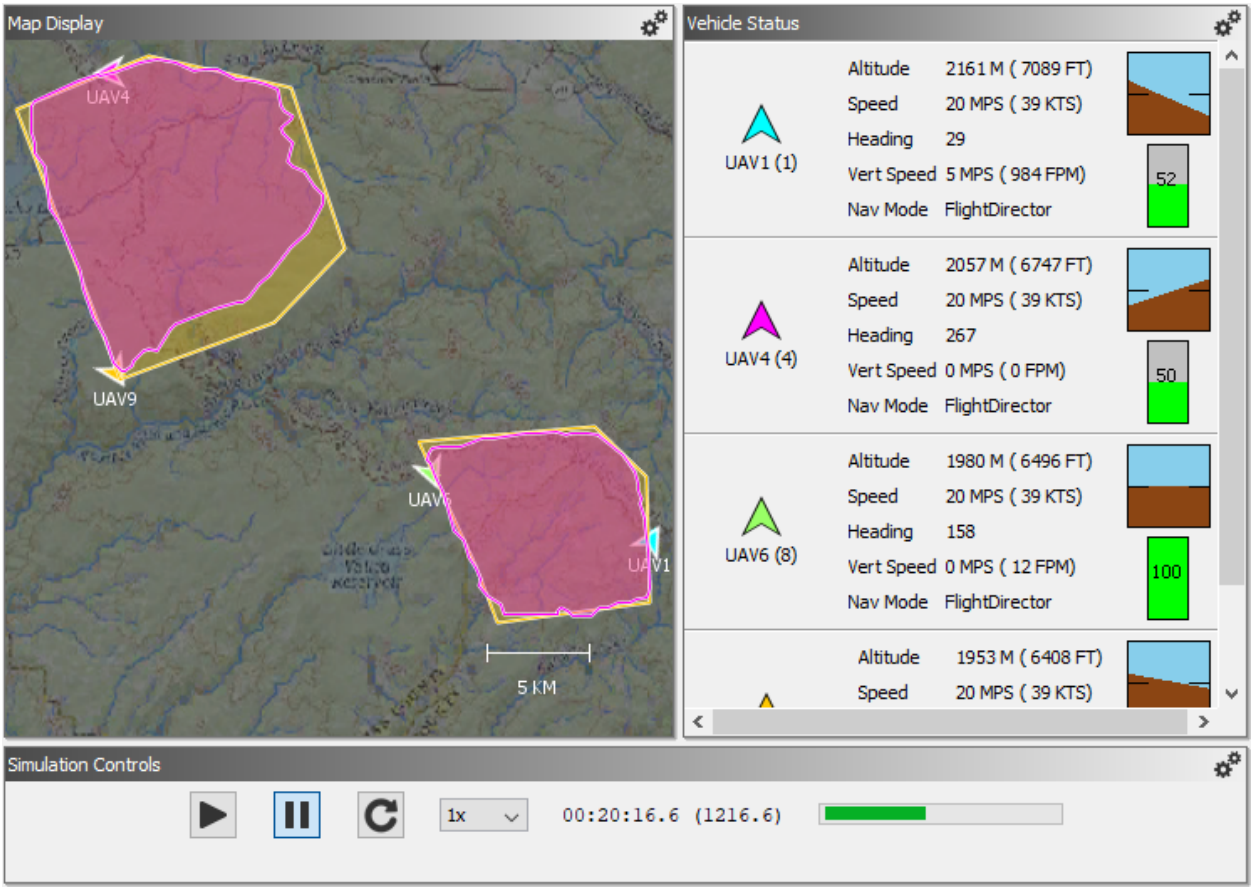



Figure 5.1: AMASE GUI screenshot. Simulation controls (bottom), UAS states (right), and a 2D map (center) depicting UAS ("flying Vs"), fires (yellow polygons), and fire boundary estimates (purple polygons) are shown.

A planning algorithm guides UAS to follow the lateral perimeter of fire boundaries and gather pertinent sensor data for fire boundary estimation. A state machine allocates UAS between exploration and exploitation (line following) tasks. A terrain avoidance altitude path planner guides the UAS to avoid ground impact but remain sufficiently low to collect fire data. The boundary estimation algorithm rapidly constructs two-dimensional, concave polygonal estimates of dynamic boundaries given point-wise observations along the boundary perimeter. For AMASE, point-wise observations come from UAS-hosted fire sensors. The primary contributions of this chapter is a

[1]UAV in the figure is not distinct from UAS

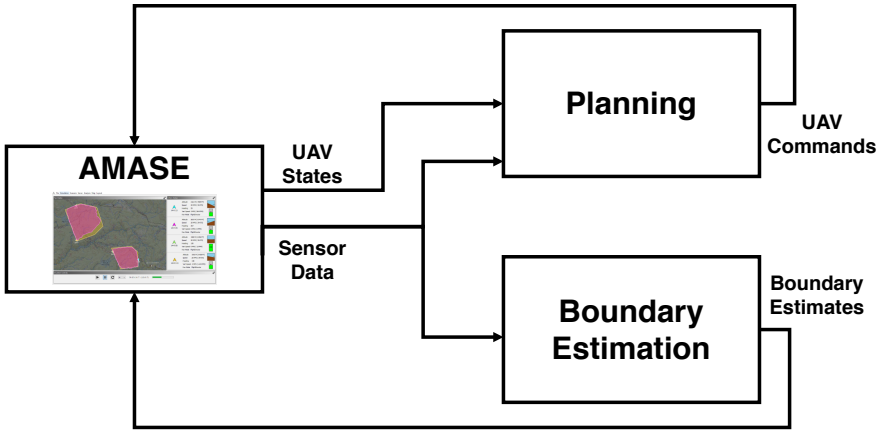



Figure 5.2: Multi-UAS software block diagram. The planner translates UAS states and local fire boundary information to UAS commands that keep UAS safe and support data collection for fire boundary estimation.

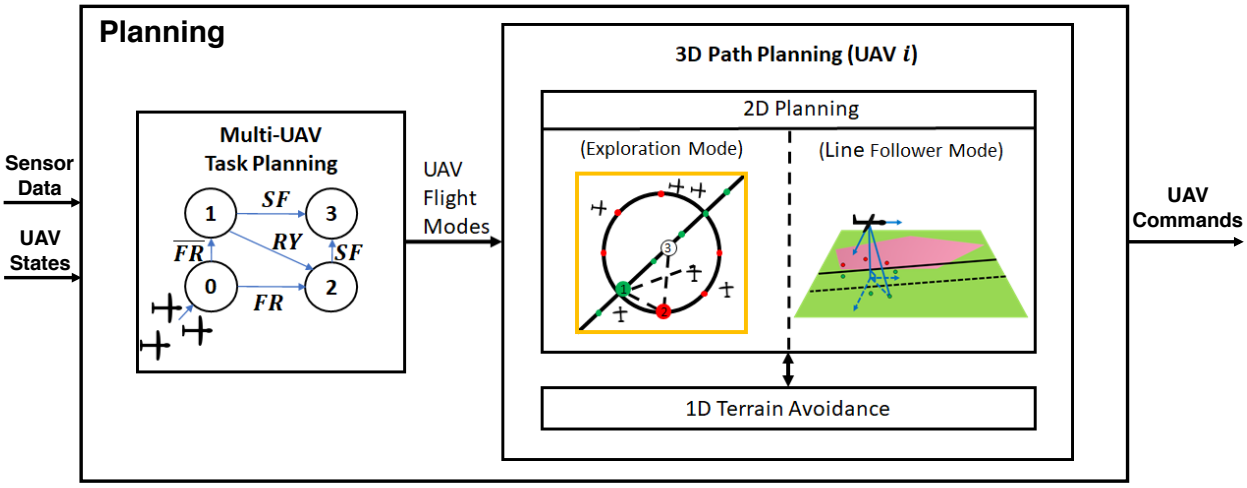



Figure 5.3: Planning module for our Multi-UAS wildfire mapping approach[1]

multi-UAS planning framework to manage UAS team data collection while avoiding hazardous conditions.

The remainder of this chapter is structured as follows. Section 5.2 is a review of related work followed by a problem statement in Section 5.3. Section 5.4 details multi-UAS path planning to find and observe wildfires without being destroyed by the fire or crashing into terrain. Finally, Section 5.5 shows AMASE simulation results for path planning followed by a discussion in Section 5.6 and conclusion in Section 5.7.

## 5.2 Related Work

Mapping a dynamic boundary of interest (e.g. a wildfire) with a team of UAS requires a system with diverse modules. Below, we present recent work on system automation architectures (Sec. 5.2.1), UAS terrain avoidance (Sec. 5.2.2), and coverage path planning (Sec. 5.2.3).

### 5.2.1 Automation System Architectures

Multi-UAS system architectures describe how mission-level and vehicle-level modules are organized, perform computations, and communicate with each other. Ref. [82] presents a classification of multi-UAS architectures with a focus on control and communication. Four main categories are described: 1) Physical coupling, 2) Formation groupings, 3) Decentralized swarms, and 4) Task-level cooperation. Our architecture represents task-level cooperation by making supervision, coordination, mission planning, and task allocation decisions centrally or in a distributed manner.

Distributed methods offer redundancy and scalability [83, 84, 85] and may reduce communication overhead when data is also distributed. Market-based techniques [84, 85] are often used for distributed task allocation, though agents require time to converge on each new allocation. Distributed processing for multi-UAS perception [86, 87] allows individual UAS to gather sensor data, compute a local representation, then share it with their neighbors to augment overall awareness. However, comprehensive data fusion and storage still requires task allocation and communication overhead.

Centralized methods require communication between all agents and a central node. The required computation at that node is significant, and increases as the number of agents increase [40, 88]. For example, Ref. [88] uses operational constraints and UAS capabilities to allocate decision making across the team with reliance on a centralized decision node communicating with each UAS. With AMASE we can assume reliable communication and GPU enabled central processing, so we defined a centralized architecture to manage computing and communication tasks.

### 5.2.2 Terrain Avoidance - 3D Safe Path Planning for Fixed-Wing UAS

For fixed-wing multi-UAS path planning, physically feasible collision-free paths that achieve mission goals must be planned for all UAS in the team. This is complicated by nonholonomic fixed-wing UAS motion constraints for turning radius and finite limits for climb/descent angle and rate. Additionally, surveillance UAS cannot collide with terrain but must remain close enough to the ground to successfully acquire data. A variety of fixed-wing terrain avoidance path planning strategies have been developed.

Geometric path planning strategies based on Dubins paths, Bezier curves, and splines are intuitive and can address nonholonomic path constraints [89, 90, 91]. Dubins paths [89] have been

widely used to rapidly compute minimum-length fixed-wing aircraft trajectories in free airspace [92]. However, geometric methods typically do not consider variable terrain and other obstacles. Sampling based methods offer another avenue and work well when extended to high dimensional search spaces [93, 94, 13]. Ref. [13] uses Bezier curves in conjunction with a sampling-based planner to generate kinematically feasible paths for a fixed-wing UAS, defining the search-space as a rapidly-exploring random tree (RRT). This can generate feasible paths over a complex 3D search space with obstacles, however, its computation time is still too slow for our application. With optimization-based planners, a variety of constraints and cost functions can be considered to fully consider both terrain and vehicle performance [95, 96, 97]. Ref. [95] uses mixed integer linear programming (MILP) with position and speed constraints, UAS collision avoidance constraints, and terrain avoidance constraints for the multi-UAS planning problem. However, MILP computation time required is large, being NP-complete and thus not reliable for real-time applications of any reasonable size. Lastly, discrete planners such as A* are very popular in low dimensional spaces due to fast computation of optimal paths [98, 99, 100, 101]. Ref. [98] uses a two-phase approach. First, A* is run on a large 3D grid to create an obstacle-free, kinematically feasible path. Second, a sampling-based local planner with motion primitives connects coarse path segments. While effective, this approach is also computationally intensive. Several methods reduce computation time by performing 2D A* with added costs from a digital elevation map to effectively plan in 3D [99, 100]. In particular, Ref. [101] pre-processes digital terrain data to generate a surface that is flyable by the UAS in every direction while maintaining a minimum height clearance before 2D A* search is performed. However, over large spaces search is also computationally expensive.

Our method is inspired by the reactionary nature of an automatic ground collision avoidance system (Auto-GCAS) [102] and separation of the 3D path planning problem into a simpler 2D lateral plane path planner with altitude-based terrain avoidance similar to techniques described in [99, 100, 101]. While this approach isn't optimal, it enables fast coverage and path planning to support the 4x real-time performance needed with AMASE. Sections 5.4.3 and 5.4.4 describe our 2D lateral plane planning while Sec. 5.4.5 introduces our altitude-based terrain avoidance method.

### 5.2.3 Multi-Vehicle Coverage Path Planning

Coverage Path Planning (CPP) is the task of planning a path that goes through all points or regions of interest while avoiding obstacles [103]. CPP survey papers, including [104], [103], and [105], describe methods and their applications to ground-based, undersea, and aerial robotic systems. Ref. [106] applies Boustrophedon decomposition to mobile (ground) robots to determine how multiple robots cover a single cell and how robots are allocated among many cells. In Ref. [107] integer programming is used for UAS coverage of a surveillance region. Ref. [108] considers a mission with a team of heterogeneous UAS searching an area of interest with three

steps: determining relative capabilities of each UAS, cooperatively assigning search areas, and generating efficient CPPs for each subregion. Ref. [109] proposes a local dispersion model for global coverage with a team of robots, while Ref. [110] uses Morse-based decomposition to assign UAS to subregions taking into account no-fly zones, environment geometry, and initial/final waypoints. Most of the CPP literature presumes a complete coverage requirement. Our application differs in that it does not require complete coverage, and the solution must be computed quickly and updated in real-time based on collected data. Our CPP method (Sec. 5.4.3) is inspired by related work but addresses these additional challenges. We specify a pre-defined set of waypoints within our rectangular search region and then use the Hungarian Algorithm [111] for initial UAS waypoint assignment. This approach is computationally efficient for real-time settings such as the AFRL Swarm and Search AI Competition.

## 5.3 Problem Statement

An $N_v$ agent fixed-wing UAS team is deployed to a large rectangular area defined by GPS boundary coordinates. This region is known to contain active wildfire(s), but no additional information on wildfire quantity, size, or location(s) is provided. The UAS team must find and accurately map fire boundaries within a fixed time limit assuming a communication link with a central server is always available. Each UAS accurately senses its own state (attitude, position, and derivatives) and hosts an ideal communication system (no latency, no data rate limit). UAS can accurately execute waypoint commands via onboard trajectory following controllers. Each UAS is equipped with a gimballed, range-limited fire hazard sensor that returns accurate binary results for the presence of fire where it is pointed at a rate of 2Hz. There are two hazards the UAS must avoid: collision with mountainous terrain and flying above any wildfire for too long. Each UAS's goal, then, is to fly close enough to the fire to observe it while avoiding the two destructive hazards. UAS also carry limited on-board energy that renders the vehicle unusable if fully depleted. If any failure event occurs (collision, burning, energy depletion) that UAS will become disabled in the AMASE simulation. UAS are considered expendable, however, so while a disabled UAS will no longer provide data, the mission can continue.

Though the number of fires is fixed per scenario, each wildfire is a dynamic polygon that can grow, shrink, and translate throughout a mission scenario. The quantity, size, and location of fires are initially unknown to the UAS and the central planner. Once a fire is found, it must be repeatedly surveyed due to its unknown propagation dynamics. Fire boundary estimates can be centrally computed due to our ideal communication assumption. For this chapter, and the competition, the prime performance metric is the accuracy of the fire boundary estimates as compared to the ground truth boundaries. Since this metric is computed over the total wildfire area amongst all fires, it

captures both the ability to find fires and estimate their individual boundaries.

The multi-UAS planner must safely guide and control team members from their initial locations to find fires and then follow their boundaries to persistently support boundary estimation. Each path must respect UAS kinematic constraints and must keep the UAS sufficiently close to the boundary and terrain surface to support collection of positive (fire) and negative (no-fire/free) points. Each UAS must consistently avoid terrain and fire hazards, must avoid collision with other UAS, and must operate efficiently to maximize endurance. Because the search area is large, sensor range is limited, team size is small, and mission duration is fixed, complete area coverage is impossible. Thus, an efficient sparse exploration UAS team planner is required.

Fire boundaries must be estimated from sparse observations; each is encoded as a hit (positive) or miss (negative) along with observation location and time. We assume all positive and negative observations are accurate and that each fire boundary can be entirely described by its perimeter (i.e., no interior holes). Because data is sparse, significant portions of the fire boundaries are not seen so a sparse data boundary estimation inference engine is required.

UAS navigate inertially with GPS data, and sensor measurements are converted into the local projected Universal Transverse Mercator coordinate system. This provides a local Euclidean reference frame from which distance, shape, and area are minimally distorted.

## 5.4 Multi-UAS Planning

Multi-UAS planners must compute exploration and boundary following paths and assign UAS team members to each. Each UAS locally executes trajectory tracking control and adjusts its altitude as needed to avoid terrain. Fig. 5.2 illustrates our approach in the context of the AMASE simulation environment. The AMASE environment including terrain and fire hazards is further described in Sec. 5.4.1. Our state machine based task planner is summarized in Sec. 5.4.2. Sections 5.4.3 and 5.4.4 describe our 2D path planning methods for exploration and boundary following, respectively, and Sec. 5.4.5 describes terrain avoidance.

### 5.4.1 AMASE Wildfire Mapping Simulation Environment

The AMASE simulation environment was used to test the presented methodologies on a group of fixed-wing UAS equipped with a fire detection sensor [80, 81] [2]. AMASE generates UAS motions over time based on fixed-wing aircraft dynamics and wind disturbances. AMASE models terrain, sensor pointing and measurements, and the dynamic evolution of wildfire boundaries.

[2]Ref. [81] is an open source repository for "openAMASE" which is a different version of AMASE than what was used for the competition and this chapter. openAMASE does not have the wildfire content. The version of AMASE we used is governed by a signed software license agreement.

Our code interacts with AMASE by sending UAS commands and receiving state and sensor data. The simulation treats collision with terrain and wildfires as hazards that disable the UAS. UAS also expend onboard fuel or battery energy during flight as a function of airspeed and climb rate. Recovery zones located sparsely throughout the scenario allowed UAS to replenish energy after remaining within their bounds for 30 seconds.

All scenarios were simulated in a 60km by 60km mountainous region in California. At times, avoiding steep mountains requires the UAS to stop observing fires for some time which provides a realistic test case requiring intermittent wildfire observations. Each scenario simulated 6-12 UAS each carrying a gimballed fire hazard sensor that returns the status (fire point or free point) of the ground location to which the sensor is pointed so long as it is within range. The sensor range limit forces the UAS to remain near the ground and thus requires effective terrain avoidance. If, for example, the range limit was removed or very large, the terrain avoidance problem would become trivial since the UAS could simply fly above all terrain. Additionally, the fire hazard sensor only returns a single data point for each observation at a low rate (2Hz). While a UAS is following a fire, this makes it much more difficult to estimate the local boundary for safe traversal than it would if say a thermal camera provided dense, high rate sensing information. Wind, UAS battery energy constraints, number of wildfires, and wildfire dynamics (translation rate and direction and growth rate) are configurable within AMASE. Every scenario ran for one hour of simulation time with UAS states updated every 0.5 simulation seconds. The AFRL Swarm and Search competition also required each one-hour scenario be run at 4x real-time.

California Digital Terrain Elevation Data (DTED) Level 2 terrain data was obtained from the U.S. Geological Survey (USGS) Earth Explorer website [112]. Data cover a uniform grid matrix of terrain elevation values with post spacing of one arc second (about 30 meters). The keep-in operating zone of 60km by 60km was centered at 39.81° N 121.06° W with maximum and minimum elevations of 2388m and 226m, respectively. The steepest terrain rises about 150m in altitude over 30m of horizontal movement. For all scenarios, the UAS had a maximum climb and descent rate of $\pm$5m/s with planar speeds of 25m/s making it impossible to avoid terrain and maintain observability of the ground with naive reactive altitude guidance. With several point-wise fire and free observations, a polygonal wildfire boundary estimate was computed by our code and sent to the AMASE simulator for scoring. Scores were computed at 20, 40, and 60 minutes into each one-hour scenario. The fire boundary estimate total score is a weighted sum of individual scores:

$$S_i = 3\,\mathrm{FHS}(t_{20})_i + 2\,\mathrm{FHS}(t_{40})_i + \mathrm{FHS}(t_{60})_i \tag{5.1}$$

$$\mathrm{FHS}(t) = \frac{\mathrm{area}\big((\mathrm{PB}_t \cap \mathrm{GT}_t) - (\mathrm{PB}_t - \mathrm{GT}_t)\big)}{\mathrm{area}\big(\mathrm{GT}_t\big)} \tag{5.2}$$

where PB and GT are the predicted and ground truth boundaries at time $t$ respectively, $\mathrm{FHS}(t) \in (-\infty, 1]$ is a Fire Hack Score function which returns 1 for a perfectly matched fire boundary estimate, 0 for no estimate, and potentially negative values for overestimating fire boundaries, and $t_X$ is a timestamp $X$-minutes into the simulation. This scoring is performed for all $i \in [1, N_F]$ fires in the scenario for the three scoring instances $t_{20}$, $t_{40}$, and $t_{60}$.

### 5.4.2 Task Planning

The UAS team is managed by a centralized state machine in constant communication with all UAS. The state machine, shown in Fig. 5.4, offers four flight modes: an initial standby mode, an exploration mode to find wildfires, a "rally" mode to assist another UAS that has found a wildfire, and a line following mode to gather observations along the wildfire perimeter. All vehicles start in the standby flight mode "Assign UAS" (FM0). Since there are no wildfires detected initially, all UAS enter exploration mode (FM1) where UAS are assigned exploration paths that sparsely cover the region and quickly find wildfires as explained in Sec. 5.4.3. If a vehicle observes a wildfire (event "saw fire" or SF), it transitions to line following mode (FM3) to traverse the perimeter of the wildfire and gather observations per Sec. 5.4.4. In addition, the UAS sends a rally signal (RY) to request its $N$ nearest neighbors in FM1 assist in gathering observations of the wildfire. UAS in FM2, "Go to Rally Point", fly straight to the observed fire point and enter FM3 when they observe a fire point. UAS assigned to a wildfire via rallying will follow the fire boundary in either the clockwise (CW) or counter-clockwise (CCW) direction in an alternating fashion to more effectively map the wildfire.

Onboard energy management is handled implicitly by flying all UAS at their most efficient flight speeds. However, recovery zones are ignored because replenishing energy diverts a UAS from finding/mapping fires, and mission time is limited. In fact, a scenario would typically end by the time a UAS finished recovering. Flying without recovery at efficient speeds allows UAS to remain alive for around 80-90% of each scenario. This strategy has benefit because there are no penalties for losing the expendable UAS.

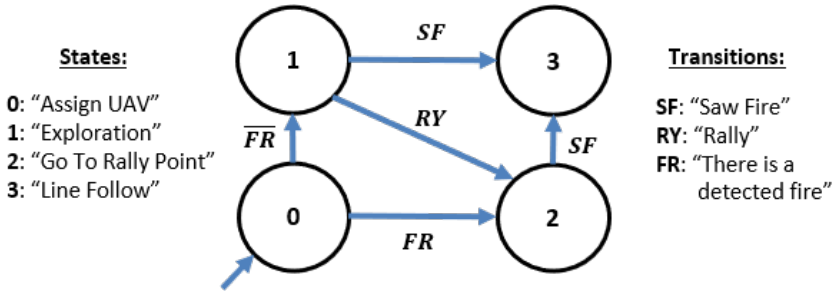



Figure 5.4: Flight modes state machine

### 5.4.3 Exploration Search Pattern

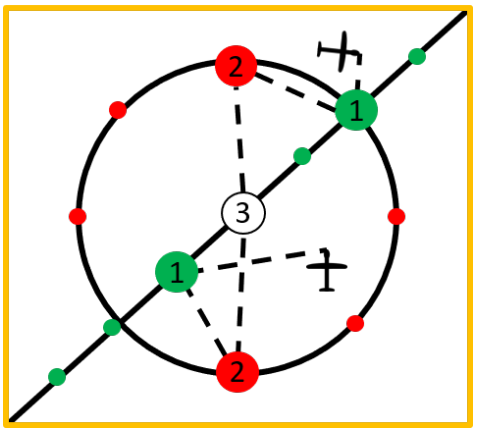
UAS assigned to FM1 are given a sparse-coverage exploration path to quickly find wildfires. We initially considered using complete coverage paths, but found that the expansive search space and limited flight time precluded their use since they would not finish in time nor were complete coverage paths effective at finding fires in the available time. We emphasized finding large-area fires quickly rather than trying to search the entire region. Once a fire was found it was also important for other UAS to be nearby so they could rally to following the fire boundary. Therefore, our manually designed exploration paths bias the UAS toward central region coverage rather than areas near the perimeter. We used a priori knowledge of the keep-in zone to make a specific solution that was effective for competition scenarios and could handle arbitrary numbers and starting locations for UAS and fires.

Fig. 5.5 depicts our method. The UAS travel to a waypoint along the diagonal of the keep-in zone, then to one of six waypoints along a circle, and finally to a waypoint at the center of the region as shown in Fig. 5.5a. The "diagonal waypoints", green in Fig. 5.5a, are $N_v$ linearly spaced point along the diagonal of the keep-in zone: one waypoint for each of the $N_v$ agents. The red "circle waypoints" are computed by creating a circle with radius equal to 0.4 times the side length of the keep-in zone. The six circle waypoints are assigned at 45 degree spacings excluding the diagonal line. When $N_v > 6$, some circle waypoints will be visited by multiple UAS. The Hungarian algorithm (Munkres assignment algorithm) [111] is used to minimize total distance traveled by the group of UAS when assigning diagonal and circle waypoints. Fig. 5.5b shows the exploration paths for a nine agent UAS team in an example AMASE environment.

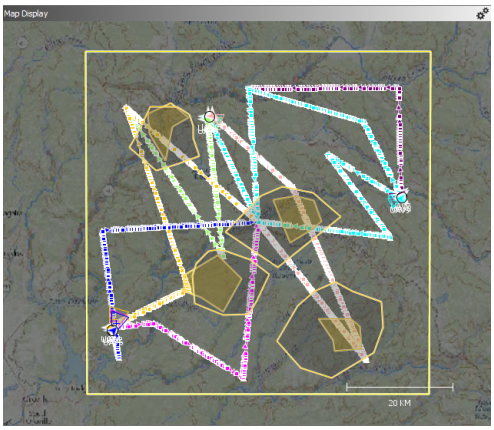
Since our exploration search pattern is calculated from the keep-in zone provided, it will automatically scale to larger scenarios. However, as regions grow, generated paths may not be traversed within the scenario time limit, or fires are found much later, potentially too late to survey. For our 60km by 60km scenarios, the first fire was typically found at around 15min into the mission. If we assume a fire is found on the first straight line path segment of at least one UAS, then the mission time to first fire identification will scale with keep-in zone side length, e.g., doubling side length to 120km by 120km would push the 15min fire ID time to 30min. The exploration search pattern will still be effective for scenario scalings up to 2x-3x particularly if onboard energy stores and/or mission time is increased. We currently assume a square keep-in zone consistent with the AMASE competition; this constraint can be extended to any rectangular zone by using an ellipse instead of a circle to compute waypoints.

Our assumption of a priori knowledge of the keep-in flight zone is consistent with general wildfire response scenarios. Manned and unmanned aircraft would likely work in tandem, coordinated at a high level from a ground control station where keep-in zones are applied to the unmanned aircraft to provide additional safety to manned aircraft and also allow the large area surveillance problem to

be broken down into smaller search sectors, one per UAS team. It should also be noted that since we search within a circle of radius 0.4 times keep-in zone side length, we are implicitly assuming that there will be fires near the center. While this was true for the competition, and it would also be true for a team deployed to a region centered on low-resolution satellite hot/bright spots, if there was a scenario that only had fires at the corners our UAS team would never find them. Additionally, our sparse coverage paths are expecting a few relatively large fires. If there were numerous, small fires in practice more dense sensor information would be required given the same team size, large search area, and mission time constraints.




(**a**) UAS traverse from initial positions to a point along a diagonal of the search space (green), then to a point on the circle shown (red), and finally to the center of the space.




(**b**) Exploration path for a nine agent scenario. Colored paths represent different UAS. As shown, UAS paths cover a large area and intersect every fire for this scenario.

Figure 5.5: Multi-UAS exploration strategy. a) Description of waypoints/paths. b) Generated paths on a scenario.

### 5.4.4 SVM-Based Line Follower

UAS assigned to FM3 are tasked with following the fire boundary. This was achieved with two sub-tasks: 1) Obtain a linear estimate of the fire boundary near the UAS, and 2) Guide the UAS to move tangent to this line at a fixed, orthogonal offset that keeps the UAS clear of fire but still supports fire point sensing. Fig. 5.6 illustrates the method.

For line estimation, we used weighted support vector machine (W-SVM) regression. Given the most recent free and fire points detected by the UAS, W-SVM regression is performed that more heavily values newer samples. W-SVM returns a line approximating the fire boundary near the UAS separating the two classes (free and fire). Our assumption of a linear local boundary (as opposed to a more complex geometry) simplifies the method and reduces computation time for both estimation and guidance.

When the actual boundary is linear and static, a linear estimate can separate free/fire points appropriately. However, when the actual boundary is dynamic, it's possible that old points are no longer on the correct side of the boundary. By weighting newer points more heavily and older points less, the estimate can more effectively follow the actual dynamic boundary. The same can be said about a curved boundary. Our straight line estimate can't follow the curve exactly. However, for the location that is most important (boundary portion closest to the UAS), a linear approximation is sufficient. The newest points are also closest to this area of interest so our W-SVM method is able to follow the curve as new data is collected. For guidance, we used a proportional heading controller (Eq. 5.3) to generate a 2D (horizontal plane) reference state such that the UAS remains a desired perpendicular offset from the line ($d_{des}$) and tracks a reference heading in the direction of the line ($\psi_{line}$). While tracking this heading, the sensor gimbal is guided to sample sensor data near the vehicle. A preset observation sequence of observation offsets from the vehicle considering terrain is executed to generate sufficient data for SVM and overall fire boundary estimation. AASLP (Sec. 5.4.5) is used to plan the altitude to avoid terrain while still being sufficiently close to sense fire points.

$$\psi_{des} = \psi_{line} + K_p \left( d_{des} - d_{line} \right) \tag{5.3}$$

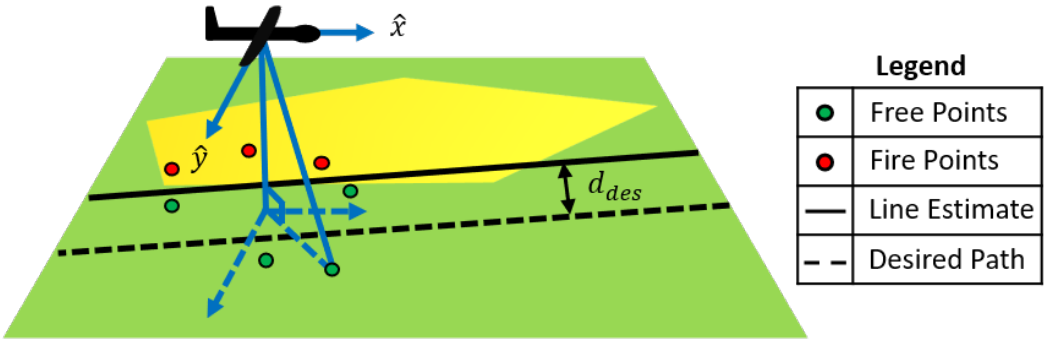



Figure 5.6: 3D line follower. Weighted SVM provides a linear fire boundary estimate with nearby free and fire points and a proportional controller follows this estimate at an offset in 2D with altitude coming from AASLP.

### 5.4.5 Altitude Aware Straight Line Planner (AASLP)

The Altitude Aware Straight Line Planner (AASLP) (Alg. 1) is used to command each UAS's altitude. AASLP prioritizes avoiding terrain while attempting to respect range limits of the onboard fire hazard sensor. In exploration and rally modes, AASLP assumes the UAS is flying in a straight line from starting location $\mathbf{r}_s$ to a goal position $\mathbf{r}_g$ and returns the altitude trajectory needed to maintain flight between $h_{min}$ and $h_{max}$ where $h_{min}$ is the lowest altitude above ground level (AGL)

the UAS should ever fly and $h_{max}$ is the highest AGL altitude the UAS can fly and collect sensor readings. Line following mode behavior is described below. Parameter $h_{min}$ is treated as the stricter constraint since violating this altitude limit could result in a crash. Climbing above $h_{max}$ does not cause an immediate crash though losing observability of the ground makes the UAS "blind" to nearby wildfires that can inflict damage over time. Given a constant forward speed $v$, maximum climb and descent rates $v_{climb} > 0$ and $v_{descent} > 0$, and time step $\Delta t$, we simply estimate $\Delta x = v\Delta t$, $\Delta z^+ = v_{climb}\Delta t$, and $\Delta z^- = v_{descent}\Delta t$. AASLP calculates the required altitude trajectory by assuring forward integration of the altitude over time given a terrain map and climb/descent angle constraints respects $h_{min}$ and $h_{max}$ limits. To meet climb and descent constraints, future terrain clearance requirements on $h_{min}$ are back-propagated so the climb starts in time to avoid future terrain. A similar strategy is followed for descent with terrain clearance $h_{min}$ is prioritized over $h_{max}$ when necessary.

**Algorithm 1** Altitude Aware Straight Line Planner

**procedure** AASLP($\mathbf{r}_s$, $\mathbf{r}_g$, $\Delta z^+$, $\Delta z^-$, $z^0$, $h_{min}$, $h_{max}$, $\Delta x$)
  **Inputs:** Start position $\mathbf{r}_s$, goal position $\mathbf{r}_g$, climb rate $\Delta z^+$, descent rate $\Delta z^-$, start altitude $z^0$, minimum AGL $h_{min}$, maximum AGL $h_{max}$, distance discretization $\Delta x$
  **Output:** An altitude plan, $Z$, for a straight line path from $\mathbf{r}_s$ to $\mathbf{r}_g$
  $\Delta\mathbf{r} \leftarrow \mathbf{r}_g - \mathbf{r}_s$
  $N \leftarrow$ ceil($\frac{|\Delta\mathbf{r}|}{\Delta x}$)+1
  $Z \leftarrow$ zeros(N)
  $z_g \leftarrow$ terrain_height($\mathbf{r}_s$)
  $Z_0 \leftarrow$ constrain($z^0$, $z_g + h_{min}$, $z_g + h_{max}$)
  **for** i $\leftarrow$ 1, N-1 **do**
    $\mathbf{r}_{next} \leftarrow \frac{i}{N-1}\Delta\mathbf{r} + \mathbf{r}_s$
    $z_g \leftarrow$ terrain_height($\mathbf{r}_{next}$)
    $Z_i \leftarrow$ constrain($Z_{i-1}$, $z_g + h_{min}$, $z_g + h_{max}$)
    **if** $Z_i - Z_{i-1} > \Delta z^+$ **then**
      **for** $j \leftarrow i-1, 0$ **do**
        **if** $Z_{j+1} - Z_j > \Delta z^+$ **then**
          $Z_j \leftarrow Z_{j+1} - \Delta z^+$
        **else**
          break
        **end if**
      **end for**
    **else if** $Z_i - Z_{i-1} < -\Delta z^-$ **then**
      $Z_i \leftarrow Z_{i-1} - \Delta z^-$
    **end if**
  **end for**
  **return** $Z$
**end procedure**

Exploration and rally flight modes use the altitude plan from AASLP directly. Line following mode looks ahead over a finite time horizon to compute the next altitude command in a manner analogous to model predictive control. Goal position ($\mathbf{r}_g$) is selected by projecting the vehicle's current position forward in a straight line based on its current heading. Straight lines at $\pm 5^\circ$ offset from the current heading are also considered to account for future heading adjustments; the maximum altitude from all three lines is used to select the line following altitude.

## 5.5 Results

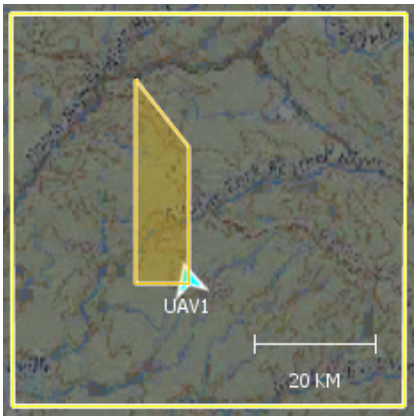
This section describes results obtained from a series of AMASE simulations. Line estimation and terrain avoidance planning results were generated on a laptop with an Intel core i7-9750H processor. Sec. 5.5.1 evaluates the weighted SVM local boundary line estimator while Sec. 5.5.2 compares our AASLP terrain avoidance algorithm against a baseline A* algorithm.

### 5.5.1 Local Boundary Line Estimation

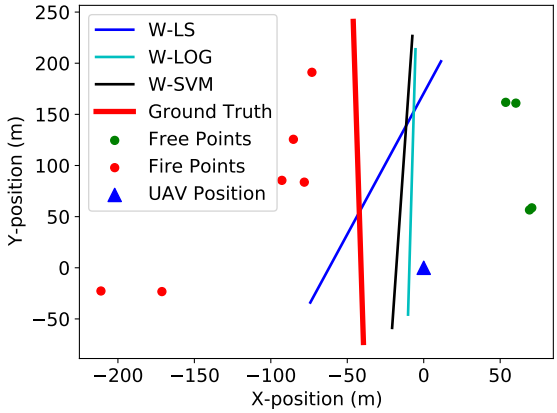
Local boundary line estimation was evaluated on three AMASE scenarios with distinct fire boundary behaviors. Data gathered from the three scenarios are used to compare local line estimates from weighted SVM against estimates from baseline weighted least squares and weighted logistic regression methods. Our combined 3D line following algorithm was used to guide a single UAS in the AMASE simulation to obtain fire and free points with associated observation times. We post-processed the data to provide the three different line estimation methods the last 15s of data for each point in time. Fig. 5.7a shows scenario 1 which has a static fire zone with a straight boundary, a river crossing (large terrain transition with missing data), an acute left turn, and an obtuse left turn. Scenario 2 has the same fire boundary now with constant translation speed of 5m/s to the East. Scenario 3 is the same region with constant fire translation speed of 5m/s to the West. Boundaries update in distinct steps every 10 seconds (e.g., 50m jumps for 5m/s travel speed). The UAS flew at 20m/s and received a new sensor reading every 0.5s. Scenarios 1, 2, and 3 had $3,357$, $826$, and $743$ test instances respectively. Scenario 1 had 3,357 test instances; for this case the UAS ended at the beginning of the third boundary edge, past the obtuse angle. Scenarios 2 and 3 had 826 and 743 test instances, respectively. In these cases the UAS only traversed a portion of the first straight edge.

Our weighted support vector machine (W-SVM) method was compared with weighted least squares (W-LS) and weighted logistic regression (W-LOG). Values were weighted linearly based on time (e.g., a current observation has weight 1, a 15s old data point has weight 0.5) for all methods. Several hyperparameters for each algorithm were considered and the best were used for testing. The regularization parameter for W-SVM was set to $C = 100$; a standard scaling was applied to W-SVM input; the regularization parameter for W-LOG was set to $C = 1$. All methods were

run with `scikit-learn` in Python. Fig. 5.7b shows line estimates from the three methods in scenario 1 at 5.25 seconds.




(**a**) Scenario 1 initial conditions




(**b**) Scenario 1 line estimate at t=5.25 seconds

Figure 5.7: Local boundary estimation results for scenario 1 with a static boundary.

Three metrics were used for evaluation: Line Heading Error, Forward Alignment Error, and Computation Time. Line Heading Error is the error in degrees between the ground truth line heading and the estimated line heading. Forward Alignment Error is the average distance between ten points on the ground truth line and ten points on the estimated line obtained by projecting the original points normally from the ground truth line onto the estimated line. Ground truth lines points are sampled in the forward direction every 10m starting at the nearest point to the UAS. Forward Alignment Error is limited to a maximum of 1000m since orthogonal line estimates could produce extremely large errors that skew statistics. Computation Time is algorithm execution time in ms. Lower is better for all three metrics. Table 5.1 shows aggregated results. W-LOG had the best Line Heading Error (9.01°) and Forward Alignment Error (21.59m), however, it also took the longest to compute (5.78ms on average). W-LS was fastest to compute (0.6ms average) but had the worst performance with a Line Heading Error of 14.26° and a Forward Alignment Error of 52.65m. Our method, W-SVM came close in performance to W-LOG (9.26° Line Heading Error and 24.85m Forward Alignment Error) and also came very close in computation time to W-LS with an average of 1.05ms.

### 5.5.2 Terrain Avoidance

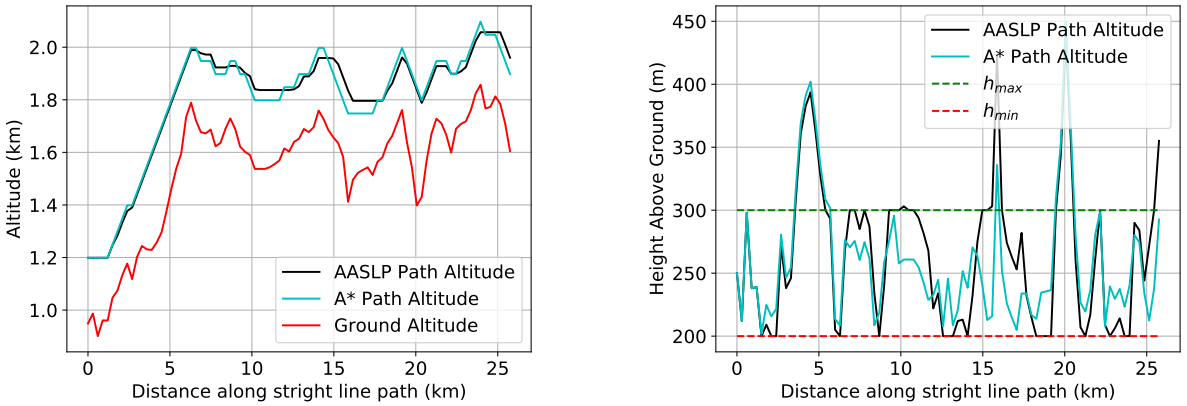
This section evaluates the performance of our altitude aware straight line planner (AASLP) as compared with a baseline A* algorithm. We used the coarse exploration waypoints from the scenario depicted in Fig. 5.5b as our test data set. Each test instance is a straight line path in

Table 5.1: Local boundary estimation results. Our method, W-SVM, provided estimates nearly as accurate as W-LOG at 5x faster. W-LS was fast but provided poor estimates.

| | Line Heading Error (°) | | | Forward Alignment Error (m) | | | Computation Time (ms) | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | $\mu$ | $\sigma$ | $MAX$ | $\mu$ | $\sigma$ | $MAX$ | $\mu$ | $\sigma$ | $MAX$ |
| W-LS | 14.26 | 11.54 | 86.74 | 52.65 | 84.23 | 1000.00 | **0.60** | 0.05 | 1.47 |
| W-LOG | **9.01** | 8.99 | 89.10 | **21.59** | 48.68 | 1000.00 | 5.78 | 0.61 | 10.48 |
| W-SVM | 9.26 | 9.35 | 88.59 | 24.85 | 63.81 | 1000.00 | 1.05 | 0.19 | 5.12 |

2D space, and each tested algorithm must plan an altitude path along this line that avoids terrain, respects vehicle performance limits, and attempts to stay below a maximum above ground level altitude. There were 27 total test instances, but only 19 had viable solutions. Both AASLP and A* failed on the same eight cases because UAS climb rate was insufficient to clear the terrain. Normally, AASLP will always return a safe, feasible path in the discretized space, with the caveat that the desired initial altitude may differ from the actual initial altitude if it is either out of the $h_{min} - h_{max}$ range initially or no collision-free path is possible from the initial altitude. For the competition this was a useful feature because it allowed us to perform additional 3D climbing maneuvers, but for the results presented here any time there was a discrepancy between actual and planned initial altitude we counted this as a failure. For both algorithms, $\Delta t$ was chosen as 10s so that the discretization along the straight line path $\Delta x$ was about 300m. In general, setting $\Delta x$ equal to terrain data resolution (about 30m for USGS) is the smallest practical value since all terrain grids will be considered. However, in the competition and these results, 300m was used since it reduces computation time while only introducing a small chance that dangerous terrain would be missed between samples. For A*, a branching factor of three was used (max climb, straight, max descent). Additionally, the UAS was assumed to fly at 30m/s with a maximum climb rate of 5 m/s ($\Delta z^+ \approx$ 50m) and a maximum descent rate of 5m/s ($\Delta z^- \approx 50$m). The minimum AGL altitude ($h_{min}$) was chosen as 200m, and the maximum AGL altitude ($h_{max}$) was chosen as 300m. The UAS always started at 250m AGL. Both A* and AASLP were run in Python.

Fig. 5.8 shows planned altitude time histories from both algorithms executing on one of the test cases. The UAS's AGL altitude never goes below $h_{min}$ to ensure no collisions with terrain. However, sometimes the steep terrain drops faster than the UAS's descent rate causing violations of $h_{max}$. Table 5.2 shows results from all case studies. A* performed slightly better with an average distance above $h_{max}$ of 14.82m vs. 18.71m with AASLP but A* required approximately 100 times longer computation time (7.22ms vs. 0.09ms).




(**a**) MSL altitude trajectory

(**b**) AGL altitude trajectory

Figure 5.8: Altitude plan example with complex terrain. AASLP meets $h_{min}$ but cannot always meet $h_{max}$. A* is slower but has slightly better performance in meeting $h_{max}$.

Table 5.2: Terrain avoidance planning algorithm benchmark results.

| | Distance Above $h_{max}$ (m) | | | Computation Time (ms) | | |
|---|---|---|---|---|---|---|
| Method | $\mu$ | $\sigma$ | $MAX$ | $\mu$ | $\sigma$ | $MAX$ |
| AASLP | 18.71 | 54.01 | 242.68 | **0.09** | 0.03 | **0.14** |
| A* | **14.82** | 43.50 | 181.29 | 7.22 | 5.98 | 25.08 |

## 5.6 Discussion

Multi-UAS Planning was developed to specifically support the fire boundary estimation mission goal. Speed, safety, and optimality are important planning metrics. Execution speed is important because updates were required in real-time. Additionally, boundary estimation relied on there being operational UAS collecting sensor data, which requires a focus on UAS safety even though small UAS are typically considered expendable. Gathering useful sensor measurements of the fire is of course also important. Our local boundary line estimation and terrain avoidance algorithms provided reasonable solutions with significantly less computation time than compared baselines. In line estimation, our W-SVM method was more than five times faster than W-LOG with comparable accuracy. Our method was also significantly more accurate than W-LS with a marginal increase in computation time. For terrain avoidance path planning, our AASLP planner was more than 50 times faster than A* with less than a four meter difference in altitude solutions. Overall, results show our practical planners methods provide safe real-time solutions. Of course, our dynamic boundary estimation domain was quite specific so extensions would likely be required for other multi-UAS surveillance missions.

## 5.7 Conclusion

This chapter described planning and fire boundary estimation techniques enabling a team of UAS to find and map fire boundaries in a large complex terrain region. UAS behaviors are governed by a state machine supported by exploration, rally, and line-following planning and guidance laws.

At a system level, AMASE simulations showed our integrated multi-UAS planning and boundary mapping system performs well despite a large operating area, complex terrain, wind disturbances, and moving fire boundaries. The team of UAS rapidly found fire boundaries, rallied around them, and successfully followed their outlines. Terrain and fire hazards were avoided while UAS were sufficient close for point sampling. Our methods were further validated when the authors placed 1st in the 2019 AFRL Swarm and Search AI Challenge.

While the AMASE environment focused on multi-UAS fire boundary characterization, the proposed planning and boundary mapping methods may be applied to any problem domain that requires multiple agents to identify and map a complex boundary from point observations. Our method only requires models for point-wise observations and ground truth polygon estimates for training the neural network. Future work is recommended to explore and potentially optimize over the suite of possible large-area exploration patterns, to incorporate more complex sensor data into boundary estimation, e.g., from thermal imagers, and to decentralize planning and local boundary estimation computations for cases when a continuous global communication link cannot be maintained.

This chapter developed a multi-UAS planning framework to manage UAS team data collection while avoiding hazardous conditions in wildfire mapping. The 3D planning problem was separated into 2D exploration and line following planning modes assigned via a state machine combined with a 1D terrain avoidance planner. Methods were computationally efficient to support rapid avoidance of terrain and fires.

# CHAPTER 6

# Deformable Formation Path Planning

This chapter investigates continuum deformation path planning strategies. We separate the planning problem into a low-level guidance problem (Sec. 6.1) and a high level team planning problem (Sec. 6.2). The team planner assumes there is a static, known environment and generates a reference trajectory for the team that can translate, rotate, and deform through space while avoiding obstacles and inter-agent collision. The guidance module follows this trajectory but additionally must deal with sudden vehicle failures and pop-up obstacles. It must be reactive to break the formation safely to avoid pop-up hazards.

## 6.1 Low-Level Vehicle Guidance

### 6.1.1 Introduction

Formation flying offers significant advantages over individual vehicles including reduced planning computation time, improved sensor coverage, and simplicity through abstraction. However, when there are sudden vehicle failures or pop-up obstacles, the geometric rigidity of a formation won't allow the team to easily avoid all hazards. To address this challenge, we propose a novel safety recovery algorithm for a multi-quadrotor system (MQS) and experimentally validate the approach in simulation and quadrotor flight experiments. We consider quadrotors in a MQS as a finite number of particles in an ideal fluid flow pattern. By classifying quadrotors as healthy and failed agents, we consider failed quadrotors as singularity points of the fluid flow. We then define desired trajectories of the healthy agents along flow streamlines so that the failed quadrotors are safely partitioned outside the motion space (see Figure 6.1). We enclose the failed quadrotors with virtual obstacles to define no-fly zones. Because we treat these obstacles as singularity points that are excluded from the motion space, our proposed safety-recovery approach is called *containment exclusion mode (CEM)* throughout this chapter.

The idea of using potential fields for real-time path planning was originally proposed for robot arms [113]. An attractive potential is placed at the goal, repulsive potentials on obstacles, and gradient descent is used to plan a path. This can be called a navigation function. This is a useful

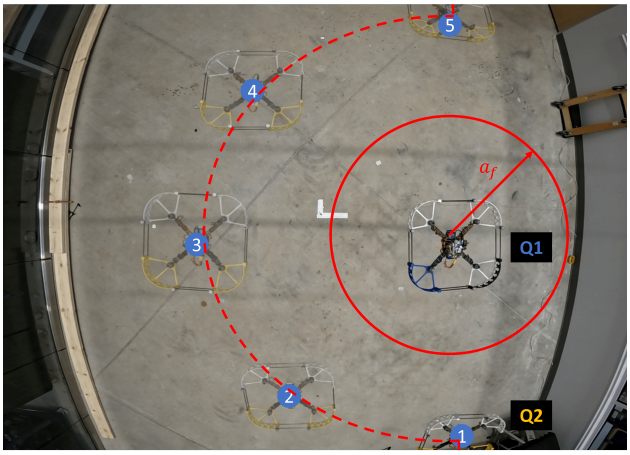



Figure 6.1: Q1 fails and stays in place within the solid red circle. Q2 uses the CEM navigation algorithm to smoothly avoid the failed vehicle and travels along the dashed red path. The blue circles indicate five time points for Q2.

approach, however it has issues of local minima. The randomized potential field approach attempts to get around this by executing a random walk whenever the planner gets stuck in a local minimum, but still provides no guarantee of planning completeness [114]. For harmonic functions subject to Laplace's equations, a navigation function can be constructed that only has saddle points as local minima such that they can be easily escaped [115]. This model has been applied to robot formations, but with explicit models of repulsive functions between robots, one goal sink node, and computation time on the order of one second. However, the scenario considered in this section needs an algorithm to quickly respond to an abrupt failure on the order of milliseconds.

Compared to the existing literature, this work offers the following contributions:

1. We propose a real-time, computationally efficient CEM navigation algorithm that dynamically modifies quadrotor sliding speed to maintain a desired maximum speed for all quadrotors in the MQS.
2. We formally specify safety and collision avoidance constraints by providing a CEM safety theorem considering a single obstacle.
3. We experimentally evaluate the CEM navigation algorithm using a team of quadrotors in formation to validate our CEM safety theorem.

This chapter is organized as follows. Sec. 6.1.2 defines the problem we solve with the approach in Sec. 6.1.3. Safety analysis is considered in Sec. 6.1.4 and simulation and experimental results to validate the approach are provided in Sec. 6.1.5. Conclusions and future work are described in Sec. 6.1.6.

### 6.1.2 Problem Statement

Consider $N_V$ vehicles flying in a deformable formation, following a global path when at $t_0$, $N_f$ abrupt vehicle failures occurs. It is assumed that these failed vehicles hover in place (e.g., a predictable behavior given lost link failure) and are enclosed by virtual cylindrical obstacles of radius $a_f$ at $z_f = (x_f, y_f)$. The remaining $N_h$ healthy vehicles must safely avoid these failed agents in real-time.

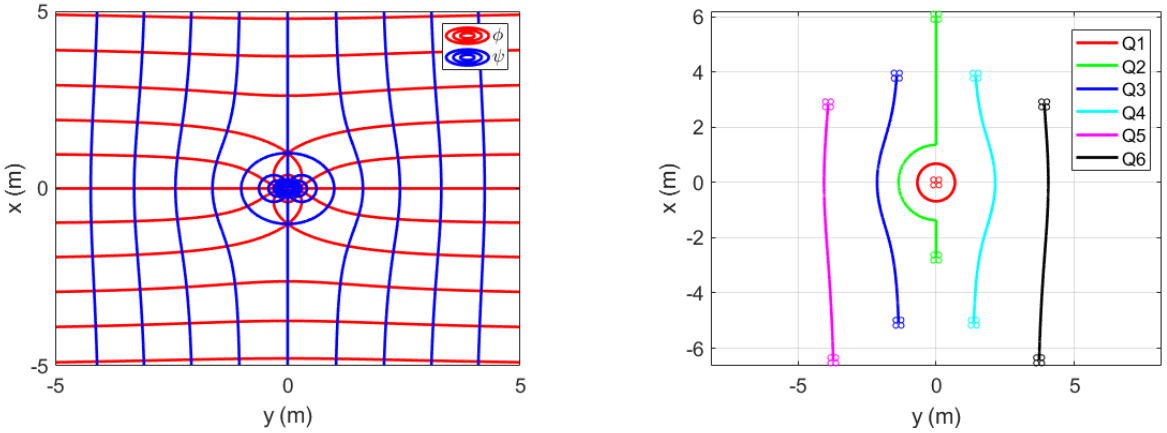
### 6.1.3 Approach

An ideal fluid flow pattern is constructed around the failed agents defined by functions

$$\phi(x,y) = \sum_{f\in\mathcal{F}} \left( \frac{(x-x_f)\left((x-x_f)^2 + (y-y_f)^2 + a_p^2\right)}{(x-x_f)^2 + (y-y_f)^2} \right) \tag{6.1a}$$

$$\psi(x,y) = \sum_{f\in\mathcal{F}} \left( \frac{(y-y_f)\left((x-x_f)^2 + (y-y_f)^2 - a_p^2\right)}{(x-x_f)^2 + (y-y_f)^2} \right) \tag{6.1b}$$

where $\phi$ is the potential function, $\psi$ is the stream function, $a_p = a_f + \delta_g + \epsilon$ is the planned exclusion radius, $\delta_g$ is healthy vehicle controller error, and $\epsilon$ is vehicle radius. Figure 6.2a shows an example of the fluid flow pattern. Note that $\phi$ and $\psi$ satisfy Laplace's equations ($\nabla^2\psi = 0, \nabla^2\phi = 0$). Healthy agents then travel along constant streamlines ($\psi((x_i(t_0), y_i(t_0))) = \psi_{i,0}, \forall i \in \mathcal{H}$) at equal, but dynamic $\dot{\phi}$ to safety. This method is called Containment Exclusion Method (CEM) and is accomplished with a real-time CEM navigation algorithm.




(**a**) $\phi$, $\psi$ Contours. $a_p = 1$, $x_f = y_f = 0$.

(**b**) Six vehicle test 2D plot

Figure 6.2: CEM navigation through ideal fluid flow pattern.

**CEM Navigation Algorithm** Algorithm 2 guides agents along constant $\psi$ streamlines in the fluid

flow pattern. This is performed in real-time such that the "$t$ for loop" (line 12) is run every $\Delta T$ seconds a total of $m$ times. $\Delta T = 10ms$ (100Hz) in this paper. To minimize computational overhead, this method acts as a navigation function, calculating the immediate command for each agent to avoid collisions and then incrementing the trajectory in $\phi$ on each subsequent loop.

**Algorithm 2** CEM Navigation

1: **procedure** CEM($\mathcal{F}$, $m$, $v_{des}$, $\mathbf{r}$)
2: **Inputs:** Failed agents $\mathcal{F}$, number of timesteps $m$, desired speed $v_{des}$, healthy vehicle positions $\mathbf{r}$
3: **Output:** Trajectory for each vehicle via sendCommands()
4:
5: $\mathbf{r}_d \leftarrow \mathbf{r}$
6: $\Delta\phi \leftarrow v_{des} * \Delta T$
7: **for** i $\leftarrow$ 1, $N_h$ **do**
8: $\phi[i] \leftarrow \phi(\mathbf{r}[i], \mathcal{F})$ ▷ Eq. 6.1a
9: $\psi[i] \leftarrow \psi(\mathbf{r}[i], \mathcal{F})$ ▷ Eq. 6.1b
10: **end for**
11:
12: **for** t $\leftarrow$ 1, m **do**
13: $\phi_{last} \leftarrow \phi$
14: **for** k $\leftarrow$ 1, $N_K$ **do**
15: **for** i $\leftarrow$ 1, $N_h$ **do**
16: $\phi[i] \leftarrow \phi_{last}[i] + \Delta\phi$
17: $\mathbf{r}_d[i], \dot{\mathbf{r}}_d[i] \leftarrow$ CalcXY($\phi[i]$,$\psi[i]$,$\mathbf{r}_d[i]$)
18: **end for**
19: $v_{max} \leftarrow \max_{i \in N_h} ||\dot{\mathbf{r}}_d[i]||$
20: $\Delta\phi \leftarrow \Delta\phi * v_{des}/v_{max}$
21: **end for**
22: sendCommands($\mathbf{r}_d$,$\dot{\mathbf{r}}_d$)
23: sleep($\Delta T$)
24: **end for**
25: **end procedure**

$\phi$ and $\psi$ values for each agent are initialized to their starting positions. Also, sliding speed $\Delta\phi$ is set to match the desired vehicle velocities for the non-distorted case. At every time step, $\phi$ is increased by $\Delta\phi$ and the resulting position and velocity for each vehicle is calculated using `CalcXY` (Alg. 3). Maximum velocity $v_{max}$ is then used to update $\Delta\phi$ for the next iteration to more closely match $v_{des}$ (Line 20). This update step assumes $v_{max} = C_g \Delta\phi_{last}$ and wants to calculate $\Delta\phi_{next}$ such that $v_{des} = C_g \Delta\phi_{next}$ where $C_g$ is a constant assumed to encapsulate the gradient for the small step sizes. Therefore, $\Delta\phi_{next} = \Delta\phi_{last} \frac{v_{des}}{v_{max}}$ should calculate the proper $\Delta\phi$ to obtain $v_{des}$. This is executed $N_K$ times. If the assumption that $C_g$ is constant is reasonable, then $N_K = 2$ is a good selection which effectively plans twice per time step, first to calculate $\Delta\phi$ and second to

calculate the trajectories using $\Delta\phi$. $N_K = 2$ was used in this paper for the main results and varied for additional results to see its effects. After the $N_K$ loop, commands are output for that time step and the process repeats for the next time step.

Alg. 3 calculates the position given $\psi$, $\phi$ using a gradient descent inspired approach. On each iteration ($N_i = 20$ in this paper), the current estimate is used to calculate the associated $\phi$ and $\psi$ and then the estimate is updated in the direction of decreasing error via the inverse of the Jacobian matrix, defined as:

$$\mathbf{J}(x, y, \mathcal{F}) = \begin{bmatrix} \frac{\partial \phi}{\partial x} & \frac{\partial \phi}{\partial y} \\ \frac{\partial \psi}{\partial x} & \frac{\partial \psi}{\partial y} \end{bmatrix}. \tag{6.2}$$

$\mathbf{r}_{noise}$ is added to the update step to escape saddle points on the edge of the exclusion zone as:

$$\mathbf{r}_{noise} = \begin{cases} |\mathcal{N}(0,\sigma)|\hat{\mathbf{r}}_a + \mathcal{N}(0,\sigma)\hat{\mathbf{r}}_b & \text{if } ||\mathbf{r} - \mathbf{r}_f|| \leq a_p \\ \mathbf{0}, & \text{else} \end{cases} \tag{6.3}$$

where $\mathcal{N}(0,\sigma)$ generates Gaussian noise with zero-mean and standard deviation $\sigma$, $\hat{\mathbf{r}}_a = \frac{\mathbf{r}-\mathbf{r}_f}{||\mathbf{r}-\mathbf{r}_f||}$ is the direction away from the exclusion zone, and $\hat{\mathbf{r}}_b = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \hat{\mathbf{r}}_a$ is perpendicular to $\hat{\mathbf{r}}_a$. $\sigma = 1$mm was used in this paper. This procedure ensures that commanded positions never violate the planned exclusion radius $a_p$. Lastly, the first-order difference equation is used to calculate the velocity.

**Algorithm 3** CalcXY

**procedure** CALCXY($\phi$,$\psi$,$\mathbf{r}$)
  $\mathbf{r}_{-1} \leftarrow \mathbf{r}$
  **for** i $\leftarrow$ 1, $N_i$ **do**
    $\phi_1 \leftarrow \phi(\mathbf{r}, \mathcal{F})$ ▷ Eq. 6.1a
    $\psi_1 \leftarrow \psi(\mathbf{r}, \mathcal{F})$ ▷ Eq. 6.1b
    $J \leftarrow J(\mathbf{r}, \mathcal{F})$ ▷ Eq. 6.2
    $\mathbf{r} \leftarrow \mathbf{r} - J^{-1} \begin{bmatrix} \phi_1 - \phi \\ \psi_1 - \psi \end{bmatrix} + \mathbf{r}_{noise}$ ▷ Eq. 6.3
  **end for**
  $\dot{\mathbf{r}} \leftarrow (\mathbf{r} - \mathbf{r}_{-1})/\Delta T$
  **return** $\mathbf{r}, \dot{\mathbf{r}}$
**end procedure**

### 6.1.4 Safety Analysis

Collision avoidance between healthy agents and obstacles is assured via the construction of the ideal fluid flow pattern. The planned exclusion radius for obstacles is $\delta_g + \epsilon$ larger than the actual

exclusion radius. Therefore, if no agents are initialized inside the planned exclusion radius, the navigation function will never take them closer than $\delta_g + \epsilon$ which guarantees safety.

Inter-agent collision avoidance is more complex as it involves distortion in the $x - y$ plane along constant $\psi$ streamlines. Theorem 4 provides a tight bound on initial formation distances for a collision avoidance guarantee condition for CEM but it works when there exists a single failed agent in the $x - y$ plane.

**Theorem 4.** *Inter-agent collision between every agent pair is avoided if each minimum separation distance $d_{min,0}$ at reference time $t_0$, when the failed agent appears, satisfies the following condition:*

$$d_{min,0} = \min_{i,j,\ i \neq j} \sqrt{(x_{i,0} - x_{j,0})^2 + (y_{i,0} - y_{j,0})^2} \geq 2(\delta_g + \epsilon) + a_p. \tag{6.4}$$

*Proof.* Inter-agent collision between agents is avoided if $d_{min}(t)$, defined by

$$d_{min}(t) = \min_{\substack{\text{i,j} \\ \text{i}\neq\text{j}}} \sqrt{(x_i(t) - x_j(t))^2 + (y_i(t) - y_j(t))^2} \qquad \forall t, \tag{6.5}$$

satisfies the following condition:

$$d_{min}(t) \geq 2(\delta_g + \epsilon), \qquad \forall t.$$

When a single failure exists in the $x - y$ plane, $\psi(x, y) = 0$ wraps the failed agent by a circle of radius $a_p$ with the center positioned at $(x_f, y_f)$ (see Eq. (6.1b), Figure 6.2a). The maximum contraction of the distance between two arbitrary recovery paths $\psi(x, y) = \psi_{i,0}$ and $\psi(x, y) = \psi_{j,0}$ is less than $a_p$, and it occurs at $x = x_f$. So, $d_{min}(t) \geq d_{min,0} - a_p = 2(\epsilon + \delta_g)\,\forall t$. Therefore, inter-agent collision between every two agent pair is avoided if the minimum separation distance $d_{min,0}$, obtained on the configuration of the healthy agents at reference time $t_0$, satisfies inequality (6.4).

□

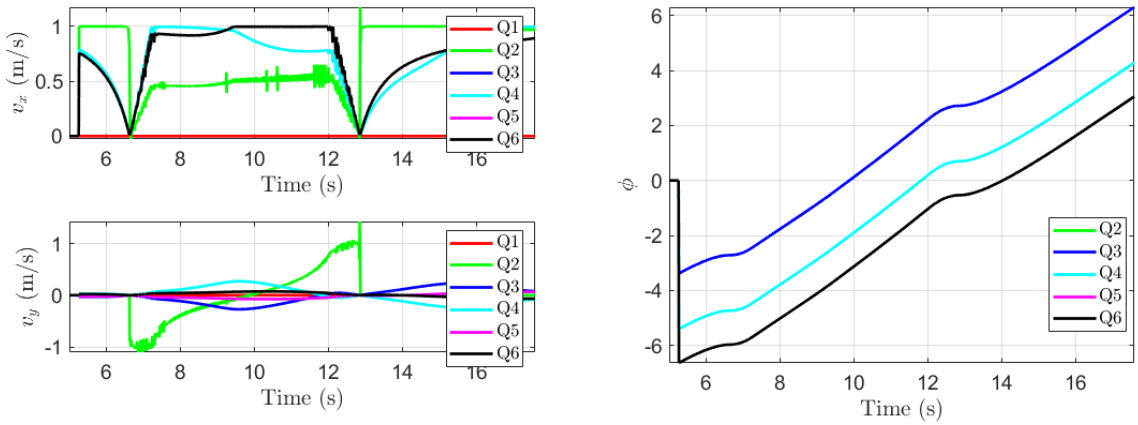
### 6.1.5 Results

The experimental system described in Chapter 2 was used to demonstrate the methods in this section. The minimum initial formation distance between any two vehicles is $d_{min} = 2(\epsilon + \delta_g) + a_p$. Additionally, when a failure occurs and CEM mode is activated we assume that the failed agent will remain within $\delta_g$ of its setpoint so $a_f = \epsilon + \delta_g$. Therefore, $a_p = 1.36m$ and $d_{min} = 2.72m$.

**CEM Navigation Validation with Six Virtual Vehicles** For this test, the ground control station computer was configured to run exactly as described in the experimental setup section, except that

only the setpoints were logged and no vehicles were flying. This was done to validate that the navigation algorithm was able to generate safe trajectories while also maintaining a desired velocity along constant streamlines and operating within runtime constraints.

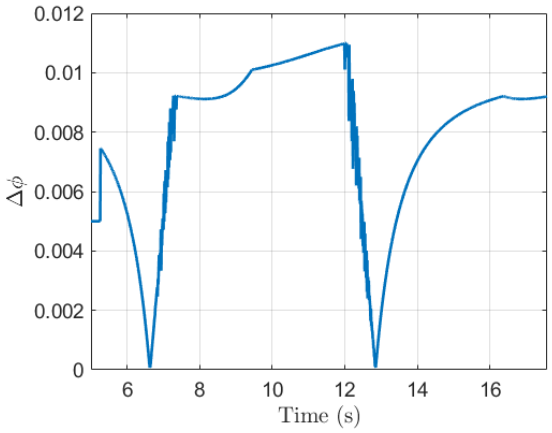
Figure 6.2b shows the formation and trajectory followed. The formation is similar to [116] with a leading triangle and interior agents. The initial formation has agents with a minimum separation of 2.72m. The vehicles begin in a six agent formation on the bottom before Q1 fails. This activates the CEM navigation algorithm. An initial ($\phi$, $\psi$) coordinate is calculated for each vehicle and then on each iteration $\phi$ is advanced in a dynamic fashion to track a 1.0 m/s desired maximum velocity. Figure 6.3a shows the commanded velocities. This shows each component of 2D velocity while the norm at each step was the quantity being tracked. Figure 6.3b shows the values of $\phi$ advancing while Figure 6.4a shows how $\Delta\phi$ is changed to maintain the specified velocity. Notice how at about 6s and 13s $\Delta\phi$ gets very small. This coincides with Q2 traversing saddle points at each end of the exclusion zone. The Jacobian is large here which causes small changes in $\phi$ to produce a large change in the $x$-$y$ plane (i.e. Q2 moves fast while all other vehicles move slowly). Additionally, Figure 6.4b shows the distance between nearest agents. Every agent pair maintains at least $2(\delta_g + \epsilon)$ separation.



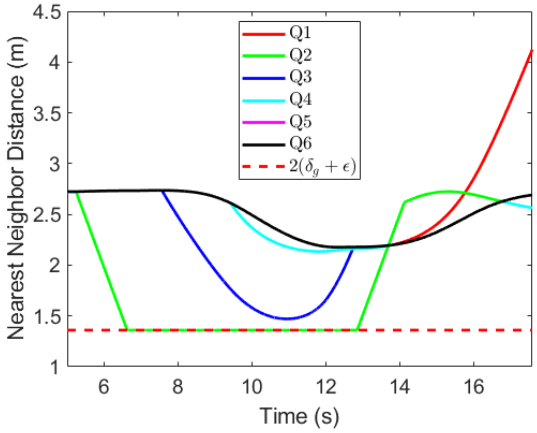

(**a**) Velocities (**b**) $\phi$

Figure 6.3: Six vehicle test velocities and $\phi$

To explore the effectiveness of the velocity tracking aspect of our navigation algorithm the six vehicle flight was repeated while the value for $N_K$ was varied in the range $\{1,2,5\}$. Figure 6.5a shows the maximum velocity of any agent for the various values of $N_K$ during one of the two saddle points that cause the most difficulty. $N_K$=1 reached a 2D speed of 2.4 m/s, much higher than the 1.0 m/s setpoint. Speed error with $N_K$=2 was significant lower, while $N_K$=5 performed a bit better. However, increasing the value of $N_K$ essentially replans with a new $\Delta\phi$ $N_K$ times each iteration. This has a runtime cost which can be seen in Figure 6.5b. Runtime is proportional to the value of



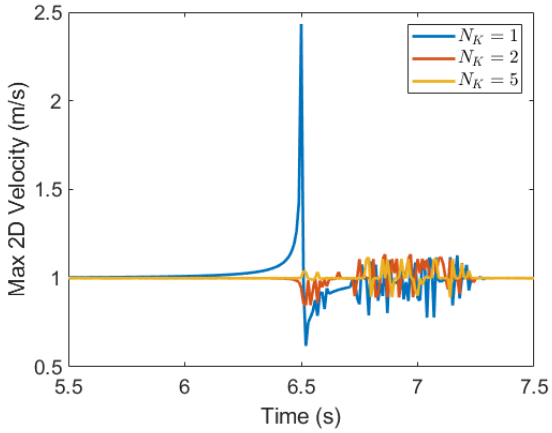

(**a**) $\Delta\phi$



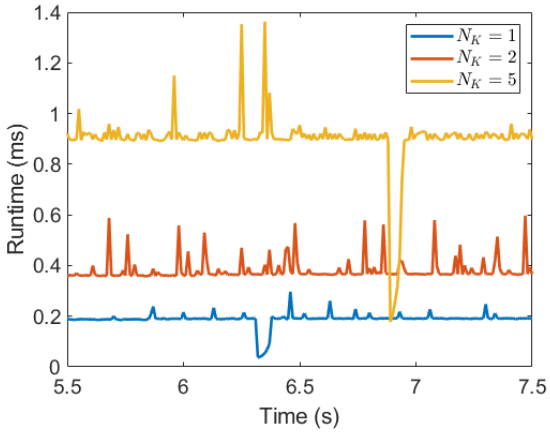

(**b**) Nearest neighbor distance

Figure 6.4: Six vehicle test $\Delta\phi$ and nearest neighbor distance

$N_K$ and the number of agents. Runtimes of roughly 0.2ms, 0.4ms, and 1ms were measured for the values of 1, 2, and 5 respectively. 10ms is the hard upper limit since the navigation algorithm is running at 100Hz. A value of $N_K = 2$ follows the velocity profile well while offering a reasonably low runtime.




(**a**) Velocity tracking

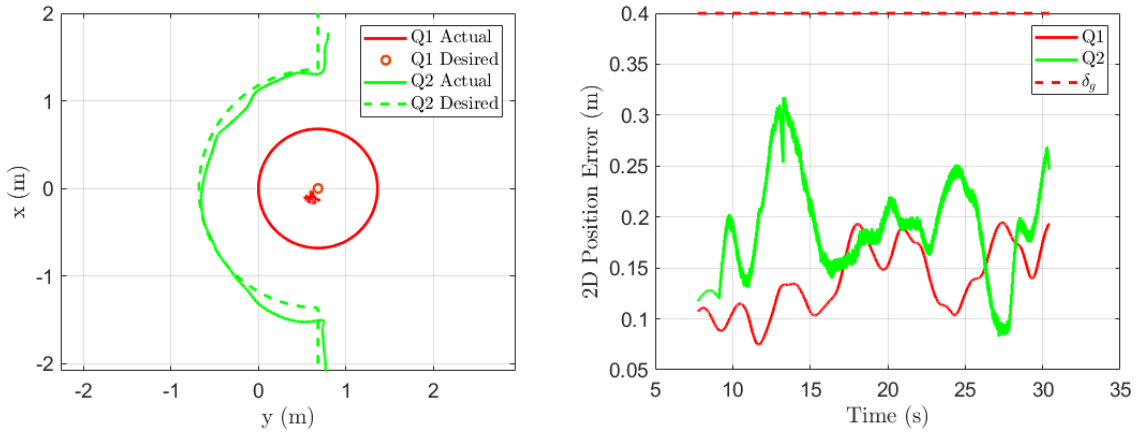



(**b**) Runtime

Figure 6.5: Effects of varying $N_K$

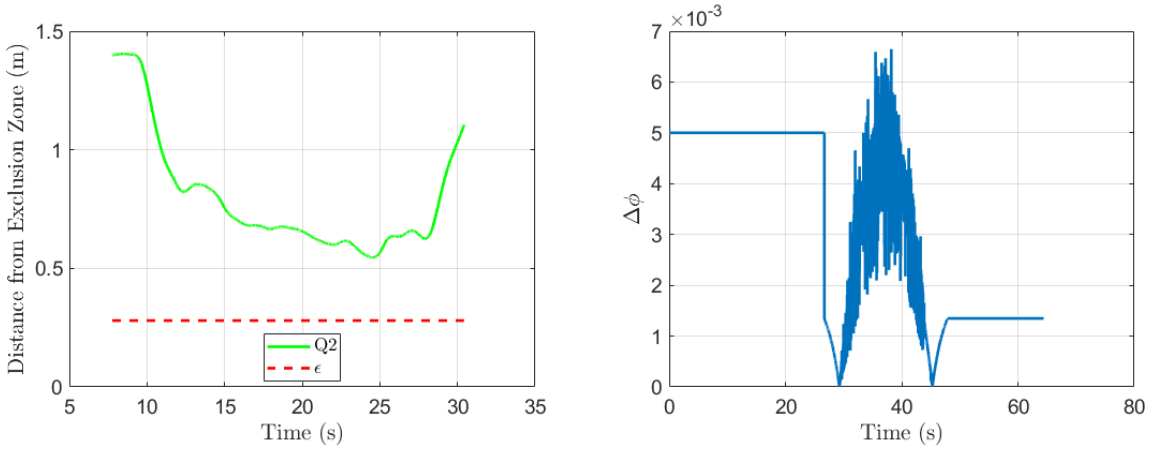
**Two Vehicle Flight Test** The same formation from the previous section in a reduced configuration (only Q1 and Q2) was flight tested to validate the navigation algorithm working safely to avoid collisions with non-zero controller error bounded by $\delta_g = 40cm$ for each vehicle. Figure 6.1 shows this test. Figure 6.6a shows the trajectories for failed agent Q1 within its exclusion zone (solid red circle) and Q2 which tracked its desired trajectory computed in real-time (dotted green) closely by its actual trajectory (solid green). The controller error of both agents remained within the $\delta_g$ bound

as seen in Figure 6.6b. Figure 6.7a shows that the actual trajectory of Q2 was a safe distance away from the exclusion zone. The dynamic value of $\Delta\phi$ can be seen in Figure 6.7b.




(**a**) 2D plot (**b**) Controller error

Figure 6.6: Two vehicle 2D plot and controller error




(**a**) Exclusion zone distance (**b**) $\Delta\phi$

Figure 6.7: Two vehicle exclusion zone distance and $\Delta\phi$

Figure 6.2b shows results of the navigation algorithm for a six vehicle test in simulation. Q1 fails and the remaining vehicles follow constant streamlines to avoid it. Validation was also successfully performed with a two vehicle experimental team.

### 6.1.6 Conclusion and Future Work

This section presented a novel, real-time, fluid-flow based navigation algorithm to allow quadrotor formation flights to safely continue in the presence of vehicle failures. Healthy agents followed

constant streamlines and a safety proof was presented that guarantees safety in the presence of single agent failure. Simulation and experimental results validate the method. Experiments were conducted with quadrotors flying in the $x$ direction of the $x$-$y$ plane i.e., constant $\psi$ direction of the $\phi$-$\psi$ plane. Using a rotation matrix, the CEM navigation algorithm can be improved further to accommodate quadrotor formation flight in any direction of the $x$-$y$ plane. We will extend simulation and experiments to larger teams with multiple failed quadrotors in future work.

## 6.2 High-Level Global Path Planning

### 6.2.1 Introduction

UTM offers the challenge of coordinating and routing many vehicles in a shared airspace safely. This problem is very complex and involves avoiding collisions between vehicles as well as generating feasible, efficient routes with reasonable planning times. In the general case, paths need to be planned for each vehicle, and collision avoidance is typically considered between pairs of vehicles. This offers flexibility in path choice but is computationally expensive and cannot easily guarantee multi-vehicle collision avoidance. Planning for vehicles to fly in a formation is more computationally efficient and offers multi-vehicle separation assurance as described with CEM above. Formation flight requires a group of vehicles to have similar starting and destination locations. Additionally, local group coordination may be required if there are communication limits that only allow for a subset of the vehicles to communicate with a centralized UTM system. Also, for applications such as terrain mapping, a formation may offer better sensor coverage. The formation can be treated as rigid body that can translate and rotate through space so that cooperative control techniques can be used to maintain the formation. However, the rigidity of this type of formation can limit its use case to open spaces with easy to navigate obstacles. Deformable formations have been proposed and explored in terms of control and formation keeping, but research in planning for coordinated teams has been limited.

This chapter proposes a general planning method for deformable formations. This includes formations that can translate, rotate, scale, and shear through space to offer the greatest flexibility in planning. We demonstrate our approach by considering a six vehicle 2D formation (the formation from Chapter 3 with an additional vehicle added). First, a minimum deformation safety parameter ($\lambda_{min}$) is calculated based on the initial formation according to continuum deformation. Second, the planning state space is defined as a decomposition of the homogeneous transformation relative to the initial formation. Third, a standard state-of-the-art planner is used to plan through our defined state space using custom inter-agent and formation-obstacle collision checkers. The inter-agent collision checker is derived as a function of the state space variables themselves and can be computed in constant time independent of team size. The formation-obstacle collision checker first computes a

convex hull approximation given the deformation variables and then feeds this into an algorithm to check for and avoid collisions with the environment. For the 2D formation considered, the convex hull is a triangle with rounded corners each approximated with six points. Three planning environments of increasing difficulty are considered using an open source planning library to explore the trade-offs between planning approaches assuming independent vehicles, a rigid formation, and a general deformable formation.

This section offers the following contributions:

1. A general approach to plan for homogeneous deformable formations with the ability to translate, rotate, scale, and shear through space.
2. Derivation of a constant-time inter-agent collision checking method for 2D deformable formations.
3. An efficient formation-obstacle collision checking technique for 2D deformable formations achieved by calculating an approximate formation convex hull and using it in standard collision checking.
4. A comparison of independent multi-vehicle, rigid formation, and deformable formation planning methods.

The remainder of this chapter section is organized as follows. Sec. 6.2.2 introduces recent related work in multi-vehicle planning. A concise problem statement is given in Sec. 6.2.3 and our approach to solve it is in Sec. 6.2.4. Results and conclusions are given in Sec. 6.2.5 and Sec. 6.2.6, respectively.

### 6.2.2 Related Work

The general multi-vehicle path planning problem considers $N_V$ vehicles and allows all vehicles to move independently through space. In $n - D$ each vehicle has a start location and goal location. Additionally, there are $N_O$ static, known obstacles in the environment. The objective is to plan goal-seeking paths for each vehicle that avoid collisions with obstacles and each other. The standard approach is to "stack" each state space and plan in the combined state space. Considering position only this makes a $3N_V$ dimensional space with $\mathcal{O}(N_V^2)$ inter-agent collision checking and $\mathcal{O}(N_V N_O)$ agent-obstacle checking. As the number of vehicles increases this becomes a computationally challenging problem.

To deal with the complexity of multi-vehicle path planning, most methods narrow the scope of potential solutions by keeping vehicles in a formation as they travel. Cooperative control techniques such as virtual structure or containment control define the behavior of the formation such that only the formation center or leader vehicles need to follow planned paths. Solutions are typically rigid formations or are selected from a small set of potential formations (e.g. line, triangle, diamond, etc).

Common path planners for these formations include evolutionary algorithms, optimization-based methods, potential field planners, and roadmap-based search [117]. Simpler potential field planners are computationally efficient but may have issues with local minima. More complex optimization-based methods can be complete but are computationally expensive. Additionally, these existing methods do not allow for general deformation of the formation that continuum deformation can provide. Ref. [14] developed a path planner for a continuum deformation formation but restricted motions to translations and used a computationally intensive particle swarm optimization (PSO) technique that doesn't guarantee convergence. In Refs. [118] and [119] an eigen decomposition is performed on the deformation. However, global planning is still accomplished with a constant size shape using A*. This shape represents a significant overestimate of the actual shape of the formation at any given time, removing many possible paths from consideration.

### 6.2.3 Problem Statement

In $n-D$ consider $N_V$ identical vehicles with starting positions $\mathbf{r}_{i,s} \in \mathbb{R}^{n\times 1}$ and goal positions $\mathbf{r}_{i,g} \in \mathbb{R}^{n\times 1}$. Additionally assume $N_O$ static, known obstacles exist in the environment. Both starting and goal positions are free of collision. Under continuum deformation $\mathbf{r}_{i,HT}(t) = \mathbf{Q}(t,t_0)\mathbf{r}_{i,s} + \mathbf{d}(t,t_0)$ is the desired position of every agent at time $t$ and is a function of initial position as well as deformation matrix $\mathbf{Q}(t,t_0) \in \mathbb{R}^{n\times n}$ and translation vector $\mathbf{d}(t,t_0) \in \mathbb{R}^{n\times 1}$. The goal of the deformable formation planner is to select $\mathbf{Q}(t,t_0)$ and $\mathbf{d}(t,t_0)$ for every time $t$ in a continuous manner such that the formation reaches its goal. In contrast to the general multi-vehicle planning problem, this formulation has a planning space with dimension $n^2+n$, inter-agent collision checking complexity $\mathcal{O}(1)$ and obstacle collision checking complexity $\mathcal{O}(N_O)$. Solutions are evaluated in terms of total travel distance of all vehicles as well as computation time. Only geometric constraints are considered and vehicles can translate and rotate in all directions. A 2-D workspace ($n=2$) is considered in this work but this dimension can be expanded to 3-D in future work.

### 6.2.4 Approach

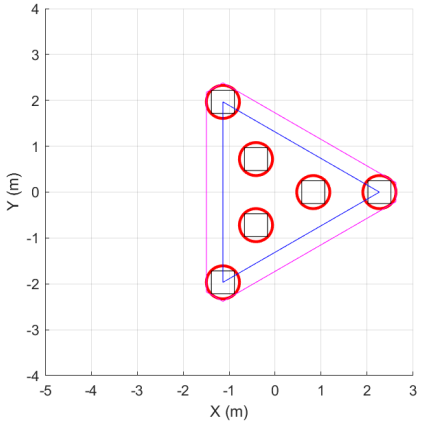
Our approach is using state-of-the-art path planning algorithms in $\mathbf{Q}-\mathbf{d}$ space with certain modifications. First, we define the state space via a decomposition of the $\mathbf{Q}$ matrix, inter-agent collision checking, and obstacle-formation collision checking.

**State Space Definition** Consider a decomposition of $\mathbf{Q}$ as follows:

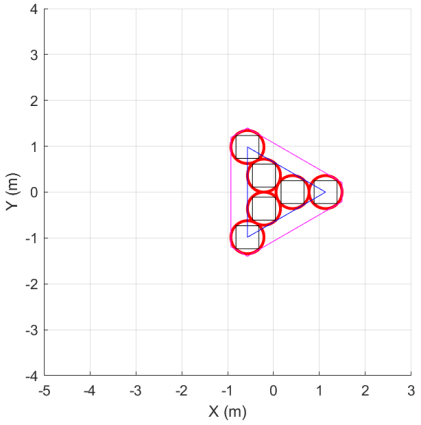
$$\mathbf{Q} = \mathbf{R}_{z,\theta_r}\mathbf{R}_{z,\theta_d}\Lambda\mathbf{R}_{z,\theta_d}^T \tag{6.6}$$

$$\mathbf{R}_{z,\theta_r} = \begin{bmatrix} cos(\theta_r) & -sin(\theta_r) \\ sin(\theta_r) & cos(\theta_r) \end{bmatrix}, \mathbf{R}_{z,\theta_d} = \begin{bmatrix} cos(\theta_d) & -sin(\theta_d) \\ sin(\theta_d) & cos(\theta_d) \end{bmatrix}, \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{6.7}$$

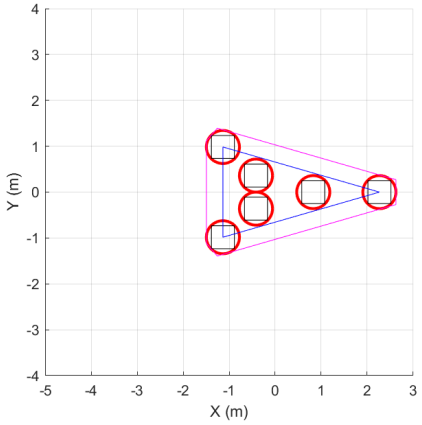
where $\theta_r$ is the formation rotation angle, $\theta_d$ is the deformation angle, and $\lambda_1$ and $\lambda_2$ are the eigenvalues of the deformation matrix [118]. This decomposition can produce any 2x2 $\mathbf{Q}$ matrix and still maintains six degrees of freedom for the full transformation, four in $\mathbf{Q}$ and two in $d\mathbf{d}$. More formally, this state space is $\mathbb{R}^4 \times SO(2)^2$. Figure 6.8 shows four example configurations using this decomposition. A six M330-Quadrotor formation ($N_V = 6$) is shown.



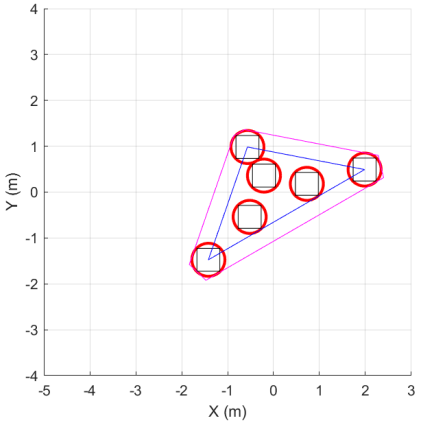

(**a**) Start state ($\lambda_1 = 1$, $\lambda_1 = 1$, $\theta_d = 0$)




(**b**) Contracted ($\lambda_1 = 0.5$, $\lambda_1 = 0.5$, $\theta_d = 0$)




(**c**) Contracted in Y ($\lambda_1 = 1$, $\lambda_1 = 0.5$, $\theta_d = 0$)




(**d**) Sheared ($\lambda_1 = 1$, $\lambda_1 = 0.5$, $\theta_d = \frac{\pi}{6}$)

Figure 6.8: Configurations of deformable formation ($\lambda_{min} = 0.5$, $\theta_r = 0$).

**Inter-Agent Collision Checking:** Continuum deformation offers a simple way to check for inter-

agent collisions. Inter-agent collision avoidance is guaranteed when the smallest eigenvalue of $\mathbf{U}_D = \left(\mathbf{Q}^T\mathbf{Q}\right)^{\frac{1}{2}}$ is greater than $\lambda_{min}$, a parameter derived from the initial formation. Since the decomposition directly contains the eigenvalues we have

$$\lambda_{\min} \leq \inf_{\forall t} \left\{ \lambda_1 \left(t\right), \lambda_2 \left(t\right) \right\}. \tag{6.8}$$

When the state space variables satisfy this inequality we can guarantee that inter-agent collisions are avoided. Therefore, inter-agent collision checking is a constant time operation that only depends on the state space variables. Moreover, this can be used to bound the search space and therefore no inter-agent collision checking is required.

**Formation-Obstacle Collision Checking** To handle formation-obstacle collision checking, a convex hull approximation must be created and checked for collisions with its environment (Figure 6.8). This effectively maps the state from $\mathbf{Q} - \mathbf{d}$ space into Euclidean space. This is a slight overestimate of the formation but results in fast collision checking. First, the leading triangle between all three leaders is considered. Each line segment is shifted by $D_s$ away from the formation center. Second, points on the leaders enclosing circles furthest from the center are calculated. Third, six intersection points are calculated to form a bounding triangles with "flattened corners." This method was inspired by Ref. [120]. This polygon can be used with standard collision checking algorithms and is generated at each collision check.

**Simulation Environment** The Open Motion Planning Library (OMPL) was used to implement the methods of this chapter [2]. Figure 6.9 shows a high level overview of the software system. More specifically, `OMPL.app` which is an extension of OMPL was used. This includes standard state-of-the-art planners, geometric environment setup, and the integration of the Flexible Collision Library. Two major changes were made to develop the approach in this chapter. First, a new State Space was made to reflect Eq. 6.6. Second, the `StateValidityChecker` was implemented for this state space. This included inter-agent collision checking and formation-obstacle collision checking as described previously.

### 6.2.5 Results

Planner performance was evaluated on three different test environments (Figures 6.10, 6.11, and 6.12). Several variations of the approach were considered per Table 6.1. RIGID is a rigid formation that can only rotate and translate and represents an important baseline since it is a common approach in the literature. HT-6DOF is the full approach as described in Sec. 6.2.4 and HT-4DOF and HT-5DOF each consider varying degrees of freedom which place them between the baseline and the full approach. Additionally, the general multi-vehicle planning approach "MULTISE2" was used as a baseline. This considers all vehicles independent with no formation constraints. RRT*, an

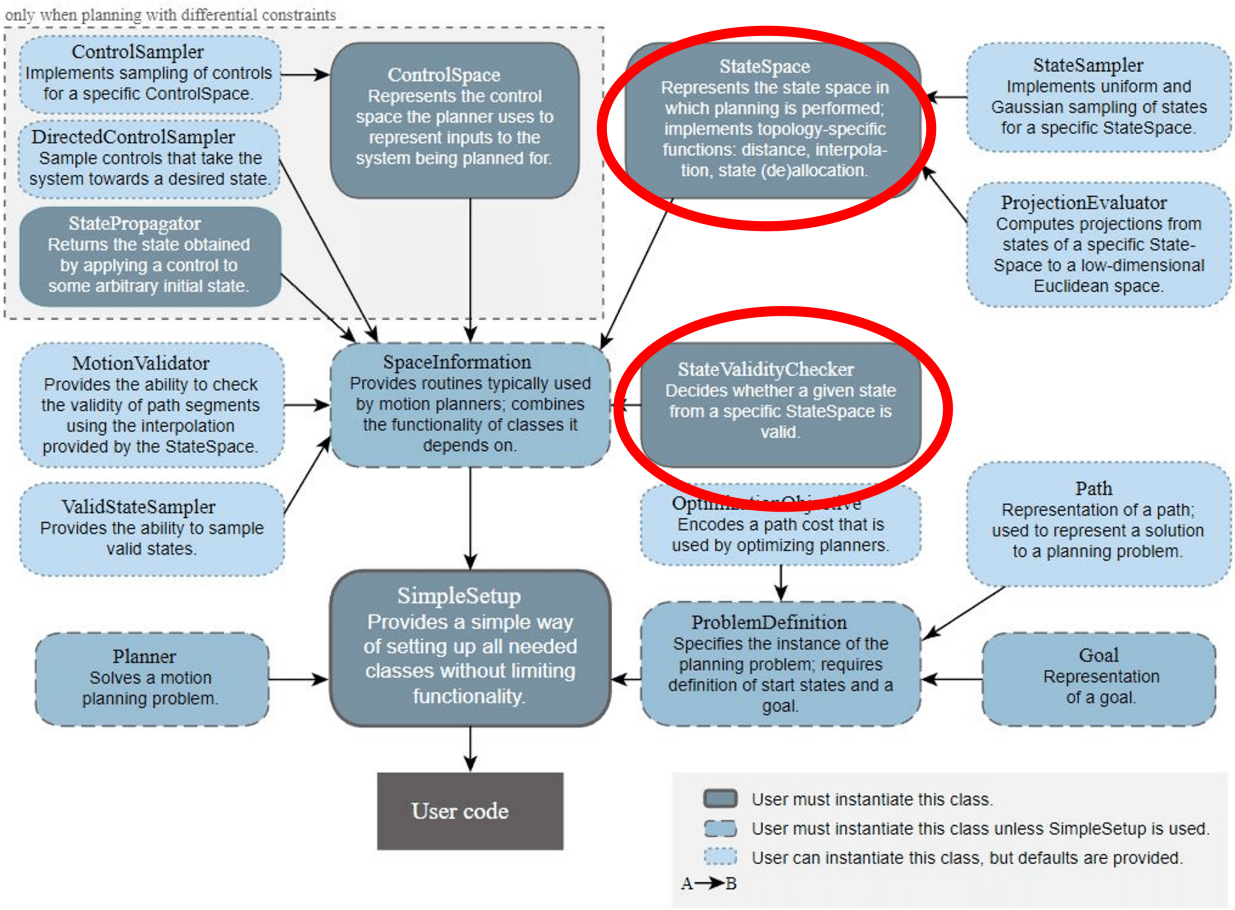



Figure 6.9: OMPL block diagram with modified portions indicated [2].

optimal sampling-based approach, was used as the planner for each method [121].

Table 6.1: Deformable planning methods state space variables bounds

| Method | $d_1$ | $d_2$ | $\theta_r$ | $\lambda_1$ | $\lambda_2$ | $\theta_d$ |
|---|---|---|---|---|---|---|
| RIGID | [-22,+22] | [-5,+5] | [-$\pi$,+$\pi$] | 1 | 1 | 0 |
| HT-4DOF | [-22,+22] | [-5,+5] | [-$\pi$,+$\pi$] | [0.5,1] | $\lambda_1$ | 0 |
| HT-5DOF | [-22,+22] | [-5,+5] | [-$\pi$,+$\pi$] | [0.5,1] | [0.5,1] | 0 |
| HT-6DOF | [-22,+22] | [-5,+5] | [-$\pi$,+$\pi$] | [0.5,1] | [0.5,1] | $SO(2)$ |

Results are shown in Tables 6.2, 6.3, and 6.4. Figures 6.13, 6.14, and 6.15 show the paths taken. ”DNF” indicates the approach did not find a solution for that test environment. With the 1s time limit in the easy environment, the baseline MULTISE2 approach did poorly with a 0% success rate. HT-4DOF was best with a 100% success rate and 129.1m total distance. However, the bottom three approaches were quite close in total distance. For the medium environment, MULTISE2 was competitive at 50% success rate and 132.3m total distance. RIGID did not find a solution since the

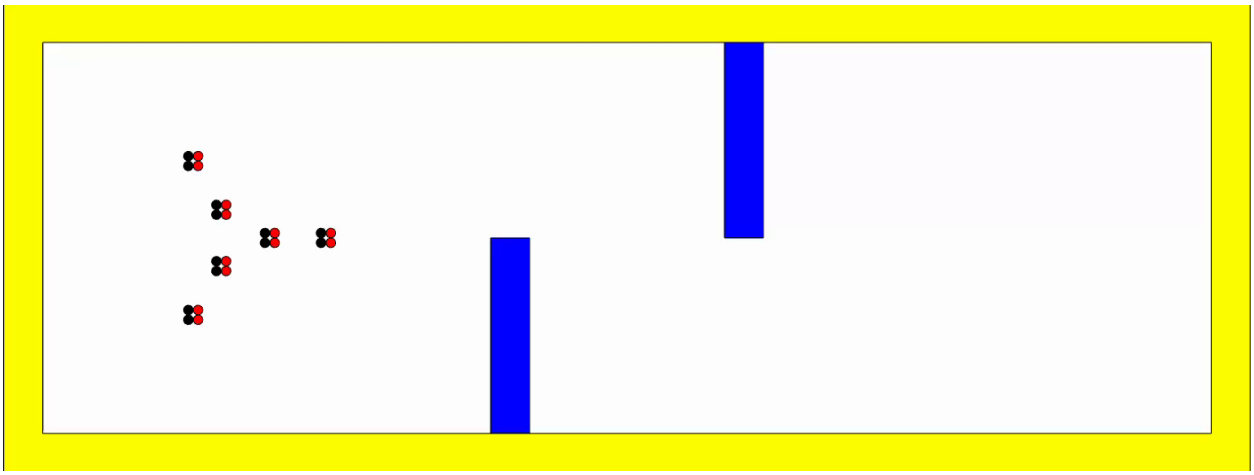


Figure 6.10: Easy environment. Rotation and translation is all that is needed to traverse.

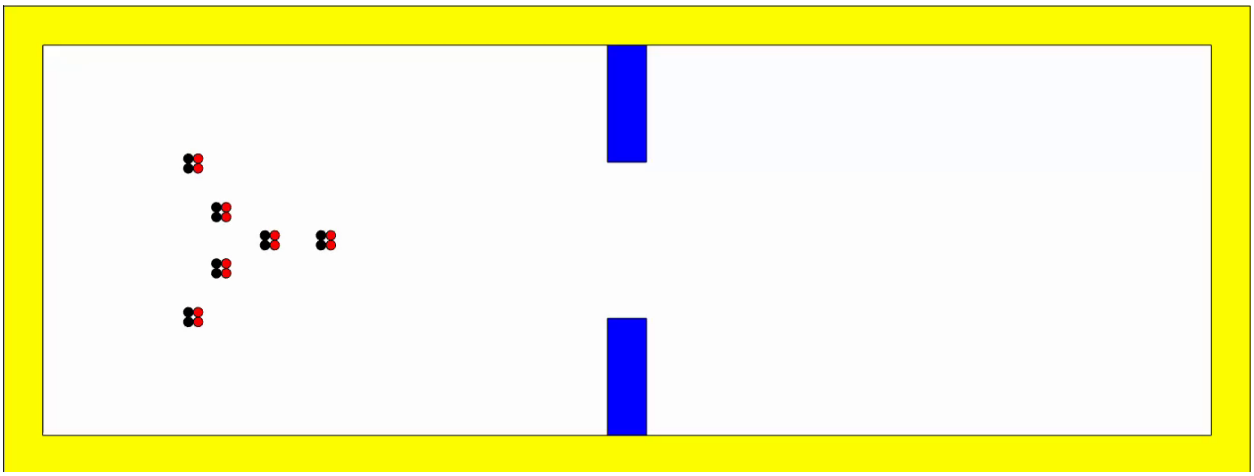


Figure 6.11: Medium environment. Scaling is needed to fit through the narrow passage.

environment required scaling. The last three performed better with HT-4DOF achieving the best total distance at 120.1m. Lastly, the impossible environment was impossible to pass through while maintaining a formation and so all of the formation approaches did not find solutions. The baseline MULTISE2 did find a solution with 20% success rate and total distance 172.1m. Similar results are seen for a 5s timeout (Table 6.3) and 60s timeout (Table 6.4). The extra time primarily helps MULTISE2 find a better solution as all of the formation methods have already converged near their optimal solutions within the first 1 second.

### 6.2.6 Conclusion and Future Work

This chapter presented a two-layer planning method for deformable formations. A 2D formation was considered that can deform through space according to a 6DOF homogeneous transformation. The deformation was decomposed into isolated components which allowed for the derivation of

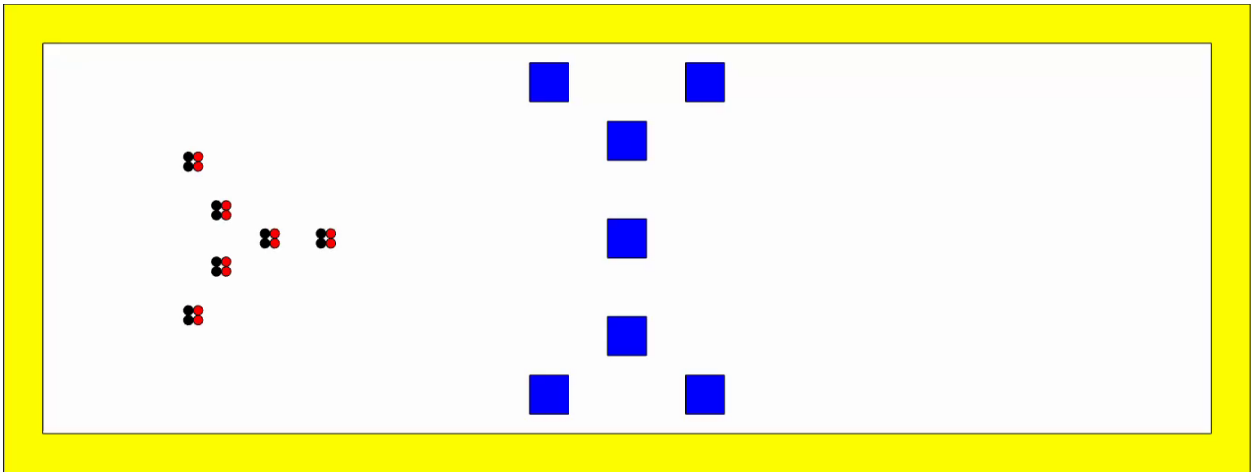


Figure 6.12: Impossible environment. Impossible to pass through while maintaining a formation.

Table 6.2: Success rate ($s_r$) and total distance ($d_t$) results averaged over 10 runs with a 1s time limit.

| | Test Environment | | | | | |
|---|---|---|---|---|---|---|
| | Easy | | Medium | | Impossible | |
| Method | $s_r$ (%) | $d_t$ (m) | $s_r$ (%) | $d_t$ (m) | $s_r$ (%) | $d_t$ (m) |
| MULTISE2 | 0 | DNF | 50 | 132.3 | **20** | **172.1** |
| RIGID | **100** | 136.2 | 0 | DNF | 0 | DNF |
| HT-4DOF | **100** | **129.1** | **100** | **120.1** | 0 | DNF |
| HT-5DOF | **100** | 130.0 | **100** | 120.2 | 0 | DNF |
| HT-6DOF | **100** | 132.9 | **100** | 120.2 | 0 | DNF |

a constant-time inter-agent collision checker. An efficient formation-obstacle collision checking technique was developed based on an approximate convex hull of the formation along with standard collision checking methods. Three environments were considered to explore the trade-offs between using the full 6DOF method, a rigid formation, or a baseline independent multi-vehicle method.

The test scenarios show the value of this general planning method for deformable formations. All three method variants (HT-4DOF, HT-5DOF, HT-6DOF) produced shorter paths than both the rigid formation (RIGID) and baseline (MULTISE2) approaches in all environments where a solution could be found. In the impossible environment, maintaining a formation was impossible and so only the MULTISE2 approach was able to find a solution. However, since the planning time was quick for the other environments (100% success in 1s), a short timeout could be utilized to quickly search for a solution and give up if one is not found and then transition to using the baseline planner. Additionally, combining the approaches directly in the planning stage could be considered as well. After planning stalls, or perhaps increasing with probability as time goes on, nodes could be connected together using the MULTISE2 approach to get through cluttered environments. These hybrid approaches should be considered in future work. A 3D extension of the method should be

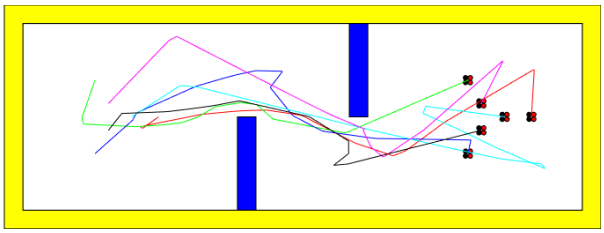
Table 6.3: Success rate ($s_r$) and total distance ($d_t$) results averaged over 10 runs with a 5s time limit.

| | Test Environment | | | | | |
|---|---|---|---|---|---|---|
| | Easy | | Medium | | Impossible | |
| Method | $s_r$ (%) | $d_t$ (m) | $s_r$ (%) | $d_t$ (m) | $s_r$ (%) | $d_t$ (m) |
| MULTISE2 | 30 | 191.6 | 60 | 124.3 | **40** | **155.7** |
| RIGID | **100** | 134.5 | 0 | DNF | 0 | DNF |
| HT-4DOF | **100** | **128.2** | **100** | **120.1** | 0 | DNF |
| HT-5DOF | **100** | 128.9 | **100** | **120.1** | 0 | DNF |
| HT-6DOF | **100** | 130.4 | **100** | **120.1** | 0 | DNF |

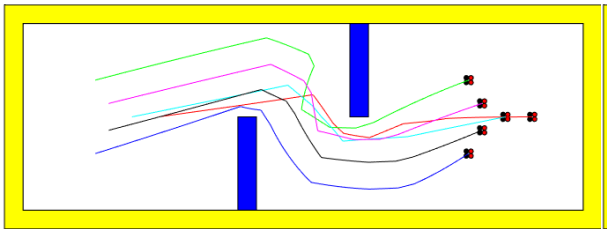
Table 6.4: Success rate ($s_r$) and total distance ($d_t$) results averaged over 10 runs with a 60s time limit.

| | Test Environment | | | | | |
|---|---|---|---|---|---|---|
| | Easy | | Medium | | Impossible | |
| Method | $s_r$ (%) | $d_t$ (m) | $s_r$ (%) | $d_t$ (m) | $s_r$ (%) | $d_t$ (m) |
| MULTISE2 | 40 | 161.9 | 90 | 124.3 | 60 | 148.2 |
| RIGID | **100** | 134.3 | 0 | DNF | 0 | DNF |
| HT-4DOF | **100** | **127.5** | **100** | **120.1** | 0 | DNF |
| HT-5DOF | **100** | 127.9 | **100** | **120.1** | 0 | DNF |
| HT-6DOF | **100** | 129.2 | **100** | **120.1** | 0 | DNF |

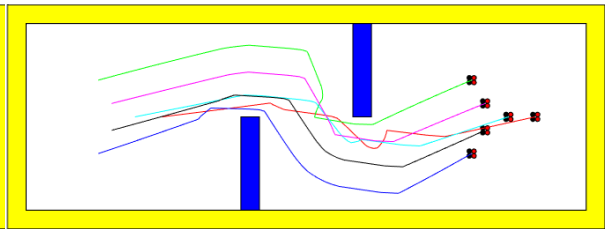
considered with a 12DOF state space decomposition, inter-agent collision checking derivation, and bounding convex polyhedron.



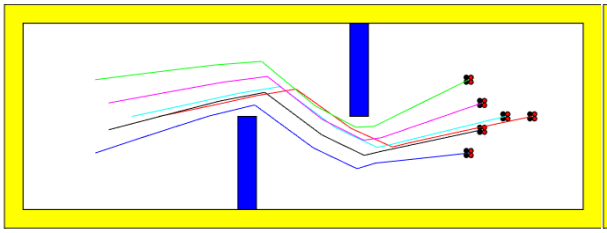
(**a**) MULTISE2, 60s



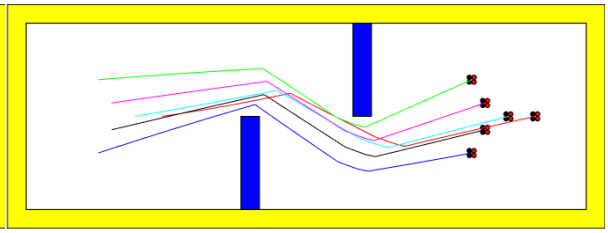
(**b**) RIGID, 1s



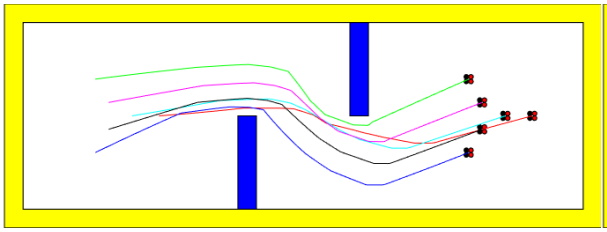
(**c**) RIGID, 60s



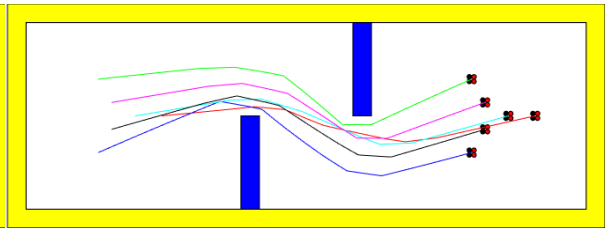
(**d**) HT-4DOF, 1s



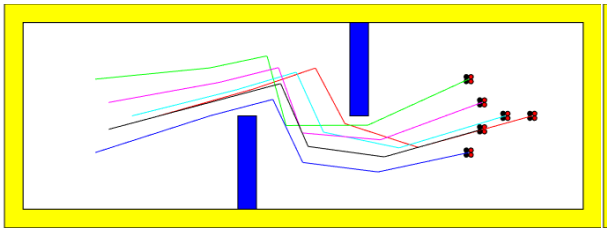
(**e**) HT-4DOF, 60s



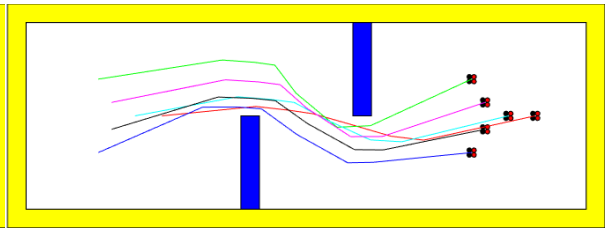
(**f**) HT-5DOF, 1s



(**g**) HT-5DOF, 60s

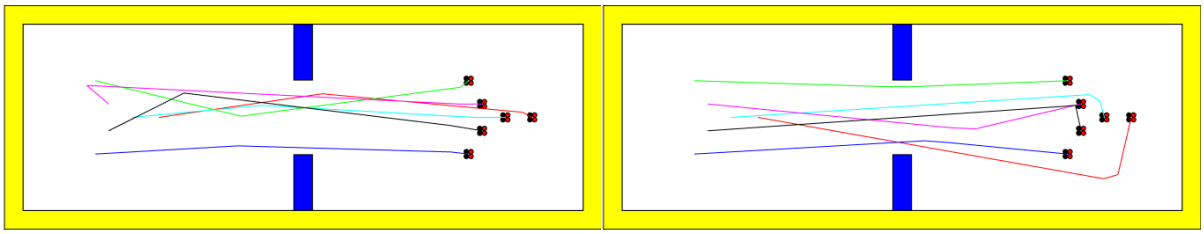


(**h**) HT-6DOF, 1s

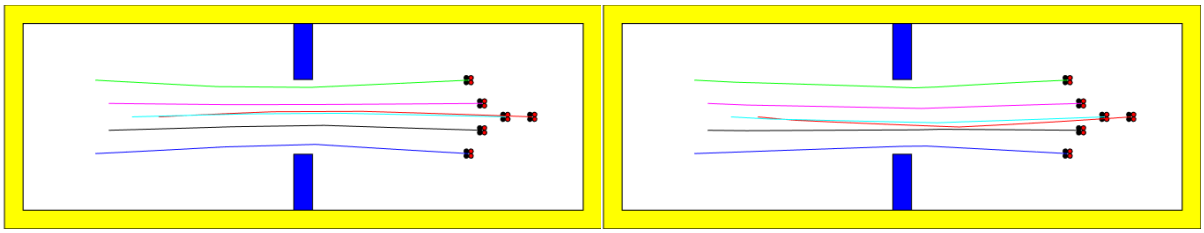


(**i**) HT-6DOF, 60s

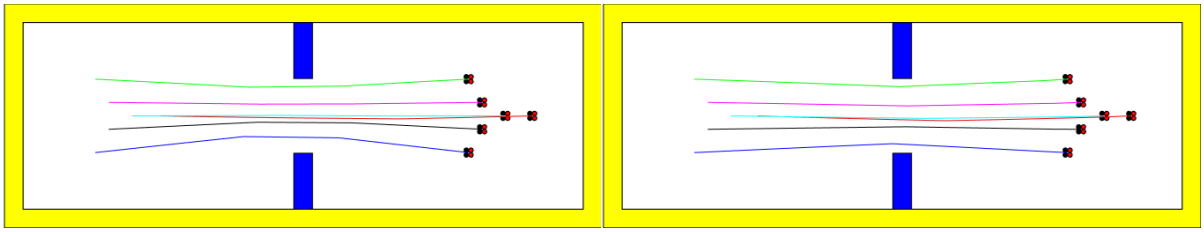
Figure 6.13: Easy environment results



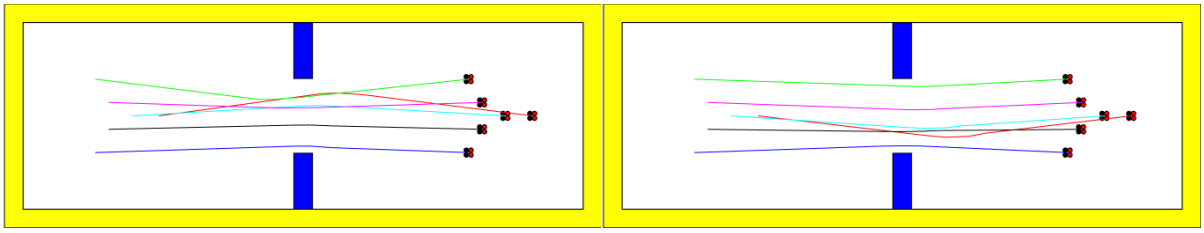
(**a**) MULTISE2, 1s (**b**) MULTISE2, 60s



(**c**) HT-4DOF, 1s (**d**) HT-4DOF, 60s

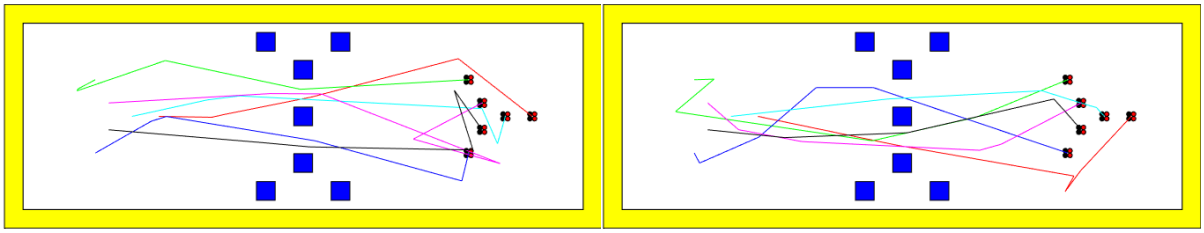


(**e**) HT-5DOF, 1s (**f**) HT-5DOF, 60s



(**g**) HT-6DOF, 1s (**h**) HT-6DOF, 60s

Figure 6.14: Medium environment results



(**a**) MULTISE2, 1s (**b**) MULTISE2, 60s

Figure 6.15: Impossible environment results

# CHAPTER 7

# Conclusion and Future Work

This dissertation has investigated autonomy to plan for and coordinate multiple UAS in a variety of mission-appropriate configurations. Formation control and planning for UAS traffic management, multilift payload transportation using haptic guidance, and wildfire mapping were considered. Extensive experimental validation and realization as well as software studies for these missions were performed. This results in well-defined proof of concepts for diverse missions that future researchers and society at large can use as a spring-board to bring them into reality. Moreover, this dissertation only scratches the surface of what can be pursued with the multi-UAS technology that is available today. Our imagination and creativity are the only limits to what we can pursue. Below are specific contributions with conclusions that lead to a discussion of future work.

## 7.1 Contributions

Chapter 2 provided the foundation for experimental realization of the multi-UAS missions considered. The M330-Quadrotor, `rc_pilot_a2sys` flight controller, and instrumented payload developed for this thesis are opensource and publicly available. This allows for more people to gain access to this technology through its low costs and support via detailed documentation. Greater access to these technologies will enable further research that will move the field forward.

In Chapter 3, a safe and deformable formation control solution based on continuum deformation was refined and experimentally validated. A derivation was made to relate experimentally obtainable vehicle controller errors to global deviation bounds needed to guarantee safety for the formation. This resulted in a definition of required inter-vehicle separation constraints such that safe trajectories could be planned for the system. This safety guarantee for any multi-UAS formation under continuum deformation extends the existing theory into practical scenarios, rather than being confined to idealized simulations.

In Chapter 4, we defined, implemented, and experimentally evaluated a haptically-guided cooperative payload transportation capability. Five quadrotors tethered to an instrumented payload were guided by a human user directly pushing on the payload. This required development of

real-time user force estimation based on inline tension force sensors as well as an admittance controller to update virtual dynamics according to user input force. This novel user interaction strategy can be used for package delivery, construction, and disaster relief applications. The results uncovered an interesting interplay between user applied force and controller effort. Both work in tandem with admittance control active, lowering the required controller effort to follow a desired trajectory. However, when deactivated at a geofence boundary, they work in opposite directions acting as a cue to the user. This results in a robust guidance system within a cluttered environment.

In Chapter 5, a multiple UAS team is directed through multi-agent coverage planning and boundary following to find and map wildfire boundaries despite complex mountainous terrain. Computationally efficient planning methods were developed to adhere to the resource constrained environment. Multi-vehicle task allocation was handled by a state machine between exploration (coverage) and exploitation (line following). Efficiency gains were achieved by separating the 3D problem into 2D lateral and 1D terrain avoidance sub-problems. Runtime constraints were a major influence of the methods developed in this chapter. For this purpose, they were effective. However, if given additional computational resources more general 3D planning methods would outperform our methods. Therefore, when applying the methods from this chapter to new problems, computational resources should be carefully considered.

In Chapter 6, path planning for a deformable formation was considered with both a high level global planner to generate a trajectory that is followed by a reactive low level vehicle guidance module. The guidance module utilized fluid flow fields to safely direct the team around obstacles. Computational efficiency was achieved with a real-time navigation function that only computed the immediate commands at each time step. Safety guarantees were provided that prescribed minimum vehicle separation given obstacle size. The global planner proposed a general method for deformable formation planning. Instead of planning in Euclidean space for individual vehicles, the method planned for the deformation (translation, rotation, scale, and shear) of the formation itself via the definition of a new state space. A constant-time inter-agent collision checker was derived based on the state space variables themselves. Formation-obstacle collision checking was performed by taking the approximation of the formation convex hull and using it in standard collision checking algorithms. Together, these two modules allow for deformable formations to be applied in a wider variety of domains. The guidance module will be effective in avoiding collision for lost link failures and pop-up obstacles, but will be ineffective with more general failures. The global planner will easily find paths in environments where formations are viable but will typically fail in cluttered environments.

## 7.2 Future Work

While the Chapter 2 autopilot and software systems were sufficient for this dissertation, there are many features that could further improve the `rc_pilot_a2sys` software package. The flight controller might include new flight modes (e.g. follow me, primitive motions) and additional sensor support. The M330-Quadrotor platform is sufficient for GPS or motion capture navigation but in environments where GPS and motion capture are inaccessible a reliable onboard sensing system is needed. Some work has been performed to integrate a RealSense system at the cost of weight and reduced flight time resulting in migration to a hexacopter platform able to carry both a companion computer and the necessary sensors. The core principles of simplicity, low-cost, and open-source should be upheld so that this project is accessible to everyone.

Chapter 3 focused on the first experimental evaluation of continuum deformation. A realistic communication topology where followers indirectly receive their setpoints allowed us to uncover some improvements needed in the theory. However, motion capture was used to allow for "perfect sensing", vehicle failures couldn't be handled, specific vehicles were set as leaders without flexibility, and only a 2D version of the formation control method was used. In the future to further develop this work we should consider using onboard sensing on all vehicles, handle vehicle failures, allow for dynamic reallocation of leadership to vehicles, and consider a full 3D continuum deformation.

Chapter 4 explored haptic guidance of a multi-lift slung payload transportation system on an real system for the first time. The results are positive and indicate further development is warranted. For a deployed application there will be several differences that need to be taken into account. First, for the work in this chapter an instrumented payload was used to provide easy access to sensor data in a centralized location as well as ground truth user applied force data. However, in a deployed application payloads will not typically be instrumented and will be of varying size, shape, and mass. Therefore, the inline tension sensors would need to be moved to the vehicle sides with data shared to achieve the same results. Additionally, improved sensors could be used to provide more reliable force data that could be further improved with online estimation of scale and bias terms of each sensor. This would also allow for full 3D haptic guidance to be added because the $z$ force estimate should be improved. Also, consideration of hybrid vehicles (e.g. aircraft with aerodynamic lifting surfaces) should be completed to enable both efficient forward flight as well as vertical take-off and landing capabilities. A human subjects study with these improvements should be performed to compare the effectiveness of the haptic interface vs. a joystick or a tablet interface. A simulated disaster zone case study with expert human subject participants would support better assessment of the feasibility for this system concept. Required flight times, payload masses, and general logistics should be considered. A path towards safety certification of this type of system should be explored. Failure probabilities of individual components can be accounted for to calculate a system wide

safety rate. This risk should then be compared against the risk from specific applications. Dangerous applications such as military or time-sensitive disaster relief scenarios will likely tolerate a poorer system wide safety rate since inaction could result in worse outcomes. However, safer applications such as consumer package delivery will likely require a higher system wide safety rate to be worth the additional risk to the user.

In Chapter 5, computationally efficient methods were developed in a simulation environment to show the feasibility of using teams of UAS to find and map wildfires. Much work remains before this system could be integrated into a real disaster response team where firefighters and other key stakeholders are able to use it. FAA regulations currently make it difficult to implement this concept system (see Part 107 for commercial operators). Every vehicle needs to have a remote pilot and the vehicle needs to remain within line of sight. Additionally, vehicles must stay below 400 feet Above Ground Level (AGL) and out of restricted airspace (including 5 miles around airports, and in particular wildfires currently). FAA regulations are in constant development and will likely change to be more supportive in the coming years. With respect to technology development, fixed wing UAS have very good range and sensing capabilities but sensor range and autonomous in-flight decision-making need to improve to assure real-time reactions and accurate real-time wildfire mapping. Detailed terrain maps exist that could be pre-loaded to vehicles to avoid terrain and GPS-based navigation allows for vehicles to track trajectories commanded from a central ground control station. Several capabilities were assumed with the vehicles including perfect communication and trajectory following controllers. In real life, emergency response systems would need to be carefully integrated with existing equipment and people so as to not cause more harm than good. Preliminary flight testing with "simulated wildfires" in controlled areas would need to be handled to assure mission efficacy. In wildfire mapping, alternative methods to acquire real-time maps of evolving wildfires include satellite imagery (visual spectrum and infrared) as well as single aircraft at higher altitudes. However, for this research the premise was that smoke would block the ability to sense if a camera was too far away, therefore, the low flying UAS offer valuable fire data. Intelligent integration into the existing emergency management response will be one of the most important factors when actually implementing a swarm of UAS for wildfire management. Existing emergency response teams will slowly integrate the new technology one step at a time. Airspace geofence boundaries can ensure that manned vehicles always know where the UAS are flying to avoid mid-air collision. Additionally, mission assignment (telling the swarm to map a specific area), and map reporting (swarm telling emergency response team where the fires are) must be smooth and effective for the extra burden of using a UAS swarm to add value.

The methods from Chapter 6 offer preliminary concepts for resilient and capable deformable formations. However, many of the assumptions made currently limit their use. For the low level guidance module, extension to 3D is required. Additionally, deriving tight bounds on safety

constraints for multiple heterogeneous failures with a heterogeneous formation should also be considered. Introducing a higher level module that decides when it is safe to transition into the fluid flow navigation function would make the method much more effective. The global planner should also be extended to 3D. Additionally, considering a hybrid approach that is able to look at the environment and select a planner that makes sense, or use a combination should be researched as that can have far reaching impact in a range of applications. For UTM in particular, both of these methods can be part of a larger planning system that efficiently routes hundreds of vehicles around complex terrain and obstacles in real-time.

# APPENDIX A

# Supplement to Hardware Description

## A.1 Printed Circuit Boards

I developed three PCBs during the course of my PhD to support experimental realizations of my work. Each is a two layer board that allows convenient and reliable interfacing between different electronic components. DipTrace was used to design them and all associated design files, gerber files, and bill of materials are available[1].

### A.1.1 Relay Cape

Serial messages sent over XBees were used to achieve reliable, low-latency motion capture integration. However, with multiple vehicles this meant that one serial port was needed for every one vehicle receiving data. Six UART (serial) senders were needed (five vehicles, one payload). A BeagleBone Black was used to receive the motion capture data packets over ethernet and construct and send custom packets to the vehicles over XBee. The five hardware serial ports (and one USB with a USB to XBee dongle) were used to achieve this. Therefore, these five ports needed to be interfaced to five XBees on a ”cape” that fits into the GPIO pins of the BeagleBone Black. Additionally, a kill switch needed to be integrated such that the XBees themselves could read the state of a pin and transmit that to XBees on the vehicles, safely turning off when a kill switch is toggled or after not receiving data for a configurable timeout value. Figure A.1 shows the 3D view of the PCB and Figure A.2 shows the schematic.

### A.1.2 XBee Kill Switch

On the receiving side of the kill switch messages over XBees is the XBee Kill Switch PCB. This board is on the quadrotor and serves two purposes. First, it connects to the BeagleBone Blue running the flight controller, receiving power and connecting the onboard XBee UART signal to it. Second, it optionally passes through the PWM signals from the BeagleBone Blue to the ESCs to

[1] https://tinyurl.com/2ncv8rae

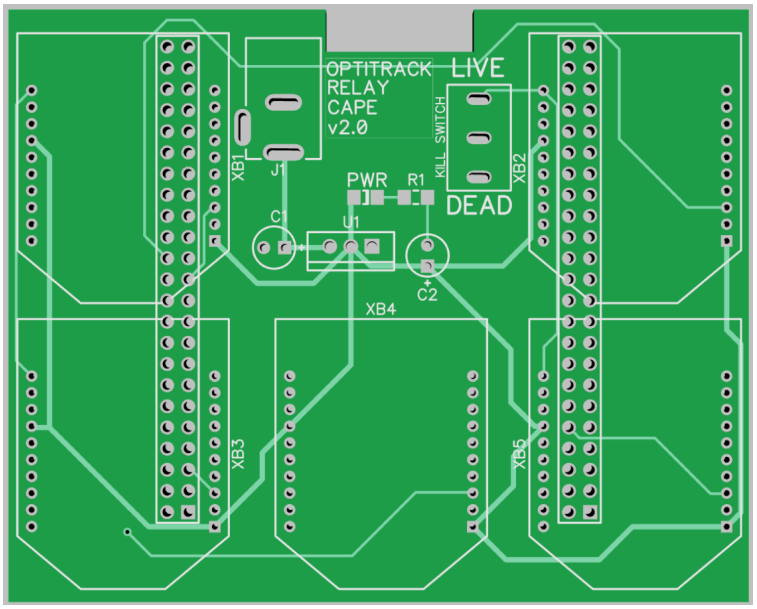



Figure A.1: Relay cape 3D view

act as a kill switch. Logical AND gates are used alongside a signal from the XBee to control this. When the GCS operator flips a switch on the Relay Cape PCB, this signal get mirrored on the XBee Kill Switch board, passing GND to the ESCs and turning off the motors. This also occcurs when a timeout on communication loss is reach, acting as a deadman's switch. Figure A.3 shows the 3D view and Figure A.4 shows the schematic of the PCB.

### A.1.3 Instrumented Payload Interface Board

The instrumented payload developed to support the research of Chapter 4 involved the use of several different electronic subsystems. A BeagleBone Blue acted as the main computer but needed to interface with five tension sensors and associated ADCs, a haptic force validation sensor, and be powered by an onboard LiPo battery. Moreover, all of this needed to be neatly contained within a lightweight Styrofoam sphere. Therefore, an interface board was developed to assist in these connections. Specifically, this board mounts to a 3D printed piece that houses the BeagleBone Blue and LiPo battery and it interfaces between the BeagleBone Blue and five inline tension sensors. Figure A.5 shows the 3D view and Figure A.6 shows the schematic of the PCB.

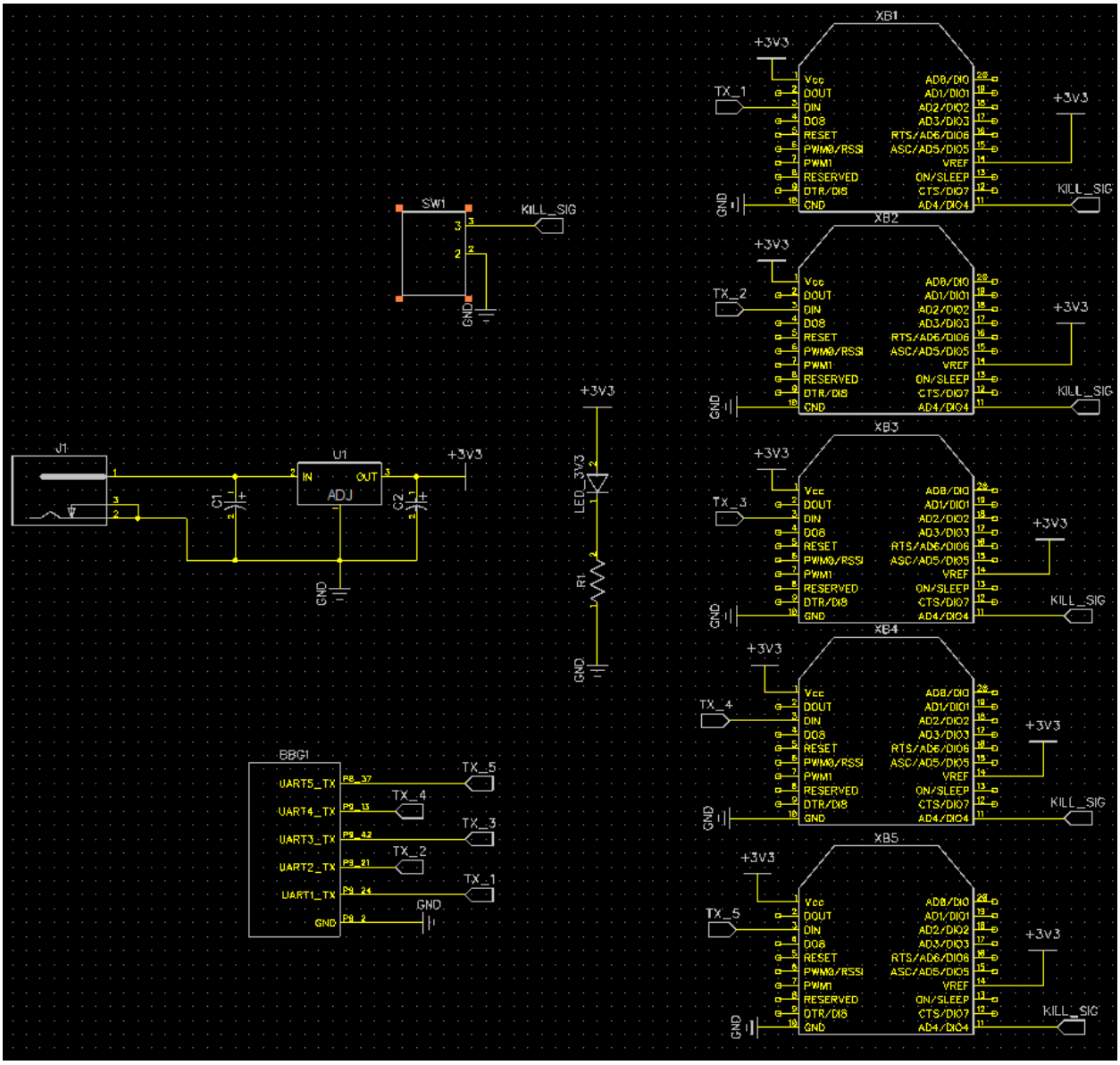



Figure A.2: Relay cape schematic

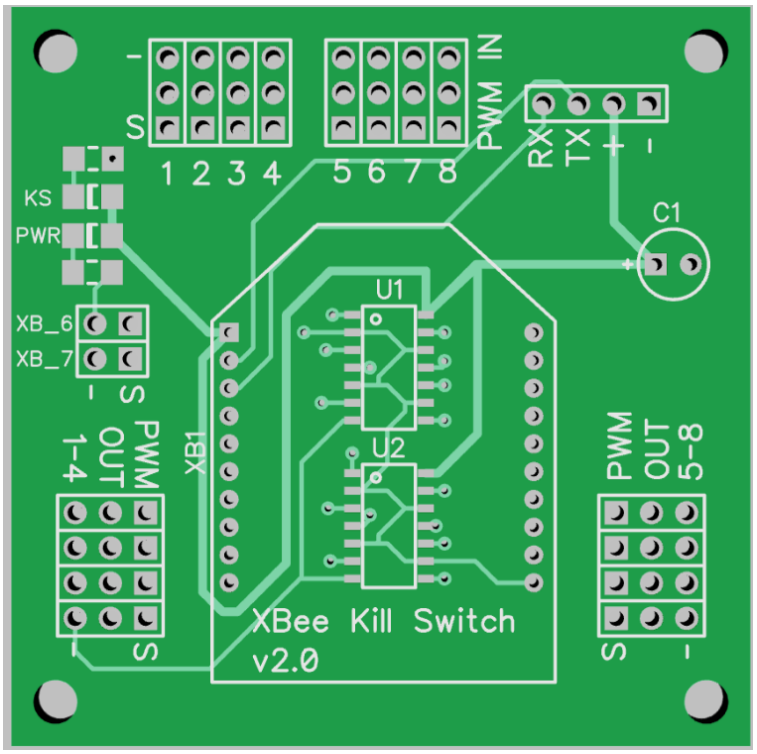



Figure A.3: XBee kill switch 3D view

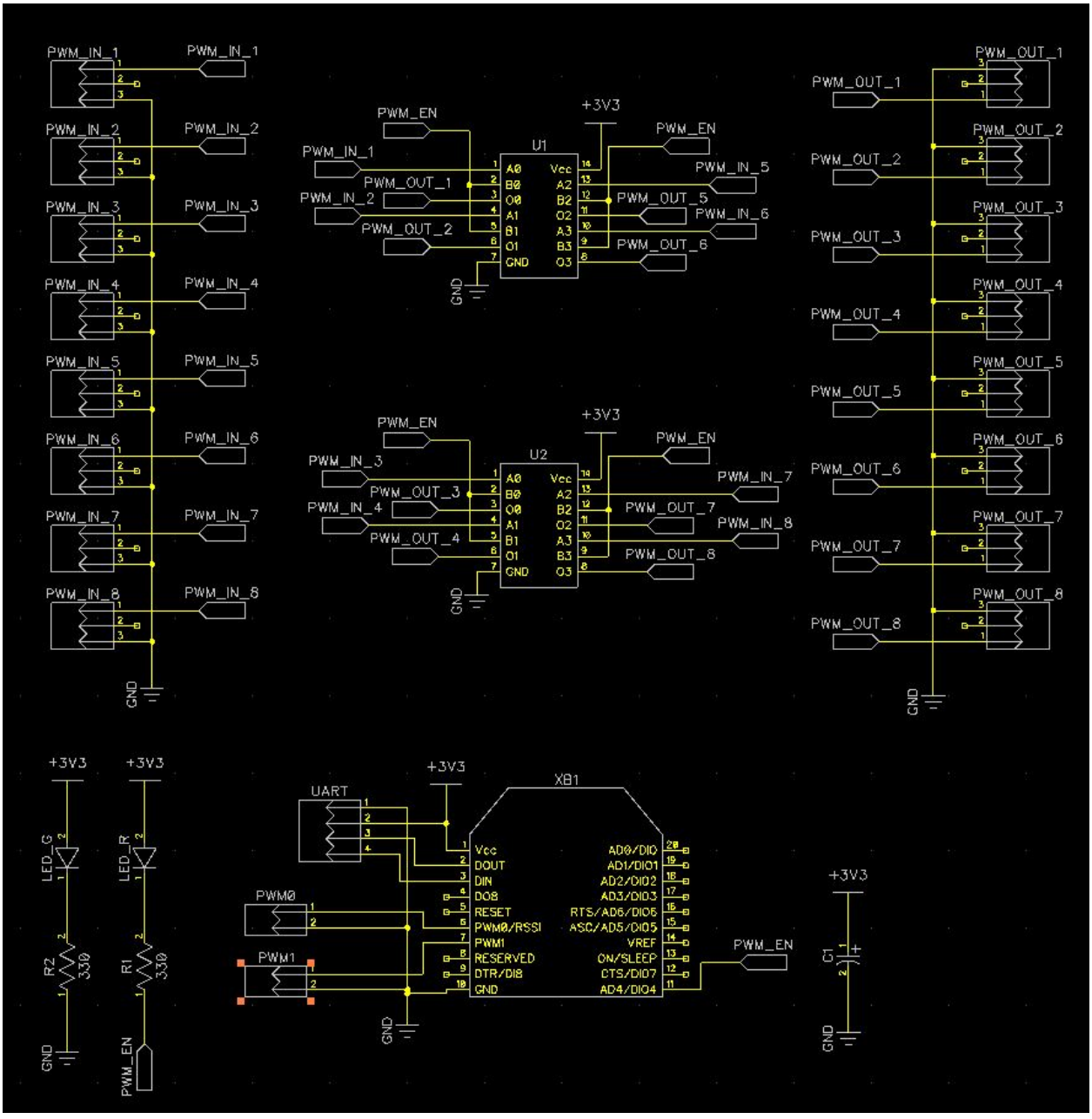



Figure A.4: XBee kill switch schematic

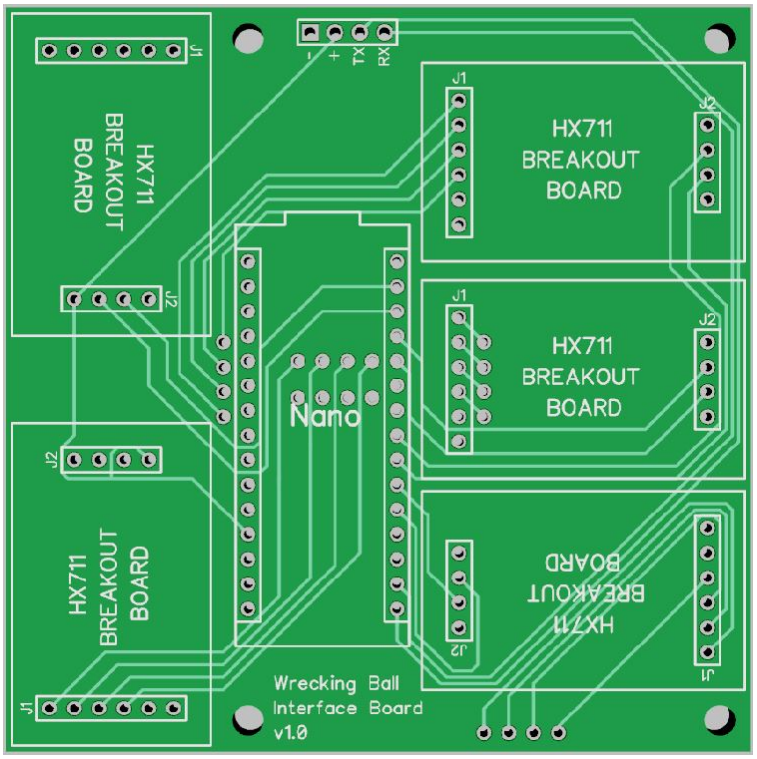



Figure A.5: Instrumented payload interface board 3D view

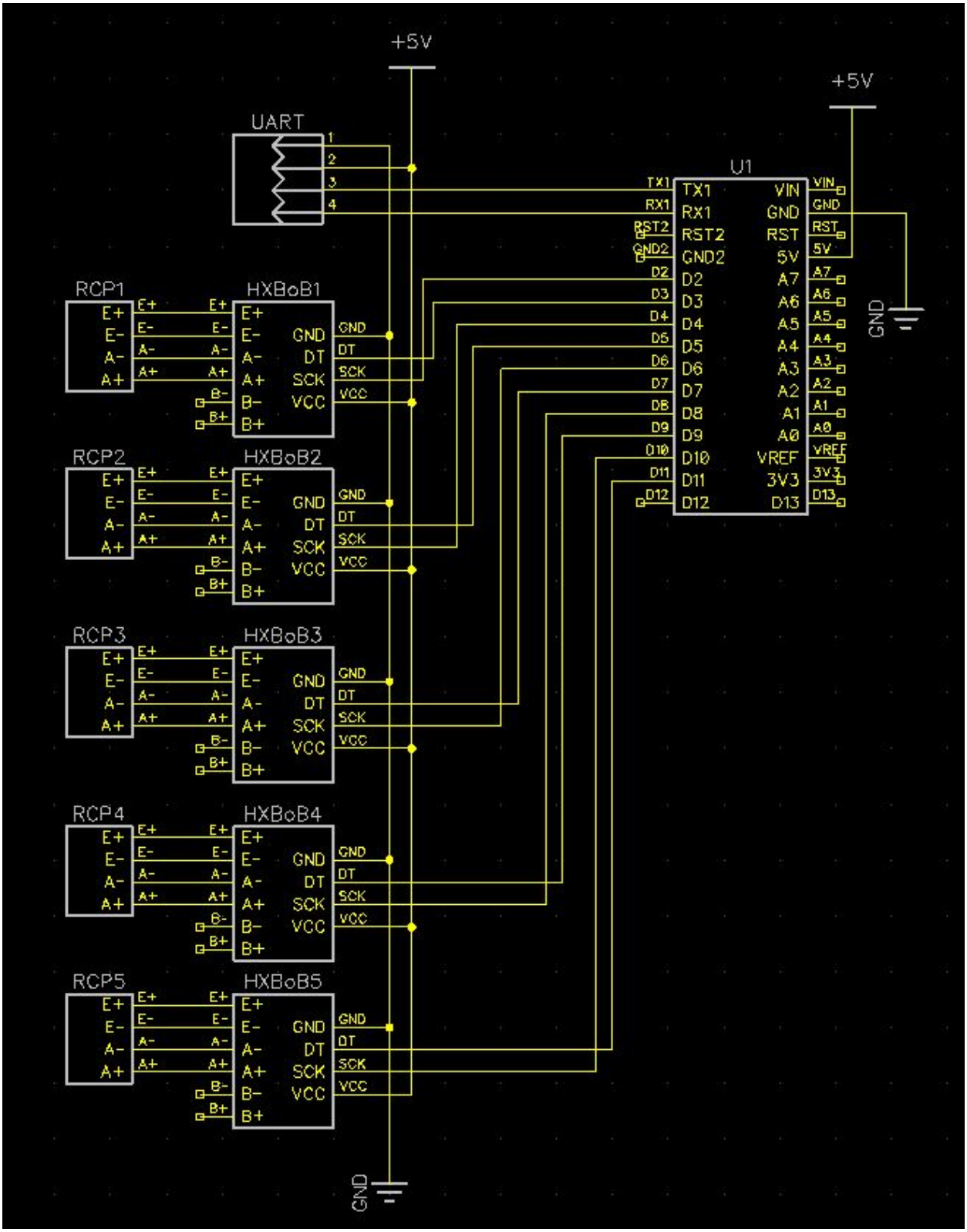



Figure A.6: Instrumented payload interface board schematic

# APPENDIX B

## Local to Global Deviation Supplemental Derivation

Verification of Zero $\mathbf{D}_g$ Equation Term
Here we show that

$$\left[(\mathbf{I}-\mathbf{W})^{-1}-\boldsymbol{\alpha}\right]\mathbf{P}_{L,HT}=0$$

$$(\mathbf{I}-\mathbf{W})^{-1}=\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1}=\begin{bmatrix}\mathbf{E} & \mathbf{F}\\ \mathbf{G} & \mathbf{H}\end{bmatrix}$$

Block matrix inversion indicates:

$$\begin{aligned}\mathbf{E}&=\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1}\\ \mathbf{F}&=-\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\\ \mathbf{G}&=-\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1}\\ \mathbf{H}&=\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\end{aligned}$$

We can then assign A,B,C,D block matrices as:

$$\mathbf{A}=\mathbf{I}_3,\ \mathbf{B}=\mathbf{0}_{3\times(n-3)}$$
$$\mathbf{C}=\begin{bmatrix}-w_{4,1} & -w_{4,2} & -w_{4,3}\\ \vdots & \vdots & \vdots\\ -w_{n,1} & -w_{n,2} & -w_{n,3}\end{bmatrix},\ \mathbf{D}=\begin{bmatrix}1 & \dots & -w_{4,n}\\ \vdots & \ddots & \vdots\\ -w_{n,4} & \dots & 1\end{bmatrix}$$

Since $\mathbf{P}_{L,HT}$ has only three non-zero entries we can ignore $\mathbf{F}$ and $\mathbf{H}$, and obtain the other two matrices as:

$$\begin{aligned}\mathbf{E} &= \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1} \\ &= \mathbf{A}^{-1} = \mathbf{I}_3\end{aligned}$$

$$\begin{aligned}\mathbf{G} &= -\left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1} \\ &= -\mathbf{D}^{-1}\mathbf{C} \\ &= -\begin{bmatrix} -1 & \dots & w_{4,n} \\ \vdots & \ddots & \vdots \\ w_{n,4} & \dots & -1 \end{bmatrix}^{-1} \begin{bmatrix} w_{4,1} & w_{4,2} & w_{4,3} \\ \vdots & \vdots & \vdots \\ w_{n,1} & w_{n,2} & w_{n,3} \end{bmatrix}\end{aligned}$$

Rastgoftar [27] has shown that the above expression is equal to a matrix of $\alpha$ parameters:

$$\mathbf{G} = \begin{bmatrix} \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} \\ \vdots & \vdots & \vdots \\ \alpha_{n,1} & \alpha_{n,2} & \alpha_{n,3} \end{bmatrix}$$

Assembling the final system gives:

$$(I_n - W)^{-1} = \begin{bmatrix} & \mathbf{I}_3 & & \times & \dots & \times \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \times & \dots & \times \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n,1} & \alpha_{n,2} & \alpha_{n,3} & \times & \dots & \times \end{bmatrix}$$

$$\boldsymbol{\alpha} = \begin{bmatrix} & \mathbf{I}_3 & & \mathbf{0}_{3\times(n-3)} \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \mathbf{0}_{1\times(n-3)} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{n,1} & \alpha_{n,2} & \alpha_{n,3} & \mathbf{0}_{1\times(n-3)} \end{bmatrix}$$

$$\mathbf{P}_{L,HT} = \begin{bmatrix} \mathbf{r}_{1,HT} & \mathbf{r}_{2,HT} & \mathbf{r}_{3,HT} & \mathbf{0}_{2\times(n-3)} \end{bmatrix}^T$$

Therefore, $\left[(\mathbf{I} - \mathbf{W})^{-1} - \boldsymbol{\alpha}\right]\mathbf{P}_{L,HT} = 0$

### B.1 Maximizing the Global Deviation Norm

Maximizing the square of the norm is equivalent to maximizing the norm, so:

$$
\begin{aligned}
&\|\mathbf{r}_{i,HT} - \mathbf{r}_i\|^2 \\
&= \left[\sum_{m=1}^{n} \mathbf{K}_{i,m}\eta_m \cos\beta_m\right]^2 + \left[\sum_{m=1}^{n} \mathbf{K}_{i,m}\eta_m \sin\beta_m\right]^2 \\
&= \sum_{m=1}^{n} \left(\mathbf{K}_{i,m}\eta_m \cos\beta_m\right)^2 + \sum_{m=1}^{n} \left(\mathbf{K}_{i,m}\eta_m \sin\beta_m\right)^2 \\
&+ 2\sum_{m=1}^{n}\sum_{t=m+1}^{n} \left(\mathbf{K}_{i,m}\eta_m \cos\beta_m\right)\left(\mathbf{K}_{i,t}\eta_t \cos\beta_t\right) \\
&+ 2\sum_{m=1}^{n}\sum_{t=m+1}^{n} \left(\mathbf{K}_{i,m}\eta_m \sin\beta_m\right)\left(\mathbf{K}_{i,t}\eta_t \sin\beta_t\right) \\
&= \sum_{m=1}^{n} \mathbf{K}_{i,m}^2\eta_m^2 \\
&+ 2\sum_{m=1}^{n}\sum_{t=m+1}^{n} \mathbf{K}_{i,m}\mathbf{K}_{i,t}\eta_m\eta_t \left(\cos\beta_m \cos\beta_t + \sin\beta_m \sin\beta_t\right)
\end{aligned}
$$

Since all entries of $\mathbf{K}$ are non-negative, this expression is maximized when $\beta_m = \beta_t \forall m, t$ and $\eta_m = \delta_{m,l} \forall m$. Thus,

$$
\begin{aligned}
\delta_{g,i} &= \sqrt{\sum_{m=1}^{n} \mathbf{K}_{i,m}^2\delta_{m,l}^2 + 2\sum_{m=1}^{n}\sum_{t=m+1}^{n} \mathbf{K}_{i,m}\mathbf{K}_{i,t}\delta_{m,l}\delta_{t,l}} \\
&= \sqrt{\left(\sum_{m=1}^{n} \mathbf{K}_{i,m}\delta_{m,l}\right)^2} \\
&= \sum_{m=1}^{n} \mathbf{K}_{i,m}\delta_{m,l}
\end{aligned}
$$

and then $\mathbf{d}_g = \mathbf{K}\mathbf{d}_l$.

# BIBLIOGRAPHY


[1] "Unmanned aircraft system traffic management (utm)," https://www.faa.gov/uas/research_development/traffic_management/, 2021, accessed: 2022-03-17.

[2] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics& Automation Magazine*, vol. 19, no. 4, pp. 72–82, 12 2012.

[3] D. C. De Oliveira and M. A. Wehrmeister, "Using deep learning and low-cost rgb and thermal cameras to detect pedestrians in aerial images captured by multirotor uav," *Sensors*, vol. 18, no. 7, p. 2244, 2018.

[4] A. E. Jimenez-Cano, P. J. Sanchez-Cuevas, P. Grau, A. Ollero, and G. Heredia, "Contact-based bridge inspection multirotors: Design, modeling, and control considering the ceiling effect," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3561–3568, 2019.

[5] R. A. Stuckey, "Mathematical modelling of helicopter slung-load systems," DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION VICTORIA (AUSTRALIA . . . , Tech. Rep., 2001.

[6] Z. Li, Z. Duan, W. Ren, and G. Feng, "Containment control of linear multi-agent systems with multiple leaders of bounded inputs using distributed continuous controllers," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 13, pp. 2101–2121, 2015.

[7] C. B. Low and Q. San Ng, "A flexible virtual structure formation keeping control for fixed-wing uavs," in *Control and Automation (ICCA), 2011 9th IEEE Intl. Conf. on*. IEEE, 2011, pp. 621–626.

[8] Y. Cao, D. Stuart, W. Ren, and Z. Meng, "Distributed containment control for multiple autonomous vehicles with double-integrator dynamics: algorithms and experiments," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 4, pp. 929–938, 2011.

[9] H. Rastgoftar, *Continuum deformation of multi-agent systems*. Springer, 2016.

[10] J. Geng and J. W. Langelaan, "Cooperative transport of a slung load using load-leading control," *Journal of Guidance, Control, and Dynamics*, p. 1–19, Mar 2020. [Online]. Available: http://dx.doi.org/10.2514/1.G004680

[11] T. Lee, "Geometric control of quadrotor uavs transporting a cable-suspended rigid body," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, p. 255–264, Jan 2018. [Online]. Available: http://dx.doi.org/10.1109/TCST.2017.2656060

[12] H. Rastgoftar and E. M. Atkins, "Cooperative aerial payload transport guided by an in situ human supervisor," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 4, pp. 1452–1467, 2018. [Online]. Available: https://doi.org/10.1109/TCST.2018.2832598

[13] A. A. Neto, D. G. Macharet, and M. F. Campos, "Feasible RRT-based path planning using seventh order Bézier curves," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 1445–1450, 2010.

[14] Z. Liang, H. Rastgoftar, and E. M. Atkins, "Multi-quadcopter team leader path planning using particle swarm optimization," Jun 2019. [Online]. Available: http://dx.doi.org/10.2514/6.2019-3258

[15] M. Romano, "M330-quadrotor documentation," https://bit.ly/3Dkmm7T, 2021.

[16] beagleboard, "librobotcontrol," https://github.com/beagleboard/librobotcontrol, 2014.

[17] A. Chovancová, T. Fico, P. Hubinskỳ, and F. Duchoň, "Comparison of various quaternion-based control methods applied to quadrotor with disturbance observer and position estimator," *Robotics and Autonomous Systems*, vol. 79, pp. 87–98, 2016. [Online]. Available: https://doi.org/10.1016/j.robot.2016.01.011

[18] E. Fresk and G. Nikolakopoulos, "Full quaternion based attitude control for a quadrotor," in *2013 European control conference (ECC)*. IEEE, 2013, pp. 3864–3869. [Online]. Available: https://doi.org/10.23919/ECC.2013.6669617

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

[20] M. Romano, Y. Chen, P. Kuevor, O. Marshall, and E. Atkins, "Nailed it: Autonomous roofing with a nailgun-equipped octocopter," in *AIAA AVIATION 2021 FORUM*, 2021, p. 3211.

[21] M. Romano, "Tethered payload," https://bit.ly/3BVfX43, 2021.

[22] J. Qin, W. X. Zheng, H. Gao, Q. Ma, and W. Fu, "Containment control for second-order multiagent systems communicating over heterogeneous networks," *IEEE transactions on neural networks and learning systems*, 2017.

[23] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Systems*, vol. 27, no. 2, pp. 71–82, 2007.

[24] H. Rastgoftar and S. Jayasuriya, "Evolution of multi-agent systems as continua," *J. Dynamic Sys., Meas., & Control*, vol. 136, no. 4, 2014.

[25] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2013.

[26] N. Rahbari-Asr, U. Ojha, Z. Zhang, and M.-Y. Chow, "Incremental welfare consensus algorithm for cooperative distributed generation/demand response in smart grid," *IEEE Transactions on Smart Grid*, vol. 5, no. 6, pp. 2836–2845, 2014.

[27] H. Rastgoftar and E. M. Atkins, "Continuum deformation of multi-agent systems under directed communication topologies," *Journal of Dynamic Systems, Measurement, and Control*, vol. 139, no. 1, 2017.

[28] W. Ren and R. W. Beard, "Decentralized scheme for spacecraft formation flying via the virtual structure approach," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 1, pp. 73–82, 2004.

[29] Y. Liu and Z. Geng, "Finite-time formation control for linear multi-agent systems: A motion planning approach," *Systems & Control Letters*, vol. 85, pp. 54–60, 2015.

[30] Z. Yang, J. Xiang, and Y. Li, "Distributed consensus based supply-demand balance algorithm for economic dispatch problem in a smart grid with switching graph," *IEEE Trans. on Indust. Electronics*, 2016.

[31] S. Rao and D. Ghose, "Sliding mode control-based autopilots for leaderless consensus of unmanned aerial vehicles," *IEEE transactions on control systems technology*, vol. 22, no. 5, pp. 1964–1972, 2014.

[32] W. Ren, "Distributed leaderless consensus algorithms for networked euler–lagrange systems," *Intl. J. of Control*, vol. 82, no. 11, 2009.

[33] J. Han and Y. Chen, "Multiple uav formations for cooperative source seeking and contour mapping of a radiative signal field," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 323–332, 2014.

[34] J. A. Guerrero, P. Castillo, S. Salazar, and R. Lozano, "Mini rotorcraft flight formation control using bounded inputs," *Journal of intelligent & robotic systems*, vol. 65, no. 1-4, pp. 175–186, 2012.

[35] Q. Song, J. Cao, and W. Yu, "Second-order leader-following consensus of nonlinear multi-agent systems via pinning control," *Systems & Control Letters*, vol. 59, no. 9, pp. 553–562, 2010.

[36] L. Xu, H. Fan, Z. Dong, and W. Wang, "Robust consensus control for leader-following multi-agent system under switching topologies," in *2016 International Conference on Cybernetics, Robotics and Control (CRC)*. IEEE, 2016, pp. 27–31.

[37] A. Drak, M. Hejase, M. ElShorbagy, A. Wahyudie, and H. Noura, "Autonomous formation flight algorithm and platform for quadrotor uavs," *Intl. J. of Robotics & Mech.*, vol. 1, no. 4, pp. 124–132, 2014.

[38] T. Namerikawa, Y. Kuriki, and A. Khalifa, "Consensus-based cooperative formation control for multiquadcopter system with unidirectional network connections," *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 4, p. 044502, 2018.

[39] N. Nigam, S. Bieniawski, I. Kroo, and J. Vian, "Control of multiple uavs for persistent surveillance: algorithm and flight test results," *IEEE Trans. on Cont. Systems Tech.*, vol. 20, no. 5, pp. 1236–1251, 2012.

[40] J. Han, Y. Xu, L. Di, and Y. Chen, "Low-cost multi-uav technologies for contour mapping of nuclear radiation field," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 401–410, 2013.

[41] H. Rastgoftar, H. G. Kwatny, and E. M. Atkins, "Asymptotic tracking and robustness of mas transitions under a new communication topology," *IEEE Trans. on Automation Science and Engineering*, 2016.

[42] P. Holoborodko, "Smooth noise robust differentiators," http://www.holoborodko.com/pavel/numerical-methods/numerical-derivative/smooth-low-noise-differentiators/, 2008.

[43] G. Wu and K. Sreenath, "Geometric control of multiple quadrotors transporting a rigid-body load," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6141–6148. [Online]. Available: https://doi.org/10.1109/CDC.2014.7040351

[44] T. Lee, K. Sreenath, and V. Kumar, "Geometric control of cooperating multiple quadrotor uavs with a suspended payload," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 5510–5515. [Online]. Available: https://doi.org/10.1109/CDC.2013.6760757

[45] H. Rastgoftar and E. M. Atkins, "Cooperative aerial lift and manipulation (calm)," *Aerospace Science and Technology*, vol. 82–83, p. 105–118, Nov 2018. [Online]. Available: http://dx.doi.org/10.1016/j.ast.2018.09.005

[46] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, no. 1, p. 73–86, Sep 2010. [Online]. Available: http://dx.doi.org/10.1007/s10514-010-9205-0

[47] K. Sreenath and V. Kumar, "Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots," *rn*, vol. 1, no. r2, p. r3, 2013.

[48] J. Enciu and J. F. Horn, "Flight performance optimization of a multilift rotorcraft formation," *Journal of Aircraft*, vol. 54, no. 4, pp. 1521–1538, 2017. [Online]. Available: https://doi.org/10.2514/1.C034111

[49] M. D. Takahashi, M. S. Whalley, M. G. Berrios, and G. J. Schulein, "Flight validation of a system for autonomous rotorcraft multilift," *Journal of the American Helicopter Society*, vol. 64, no. 3, pp. 1–13, 2019. [Online]. Available: https://doi.org/10.4050/JAHS.64.032001

[50] R. M. Murray, "Recent Research in Cooperative Control of Multivehicle Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 571–583, 05 2007. [Online]. Available: https://doi.org/10.1115/1.2766721

[51] M. Ji, G. Ferrari-Trecate, M. Egerstedt, and A. Buffa, "Containment control in mobile networks," *IEEE Transactions on Automatic Control*, vol. 53, no. 8, pp. 1972–1975, 2008. [Online]. Available: https://doi.org/10.1109/TAC.2008.930098

[52] C. B. Low and Q. S. Ng, "A flexible virtual structure formation keeping control for fixed-wing uavs," in *2011 9th IEEE International Conference on Control and Automation (ICCA)*, 2011, pp. 621–626. [Online]. Available: https://doi.org/10.1109/ICCA.2011.6137876

[53] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71–82, 2007. [Online]. Available: https://doi.org/10.1109/MCS.2007.338264

[54] H. Rastgoftar and E. M. Atkins, "Continuum Deformation of Multi-Agent Systems Under Directed Communication Topologies," *Journal of Dynamic Systems, Measurement, and Control*, vol. 139, no. 1, 09 2016, 011002. [Online]. Available: https://doi.org/10.1115/1.4033866

[55] I. Maza, K. Kondak, M. Bernard, and A. Ollero, "Multi-uav cooperation and control for load transportation and deployment," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1–4, p. 417–449, Aug 2009. [Online]. Available: http://dx.doi.org/10.1007/s10846-009-9352-8

[56] F. A. Goodarzi and T. Lee, "Stabilization of a rigid body payload with multiple cooperative quadrotors," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 12, Aug 2016. [Online]. Available: http://dx.doi.org/10.1115/1.4033945

[57] A. Tagliabue, M. Kamel, R. Siegwart, and J. Nieto, "Robust collaborative object transportation using multiple mavs," *The International Journal of Robotics Research*, vol. 38, no. 9, p. 1020–1044, Jun 2019. [Online]. Available: http://dx.doi.org/10.1177/0278364919854131

[58] M. Zürn, K. Morton, A. Heckmann, A. McFadyen, S. Notter, and F. Gonzalez, "Mpc controlled multirotor with suspended slung load: System architecture and visual load detection," in *2016 IEEE Aerospace conference*. IEEE, 2016, pp. 1–11. [Online]. Available: https://doi.org/10.1109/AERO.2016.7500543

[59] D. E. Whitney, "Force feedback control of manipulator fine motions," *Journal of Dynamic Systems, Measurement, and Control*, vol. 99, no. 2, p. 91–97, Jun 1977. [Online]. Available: http://dx.doi.org/10.1115/1.3427095

[60] A. Q. Keemink, H. van der Kooij, and A. H. Stienen, "Admittance control for physical human–robot interaction," *The International Journal of Robotics Research*, vol. 37, no. 11, p. 1421–1444, Apr 2018. [Online]. Available: http://dx.doi.org/10.1177/0278364918768950

[61] L. Bascetta, G. Ferretti, G. Magnani, and P. Rocco, "Walk-through programming for robotic manipulators based on admittance control," *Robotica*, vol. 31, no. 7, p. 1143–1153, May 2013. [Online]. Available: http://dx.doi.org/10.1017/S0263574713000404

[62] C. Carignan, J. Tang, and S. Roderick, "Development of an exoskeleton haptic interface for virtual task training," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3697–3702. [Online]. Available: https://doi.org/10.1109/IROS.2009.5354834

[63] W. Huo, J. Huang, Y. Wang, J. Wu, and L. Cheng, "Control of upper-limb power-assist exoskeleton based on motion intention recognition," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2243–2248. [Online]. Available: https://doi.org/10.1109/ICRA.2011.5980483

[64] H. Kim, L. M. Miller, Z. Li, J. R. Roldan, and J. Rosen, "Admittance control of an upper limb exoskeleton - reduction of energy exchange," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2012, pp. 6467–6470. [Online]. Available: https://doi.org/10.1109/EMBC.2012.6347475

[65] F. Augugliaro and R. D'Andrea, "Admittance control for physical human-quadrocopter interaction," in *2013 European Control Conference (ECC)*, 2013, pp. 1805–1810. [Online]. Available: https://doi.org/10.23919/ECC.2013.6669643

[66] K. Kaneko, F. Kanehiro, M. Morisawa, E. Yoshida, and J.-P. Laumond, "Disturbance observer that estimates external force acting on humanoid robots," in *2012 12th IEEE International Workshop on Advanced Motion Control (AMC)*, 2012, pp. 1–6. [Online]. Available: https://doi.org/10.1109/AMC.2012.6197026

[67] B. Yüksel, C. Secchi, H. H. Bülthoff, and A. Franchi, "A nonlinear force observer for quadrotors and application to physical interactive tasks," in *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2014, pp. 433–440. [Online]. Available: https://doi.org/10.1109/AIM.2014.6878116

[68] S. Waslander and C. Wang, "Wind disturbance estimation and rejection for quadrotor position control," in *AIAA Infotech@Aerospace Conference*. American Institute of Aeronautics and Astronautics, Apr 2009, pp. 1–14. [Online]. Available: http://dx.doi.org/10.2514/6.2009-1983

[69] D. Lee, K. D. Kumar, and M. Sinha, "Fault detection and recovery of spacecraft formation flying using nonlinear observer and reconfigurable controller," *Acta Astronautica*, vol. 97, p. 58–72, Apr 2014. [Online]. Available: http://dx.doi.org/10.1016/j.actaastro.2013.12.002

[70] R. G. Lyons, *Understanding digital signal processing, 3/E*. Pearson, 2004.

[71] L. Meier, "Mavlink – micro air vehicle message marshalling library." https://github.com/mavlink/mavlink, 2009.

[72] R. W. Beard, "Quadrotor dynamics and control," *Brigham Young University*, vol. 19, no. 3, pp. 46–56, 2008.

[73] M. Romano, "coop_payload_sim," https://gitlab.eecs.umich.edu/mmroma/coop_payload_sim, 2020.

[74] P. P. Brown and D. F. Lawler, "Sphere drag and settling velocity revisited," *Journal of environmental engineering*, vol. 129, no. 3, pp. 222–231, 2003. [Online]. Available: https://doi.org/10.1061/(ASCE)0733-9372(2003)129:3(222)

[75] B. Davoudi, E. Taheri, K. Duraisamy, B. Jayaraman, and I. Kolmanovsky, "Quad-rotor flight simulation in realistic atmospheric conditions," *AIAA Journal*, vol. 58, no. 5, p. 1992–2004, May 2020. [Online]. Available: http://dx.doi.org/10.2514/1.J058327

[76] J. Svacha, K. Mohta, and V. Kumar, "Improving quadrotor trajectory tracking by compensating for aerodynamic effects," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 860–866. [Online]. Available: https://doi.org/10.1109/ICUAS.2017.7991501

[77] M. Romano, "Cooperative transportation of a slung load using haptic guidance," https://www.youtube.com/watch?v=22yr16A7ELs, 2021.

[78] D. Romero, "California had nation's worst fire season in 2018," https://www.nbcnews.com/news/us-news/california-had-nation-s-worst-fire-season-2018-n981431, 2018, accessed: 2020-09-14.

[79] J. Hempstead, "Afrl - swarm and search ai competition," https://www.wbi-innovates.com/blogs/post/2019-swarm-search-ai-challenge-fire-hack-press-kit, 2020, accessed: 2020-09-14.

[80] M. Duquette, "Effects-level models for UAV simulation," in *AIAA Modeling and Simulation Technologies Conference*. American Institute of Aeronautics and Astronautics, Aug 2009. [Online]. Available: https://doi.org/10.2514/6.2009-6139

[81] "Github - amase software," https://github.com/afrl-rq/OpenAMASE, 2020, accessed: 2020-09-14.

[82] I. Maza, A. Ollero, E. Casado, and D. Scarlatti, "Classification of multi-UAV architectures," in *Handbook of Unmanned Aerial Vehicles*. Springer Netherlands, Aug. 2014, pp. 953–975. [Online]. Available: https://doi.org/10.1007/978-90-481-9707-1_119

[83] H. X. Pham, H. M. La, D. Feil-Seifer, and M. Deans, "A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sep 2017, pp. 6648–6653. [Online]. Available: http://dx.doi.org/10.1109/IROS.2017.8206579

[84] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[85] S. Ismail and L. Sun, "Decentralized hungarian-based approach for fast and scalable task allocation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 23–28.

[86] L. Merino, F. Caballero, J. R. Martínez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple uavs: Application to automatic detection of forest fires," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 165–184, 2006.

[87] P. Schmuck and M. Chli, "Multi-UAV collaborative monocular SLAM," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017, pp. 3863–3870. [Online]. Available: https://doi.org/10.1109/icra.2017.7989445

[88] J. Gancet, G. Hattenberger, R. Alami, and S. Lacroix, "Task planning and control for a multi-uav system: architecture and algorithms," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 1017–1022. [Online]. Available: http://dx.doi.org/10.1109/IROS.2005.1545217

[89] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[90] M. Shanmugavel, A. Tsourdos, B. A. White, and R. Żbikowski, "Differential Geometric Path Planning of Multiple UAVs," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 620–632, 01 2007. [Online]. Available: https://doi.org/10.1115/1.2767657

[91] S. Hota and D. Ghose, "Optimal geometrical path in 3d with curvature constraint," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 113–118.

[92] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a dubins airplane," in *Decision and Control, 2007 46th IEEE Conference on*. Piscataway, NJ: IEEE, 2007, pp. 2379–2384.

[93] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[94] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 3067–3074.

[95] E. I. Grøtli and T. A. Johansen, "Path planning for UAVs under communication constraints using SPLAT! and MILP," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 65, no. 1-4, pp. 265–282, 2012.

[96] J. L. Foo, J. Knutzon, V. Kalivarapu, J. Oliver, and E. Winer, "Path planning of unmanned aerial vehicles using b-splines and particle swarm optimization," *Journal of aerospace computing, Information, and communication*, vol. 6, no. 4, pp. 271–290, 2009.

[97] H.-b. Duan, X.-y. Zhang, J. Wu, and G.-j. Ma, "Max-min adaptive ant colony optimization approach to multi-uavs coordinated trajectory replanning in dynamic and uncertain environments," *Journal of Bionic Engineering*, vol. 6, no. 2, pp. 161–173, 2009.

[98] M. Hwangbo, J. Kuffner, and T. Kanade, "Efficient two-phase 3d motion planning for small fixed-wing uavs," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 1035–1041.

[99] M. Quigley, B. Barber, S. Griffiths, and M. Goodrich, "Towards real-world searching with fixed-wing mini-UAVs," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3028–3033. [Online]. Available: http://ieeexplore.ieee.org/document/1545051/

[100] Z. Qi, Z. Shao, Y. S. Ping, L. M. Hiot, and Y. K. Leong, "An improved heuristic algorithm for UAV path planning in 3D environment," *Proceedings - 2010 2nd International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2010*, vol. 2, pp. 258–261, 2010.

[101] X. Li, J. Xie, M. Cai, M. Xie, and Z. Wang, "Path planning for UAV based on improved heuristic A* algorithm," *ICEMI 2009 - Proceedings of 9th International Conference on Electronic Measurement and Instruments*, vol. 3, pp. 3488–3493, 2009.

[102] Wright Brothers Institute, "AFRL - Automatic Collision Avoidance Technology (ACAT)," https://www.wpafb.af.mil/Welcome/Fact-Sheets/Display/Article/1578307/afrl-automatic-collision-avoidance-technology-acat/, 2020, accessed: 2020-09-14.

[103] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, p. 1258–1276, Dec 2013. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2013.09.004

[104] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113–126, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1016639210559

[105] A. Khan, I. Noreen, and Z. Habib, "On complete coverage path planning algorithms for non-holonomic mobile robots: Survey and challenges," *J. Inf. Sci. Eng.*, vol. 33, pp. 101–121, 2017.

[106] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient boustrophedon multi-robot coverage: an algorithmic approach," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2, pp. 109–142, Apr 2008. [Online]. Available: https://doi.org/10.1007/s10472-009-9120-2

[107] A. Ahmadzadeh, J. Keller, G. Pappas, A. Jadbabaie, and V. Kumar, "An optimization-based approach to time-critical cooperative surveillance and coverage with uavs," in *Experimental Robotics: The 10th International Symposium on Experimental Robotics*, O. Khatib, V. Kumar, and D. Rus, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 491–500. [Online]. Available: https://doi.org/10.1007/978-3-540-77457-0_46

[108] I. Maza and A. Ollero, "Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in *Distributed Autonomous Robotic Systems 6*, R. Alami, R. Chatila, and H. Asama, Eds. Tokyo: Springer Japan, 2007, pp. 221–230.

[109] M. A. Batalin and G. S. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *Distributed Autonomous Robotic Systems 5*, H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, Eds. Tokyo: Springer Japan, 2002, pp. 373–382.

[110] D. C. Guastella, L. Cantelli, G. Giammello, C. D. Melita, G. Spatino, and G. Muscato, "Complete coverage path planning for aerial vehicle flocks deployed in outdoor environments," *Computers & Electrical Engineering*, vol. 75, pp. 189–201, 2019. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1112541809

[111] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[112] "Usgs - earth explorer," https://earthexplorer.usgs.gov/, 2020, accessed: 2020-09-14.

[113] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 500–505.

[114] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[115] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using laplace's equation," in *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE, 1990, pp. 2102–2106.

[116] M. Romano, P. Kuevor, D. Lukacs, O. Marshall, M. Stevens, H. Rastgoftar, J. Cutler, and E. Atkins, "Experimental evaluation of continuum deformation with a five quadrotor team," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 2023–2029.

[117] Y. Liu and R. Bucknall, "A survey of formation control and motion planning of multiple unmanned vehicles," *Robotica*, vol. 36, no. 7, p. 1019–1047, 2018.

[118] H. Rastgoftar, E. M. Atkins, and I. V. Kolmanovsky, "Scalable vehicle team continuum deformation coordination with eigen decomposition," *IEEE Transactions on Automatic Control*, vol. 67, no. 5, pp. 2514–2521, 2022.

[119] H. Rastgoftar, "Integration of a* search and classic optimal control for safe planning of continuum deformation of a multi-quadcopter system," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–1, 2022.

[120] M. N. Stevens and E. M. Atkins, "Layered geofences in complex airspace environments," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3348.

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