

Network Decentralized Collision Avoidance with Applications in a Scalable Unmanned Aerial System Testbed

P.W. Ledzian

Master of Science Thesis

Network Decentralized Collision Avoidance with Applications in a Scalable Unmanned Aerial System Testbed

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Systems and Control at Delft
University of Technology

P.W. Ledzian

July 26, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of
Technology

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF
DELFT CENTER FOR SYSTEMS AND CONTROL (DCSC)

The undersigned hereby certify that they have read and recommend to the Faculty of
Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis
entitled

NETWORK DECENTRALIZED COLLISION AVOIDANCE WITH APPLICATIONS IN A
SCALABLE UNMANNED AERIAL SYSTEM TESTBED

by

P.W. LEDZIAN

in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE SYSTEMS AND CONTROL

Dated: July 26, 2019

Supervisor(s):

Dr. ir. G. Giordano

Reader(s):

Prof. dr. ir M. Verhaegen

Dr.ir. K. Batselier

Dr. B. Shyrokau

Abstract

Decentralized control and estimation are both active research areas in the field of systems and control. A new approach to these topics utilizes graph theory to characterize inter-agent communication as a graph that, in this thesis, can have time-varying topology. This approach has been named "network-decentralized" and the use of network-decentralized control and estimation can enable multi-agent systems to achieve tasks in low information environments. In this thesis a novel network-decentralized control algorithm is proposed to enable collision avoidance and formation producing behavior in a multi-agent system. Additionally, a network-decentralized estimation algorithm is also proposed that is combined with the network-decentralized control approach yielding a first of its kind network-decentralized network-estimated control algorithm. A multitude of simulation environments are developed to test the algorithm in a 2-D holonomic multi-agent environment and a parameter tuning method is presented. Strong performance is shown in 2-D with collision avoidance guarantees, at the cost of difficult to tune parameters. The use of holonomic agents in 2-D is consistent with most research in decentralized control and estimation. Since this type of agent is common, this thesis extends its work to the 3-D non-holonomic agent case in an attempt to help characterize how accurate, scalable, and useful existing decentralized research is for real world applications. To develop the network-decentralized algorithm to this point, it is first expanded to the holonomic 3-D case where it is further tested in newly developed simulation environments, where again, collision avoidance guarantees are provided. A novel free-space metric is introduced which allows for tests to be compared across dimensions (eg: 2-D and 3-D) and also yields additional insight as to when parameter-performance breakdown will occur. For the extension of the algorithm to a non-holonomic vehicle in 3-D a suitable quadcopter platform is designed. In addition, the Delft Center for Systems and Control (DCSC) Distributed Robotics Lab is re-designed and completely virtualized to allow for students to easily access lab resources, such as motion capture (MoCap) information, quadcopter test-code environments, and documentation all in one location. A first-principles based system identification process is presented and used to identify the designed quadcopter platform so that a suitable controller can be found for in-lab use. Two quadcopter simulation environments are developed to allow for the design of hover-envelope controllers based on identified parameters. Since the DCSC department was unable to acquire multiple physical vehicles for swarm testing the

system identification process and quadcopter simulation environments are validated against the test results from a single physical quadcopter in the newly designed lab environment. A more comprehensive multi-agent implementation of the collision avoidance portion of the algorithm is tested in a real-time vehicle simulation environment. It is found that while the single physical agent can closely track the behavior of holonomic agents and quadcopter simulation results, the real-time multi-agent case exhibits a degradation in performance that further declines in obstacle dense environments. This explains a loss of performance that can lead to collisions in the real-time setting where in the holonomic case the exact same test yields successful results.

Table of Contents

Acknowledgements	xi
1 Introduction	1
2 Network-Decentralized Approaches for Multi-Agent Systems	5
2-1 The 2-D Holonomic Agent	5
2-2 2-D Network-Decentralized Estimation	8
2-3 2-D Network-Decentralized Collision Avoidance	12
2-4 Analysis and Identification of Key Algorithm Parameters	14
3 2-D Holonomic Simulations	17
3-1 Baseline Parameter Evaluation via Synchronous Simulation	17
3-2 Improving Collision Avoidance	20
3-2-1 Continuously Defined Detection Regions	21
3-2-2 Selecting an Appropriate Avoidance Angle	22
3-2-3 Tuning Network-Decentralized Estimation Parameters	26
3-3 Conclusions	31
4 3-D Holonomic Simulations	33
4-1 Network-Decentralized Network-Estimated Collision Avoidance in 3-D	33
4-2 Evaluation of the Extension to 3-D	34
4-3 Conclusions	40
5 Lab and Vehicle Design	41
5-1 Lab Environment: Closing the Loop	41
5-1-1 High Level Overview	41
5-1-2 DCSC Distributed Robotics Lab: Original Design	45
5-1-3 Lab Re-design	46
5-2 Vehicle Selection	50
5-3 Flight Controller Selection	53

6	Quadcopter Dynamics and Identification	57
6-1	Quadcopter Dynamics and Control	57
6-1-1	Dynamics	57
6-1-2	Control	62
6-2	System Identification	64
7	Quadcopter – Testing	71
7-1	Single Agent Performance	71
7-1-1	Takeoff and Hover	71
7-1-2	Waypointing	73
7-1-3	Circling	75
7-2	Multi-Agent Collision Avoidance	76
7-2-1	Two Agents Collision Avoidance – Planar	77
7-2-2	Two Agents Collision Avoidance – Altitude Varying	78
7-2-3	Six Agents Collision Avoidance – Sphere	79
7-2-4	Six Agents Collision Avoidance – Planar	81
8	Conclusions and Future Work	83
8-1	Conclusions	83
8-2	Future Work	85
	Bibliography	87
	Glossary	93
	List of Acronyms	93
	List of Symbols	95

List of Figures

1-1	One of many formations that birds create while flocking [1].	2
2-1	A 2-D holonomic agent with radius R	6
2-2	Speed limiting via vector projection	7
2-3	A 2-D holonomic agent with two distinct collision detection regions.	8
2-4	The 2-D agent collision avoidance velocity update.	9
2-5	A basic undirected communication graph.	10
3-1	A simplified diagram of the system, with the indication of some delays that are present in a physical implementation of the simulation.	18
3-2	A 2-D network-estimated network-decentralized test.	19
3-3	Continuous projection definition of the collision avoidance window.	21
3-4	Full information collision avoidance in 2-D with holonomic agents, pt. 1.	22
3-5	Full information collision avoidance in 2-D with holonomic agents, pt. 2.	23
3-6	Example of an avoidance vector tangent to R_{pref_dist}	24
3-7	Agent required heading change for avoidance at different distances.	25
3-8	How changes in β impact agent responsiveness.	26
3-9	Observer performance of two agents.	27
3-10	Observer performance of all agents.	28
3-11	Agent behavior comparison of dynamic vs fixed communication radius.	29
3-12	Fixed communication radius, $R_{comm} = 4.2$, observer behavior.	29
3-13	Fixed communication radius, $R_{comm} = 2.2$, observer behavior.	30
3-14	Traveling circle formation, proposed "worst case."	30
4-1	Fibonacci sphere algorithm initialization of 100 agents.	35
4-2	Fibonacci sphere algorithm projections.	36

4-3	Fully network decentralized traveling sphere test with 10 agents, part 1.	36
4-4	Fully network decentralized traveling sphere test with 10 agents, part 2.	37
5-1	Massachusetts Institute of Technology (MIT)'s Real-time indoor Autonomous Vehicle test ENvironment (RAVEN) lab with radio controlled (RC) transmitters and MoCap system [2].	42
5-2	University of Pennsylvania (UPenn)'s general robotics, automation, sensing, and perception (GRASP) lab quadcopter acrobatics [3]	43
5-3	ETH Zurich's Flying Machine Arena (FMA) testbed. [4]	44
5-4	A typical indoor quadcopter lab for autonomous control testing.	45
5-5	Robotics test area with protective netting.	45
5-6	Position and trajectory computation setup.	46
5-7	Typical operating system (OS) vs hypervisor types [5].	47
5-8	The entire virtualized lab environment hosted on a Dell r740 server.	49
5-9	Ascending Technologies Hummingbird platform [6]	50
5-10	Draganflyer V Ti Pro [7]	51
5-11	Lumenier QAV-R2 frame [8]	52
5-12	Crazyflie 2.1 vehicle [9]	54
5-13	Quadcopter hover envelope control logic.	55
6-1	Diagram of a quadcopter with two reference frames: inertial (E) and body (B) [10].	58
6-2	Graphic detailing the thrust force offset caused by the rotor flapping effect [11]. .	61
6-3	Nested PID control loop for quadcopter vehicle [12].	63
6-4	computer aided design (CAD) design of the QAV-R2 quadcopter	65
6-5	Motor testbench to collect data for system identification.	66
6-6	Thrust and torque constants during a ramp test of a 2206-2300KV motor.	67
6-7	Thrust and torque forces vs motor speeds with a second-order polynomial of best fit.	67
6-8	Information flow from the electronic speed controller (ESC) input to the final position/orientation.	68
6-9	A size, weight, and power (SWAP) plot denoting performance for the 2206-2300KV motor.	69
7-1	Crazyflie 2.1 hovering at 1.0 meters.	72
7-2	Plot of the Crazyflie position over time during the hover test.	72
7-3	Comparison of physical Crazyflie 2.1 and various simulations in a square pattern.	73
7-4	Comparison of physical Crazyflie 2.1 and various simulations in an altitude varying square pattern.	74
7-5	Comparison of physical Crazyflie 2.1 and various simulations in a circling pattern.	75
7-6	Developed user interface with Gazebo using ROS libraries.	76
7-7	Gazebo environment of two vehicles that will avoid on the same plane.	77
7-8	Two agents avoiding in real-time on the same plane.	78

7-9	Two agents avoiding in real-time in an altitude varying setting.	79
7-10	Two agents avoiding in real-time in an altitude varying setting, another perspective.	79
7-11	Initialization of a 6-agent sphere formation.	80
7-12	Holonomic vs quadcopter simulation, 6-agent sphere avoidance performance.	80
7-13	Initialization of a 6-agent formation on a single plane.	81
7-14	Holonomic vs quadcopter simulation, 6-agent planar collision avoidance performance.	82

List of Tables

2-1	Individual agent tuning parameters	14
2-2	Network-decentralized tuning parameters	14
3-1	Agent initial points and setpoints	18
3-2	Baseline test parameters	19
3-3	Breakdown of baseline test results without livelock breaking	19
3-4	Basic dual region test results without livelock breaking	20
3-5	Agent test parameters	23
3-6	Expanded detection zones, maximum avoidance maneuver box test	26
3-7	Updated agent test parameters	27
3-8	Worst case communication, $n_{conn} = 1$, results	31
4-1	Results for 10 agent 3-D sphere test	35
4-2	Measure of agent free space in 2-D and 3-D	38
4-3	Results for 25 agent 3-D sphere test	38
4-4	Results for 50 agent 3-D sphere test	39
4-5	Comparion of test characteristics and corresponding performance	39
5-1	Performance data for two different methods of accessing the lab environment. . .	48
6-1	List of parameters for first principles system identification	64
7-1	Crazyflie 2.1 hover test in the virtualized lab environment.	72
7-2	Comparison of mean-squared error between simulation and physical results in a simple square test.	73
7-3	Comparison of mean-squared error between simulations and physical tests in an altitude varying test.	74

7-4	Comparison of mean-squared error between simulations and physical tests in a circling test.	75
7-5	Holonomic vs quadcopter simulation, 2-agent planar avoidance performance. . .	77
7-6	Holonomic vs quadcopter simulation, 2-agent altitude varying performance. . .	78
7-7	Difference between the holonomic agent and quadcopter simulation sphere avoidance results.	81
7-8	Difference between the holonomic agent and quadcopter simulation planar avoidance results.	82

Acknowledgements

When I first set foot in the Netherlands two years ago I could not have imagined the journey that awaited me. It was my first time living in a foreign country, my first time in a new educational system, not to mention the long and difficult process to even qualify for studying abroad. I felt as if I had come so far, but I had not even begun my Systems and Control studies! I must say that I would not be here if not for the tireless and thoughtful efforts of my Fulbright mentor Dr. Alina Zapalska. She believed I had what it took to be a Fulbright Fellow long before I believed it myself. I also must thank the Electrical Engineering faculty that made a home for me during my four years at the US Coast Guard Academy. With a special thanks to Dr. Richard Hartnett and Dr. Paul Crilly, without their kindness, mentorship, and amusing stories many students, myself included, would not be half as successful as they are today. Of course I would not have survived the Academy without the lifelong friends I made along the way, c/o 2017 EEs you are my family. Julian Blanco, Kent Altobelli, Caleb Stewart, and Aaron Dahlen your persistence, passion, and pure creativity is what got me into tinkering and UAS work and I can't imagine my life without it. "This is fine."

While in Delft, my first six months of coursework were almost more than I could handle. Without constant support from my brother Austin Ledzian, over a well timed cup of tea, I'm certain I would not have been able to complete my studies. I also must thank the amazing friends I have made during my time in Delft: Julia, Leo, Daniél, and Alex, for great lunch conversations and those many hours we sat across from one another staring at our computers; Ola, Karol, Bart, and Máté, for our many great adventures, our friend group and my experience here would not be the same without you.

The road to completing a MSc. in Systems and Control on time in two years has been a challenging one. I am convinced that my advisor Dr. ir. G. Giordano was handcrafted for a student like myself. When I first met her I told her my number one priority was to graduate on time, and she said "then let's make it happen". Since then she has been an invaluable source of knowledge, timely feedback, and a tirelessly positive attitude, while also giving me the freedom to explore my own ideas and work independently. She definitely helped to make my experience in Delft a positive one. Lastly I am indebted to Dr. ir. Tamas Keviczky for providing my project with funding and to Ing. Will van Geest for his technical expertise and assistance in the lab.

Chapter 1

Introduction

The field of robotics is rapidly advancing in its development of dexterity, complex dynamics, and inter-agent cooperation. Human-like robotic hands are leveraging reinforcement learning to learn how to manipulate complex surfaces such as cubes and pencils [13]. Only a few years ago agile legged robots were a concept of science fiction. Recent developments have animal-like machines moving at up to $6 \frac{m}{s}$ and jumping [14]. Similar improvements have been seen with inter-agent cooperation. Entire warehouses are being automated similarly to the designs proposed by Kiva systems [15]. Despite these improvements different components, either within the same system or as part of multi-agent systems, must share large amounts of information in order to successfully complete tasks. To those familiar with the cooperative behavior of animals, these lofty communication requirements leave something to be desired. After all, schools of fish do not have leaders who command them to swarm; flocks of birds effortlessly avoid collisions, as in Figure 1-1; and ants can perform rudimentary cost-benefit analysis deciding which direction to travel at a fork in the road, or whether to build a living bridge out of their own bodies [16, 17, 18].

Instead of centralized communication, the coordinated group behavior of animals relies on each individual animal to make a small calculation based-on limited external sensing. This has the potential to yield complex global results, and for researchers that try to replicate this behavior in robotics, this is called decentralized control. There are many variations of this field of research, as outlined by Cao [19]. One promising avenue is decentralized estimation. This seeks to utilize local information sharing to determine global information. A potentially novel approach concerns the use of algebraic graph theory to describe time-varying communication behavior among agents in a multi-agent system. So called "network-decentralized" research has found application in both control and estimation [20, 21, 22, 23]. In this thesis network-decentralized approaches are extended for both control and estimation within a multi-agent system resulting in collision avoidance guarantees and formation producing behavior.

As stated before, this thesis seeks to develop and evaluate the combination of network-decentralized estimation and control. Most decentralized research focuses on holonomic single, and sometimes, double integrator representations of agents. The work done in this thesis starts with these common assumptions and agents in 2-D. Here an algorithm for collision

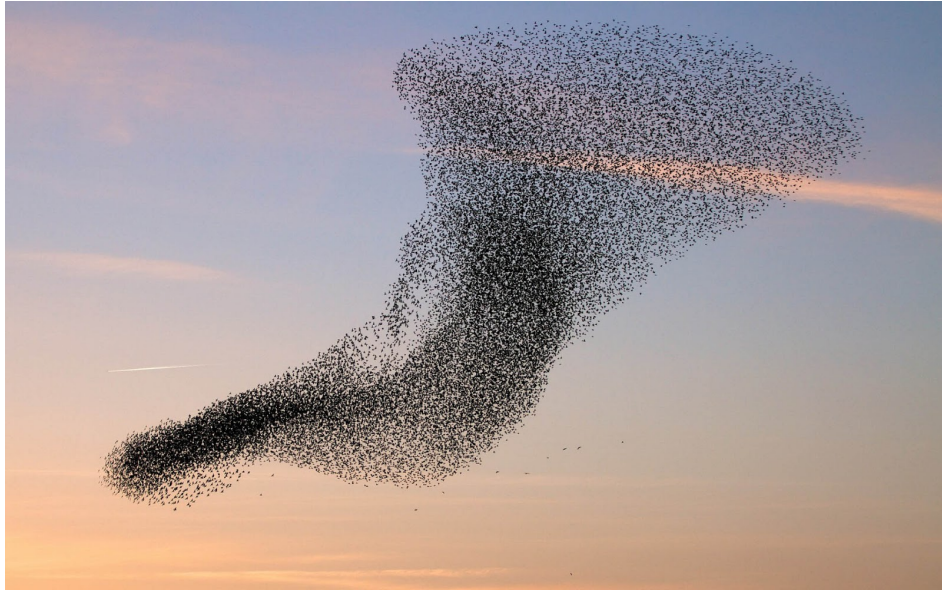


Figure 1-1: One of many formations that birds create while flocking [1].

avoidance and formation production is implemented leveraging network-decentralized control and estimation. The work is then extended to 3-D, and finally to a real-time vehicle swarm of quadcopters. This is done with a few goals in mind. First is to analyze how applicable current network-decentralized research is to actual real-time physical systems. Since most literature in this field utilizes simplified systems, determining how closely a real-time physical system tracks those results will help guide future research. Second, it is of interest to determine how to implement network-decentralized algorithms on real-time physical systems. This is done through the lens of the Delft Center for Systems and Control (DCSC) Distributed Robotics Lab, and focuses on the development of a lab environment that allows for rapid, user-friendly prototyping.

This thesis contains the following contributions:

- **Novel Network-Decentralized Algorithm:** The development of a new formation producing algorithm for multi-agent systems that integrates network-decentralized control and estimation, along with a decentralized collision avoidance protocol, both in 2-D and 3-D environments. The 3-D case ensures that agents are protected from potentially hazardous propeller downwash effects.
- **Holonomic Simulations:** Extensive simulations to validate the new algorithm in an ideal setting and the development of guidelines for how to effectively tune the algorithm parameters.
- **Novel Free-Space Metric:** The development and discussion of a free-space metric which allows for results achieved in environments with different dimensions to be compared.
- **Re-designed, Virtualized Lab Environment:** The complete design, development, and implementation of a new lab environment that allows for rapid testing of swarms

of quadcopters. As well as a reliable life-cycle for system identification to bring new quadcopter platforms into the lab environment.

- **Non-Holonomic Simulations:** Extensive simulations in Gazebo that take into account real-time non-linear non-holonomic dynamics to show the applicability of the algorithm in a non-ideal setting.
- **Physical Vehicle Testing:** Tests that showcase the effectiveness of the developed platform in the lab environment with a single quadcopter. Due to a lack of test vehicles in the department and significant order delays, implementing the swarm case physically in the lab is left as future work.

The thesis is structured as follows:

- **Chapter 2:** Background information is presented concerning the 2-D holonomic vehicle that is considered later in the thesis, as well as core network-decentralized concepts that are also relevant in later chapters.
- **Chapter 3:** Presents simulations of the network-decentralized algorithm in 2-D. An analysis of the network-estimation performance is included as well as a tuning process for the algorithm parameters validated through Monte-Carlo methods. The formation producing properties and collision avoidance guarantees are developed, discussed, and validated through simulation. Additional content summarizes essential tuning parameters which is necessary for the rest of the presented work.
- **Chapter 4:** Extends the previously discussed work into 3-D. Another set of simulation environments are presented along with additional testing and validation of the extension to 3-D using Monte-Carlo methods. A free-space metric is presented which allows tests to be compared even across dimensions, eg: 2-D and 3-D.
- **Chapter 5:** Discusses quadcopter lab designs present in literature and their respective vehicle design choices. Advantages and disadvantages of these labs are discussed and a design for the DCSC lab is developed and implemented. Further analysis is done on the topic of vehicle selection with regard to the categories of: complexity, size, weight, and power. A suitable platform for long term, widespread use by the department is presented.
- **Chapter 6:** Details the vehicle dynamics concerning a quadcopter. A nested controller linearized about the vehicle hover envelope is discussed and relevant system identification parameters are highlighted. Identification of these parameters is performed and controller synthesis is shown through simulation.
- **Chapter 7:** The network-decentralized algorithm presented in this thesis is applied to a non-linear non-holonomic real-time simulation of a swarm of quadcopters. The new lab design is validated through testing a single physical quadcopter and results from the physical tests are compared with the vehicle in simulation and the holonomic agent case. It is shown that there is similar behavior between the holonomic agent and real-time quadcopter implementations.

Network-Decentralized Approaches for Multi-Agent Systems

It is common to see the complexities of collision avoidance, especially with the added difficulties of network-decentralization, tested in simulation using a single integrator system [24, 25, 19, 26, 27]. Newer and less explored is the use of double integrator systems for the testing of network-decentralized algorithms [20, 28, 29, 30, 31, 32, 33, 34, 35]. The last and least explored is the use of actual non-linear vehicle dynamics [36, 37, 38, 39]. This chapter presents work inspired by a previous thesis which performed preliminary tests, in simulation, of a novel 2-D network-decentralized collision avoidance algorithm [40]. In the previous thesis the approach was applied to a homogeneous set of holonomic vehicles.

2-1 The 2-D Holonomic Agent

As mentioned previously, the work presented in this thesis builds upon an original network-decentralized collision avoidance algorithm. However, the vehicle of choice is not unique in this field of study.

Shown in Figure 2-1 is a vehicle represented in two dimensions. Its center is a point, it is of uniformly distributed mass m , and its radius is R . Note that in previous work R is typically taken as homogeneous for the set of all vehicles, and R is known a priori. For the purpose of later work presented in this thesis, the radii of the various agents are assumed to be potentially different. However, it is required that R is the radius of a n-sphere that entirely encompasses the vehicle. The heterogeneity of the agents is described as:

$$V_i \in \mathcal{V} \quad \neg \Box (V_i = V_j) \quad \forall i, j \in \{1 \dots N\} \quad (2-1)$$

Where in (2-1), V_i is a vehicle, $\mathcal{V}$ is the set of all vehicles, and N is the total number of vehicles in the set. $\neg \Box$ is modal logic notation expressing the condition "not necessary". The set $\mathcal{V}$

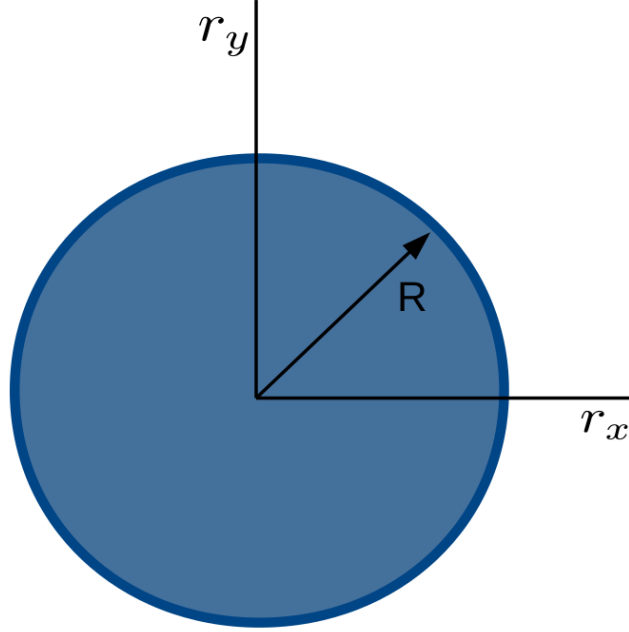


Figure 2-1: A 2-D holonomic agent with radius R .

is a connected graph of agents. This means that given reliable communications, all agents in the set can communicate at least indirectly with every other agent. This will be presented in more detail later.

The holonomic vehicle dynamics can be described by a linear state-space system of equations.

$$\underbrace{\begin{bmatrix} \dot{r}_{x,i} \\ \dot{r}_{y,i} \\ \ddot{r}_{x,i} \\ \ddot{r}_{y,i} \end{bmatrix}}_{\dot{x}_i} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} r_{x,i} \\ r_{y,i} \\ \dot{r}_{x,i} \\ \dot{r}_{y,i} \end{bmatrix}}_{x_i} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m_i(t)} & 0 \\ 0 & \frac{1}{m_i(t)} \end{bmatrix} u_i(t) \quad (2-2)$$

Equation (2-2) shows how the 2-D model evolves over time. The input $u_i(t)$ is an acceleration and the mass can be time-varying. The time-varying mass is not explored in depth in this work, but is presented to demonstrate a feature of the avoidance algorithm. The agents are speed-limited. Speed, in this case, is represented as:

$$S_i(t) \triangleq \sqrt{\dot{r}_{x,i}^2 + \dot{r}_{y,i}^2} \quad (2-3)$$

Note that (2-3) shows the speed of an agent, and this is not the same as its velocity components. In this case, speed is the length of the agent's 2-D velocity vector. To scale the velocity vector of an agent to meet maximum speed requirements a vector projection method is utilized.

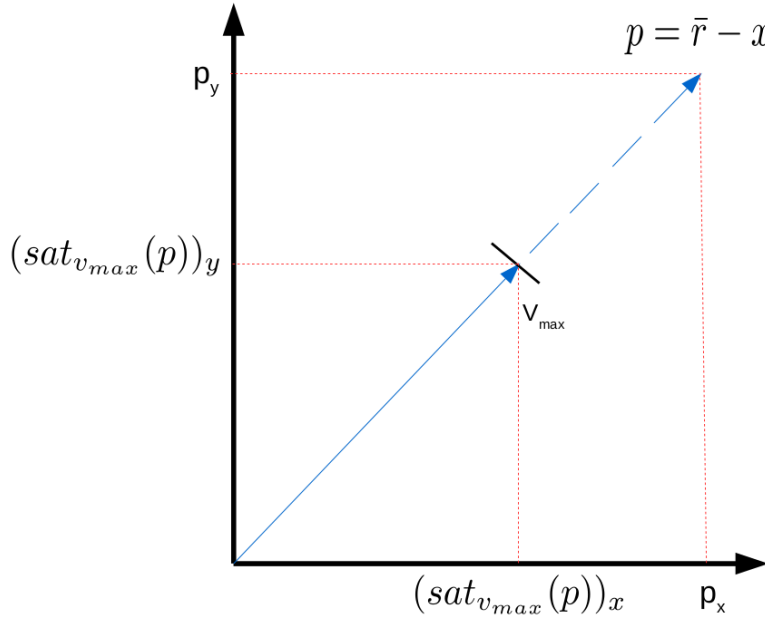


Figure 2-2: Speed limiting via vector projection

Figure 2-2 intuitively shows an adequate scaling method for vehicle speed. It is important that this is applied correctly, as it is one of the essential components for successful avoidance. This can be neatly represented as:

$$v_{i,scaled}(t) = \frac{\dot{\mathbf{r}}_i}{\|\dot{\mathbf{r}}_i\|} v_{i,max} \quad (2-4)$$

In Equation (2-4), $\dot{\mathbf{r}}_i$ is a vector of the x and y velocity components. Note that this can be extended to 3-D dimensions as well and this is done in Chapter 4.

For control of the agents, acceleration components as inputs can be difficult to work with. Additionally, the saturation of the vehicle speed introduces non-linearities to the system. As a result backstepping control is utilized to stabilize the system.

$$\ddot{\mathbf{x}}_i(t) = sat_{s_{lim,i}}[(\bar{\mathbf{r}}_i - \mathbf{x}_i(t))k_1] \quad (2-5)$$

$$\mathbf{u}_i(t) = (\ddot{\mathbf{x}}_i(t) - \dot{\mathbf{x}}_i(t))k_2 m_i(t) \quad (2-6)$$

This is essentially a twice nested proportional controller, where the inner-loop generates a virtual velocity, $\ddot{\mathbf{x}}_i(t)$, that is then tracked by the outer-loop which produces the vehicle acceleration input, $\mathbf{u}_i(t)$. The tuning parameters for this controller, k_1 and k_2 , are not the focus of the algorithm and simply need to be tuned to a relatively low overshoot, $\pm 5\%$. With respect to the time-varying mass, it has been shown that by substituting $\mathbf{u}_i(t)$ into (2-2), that the masses cancel out [40]. Subsequently, the closed-form expression and its stability, evaluated by its eigenvalues, are independent of the mass.

2-2 2-D Network-Decentralized Estimation

Given the specific vehicle discussed in the previous section, a collision avoidance algorithm has been developed. The estimation and information sharing segment of this approach will now be detailed.

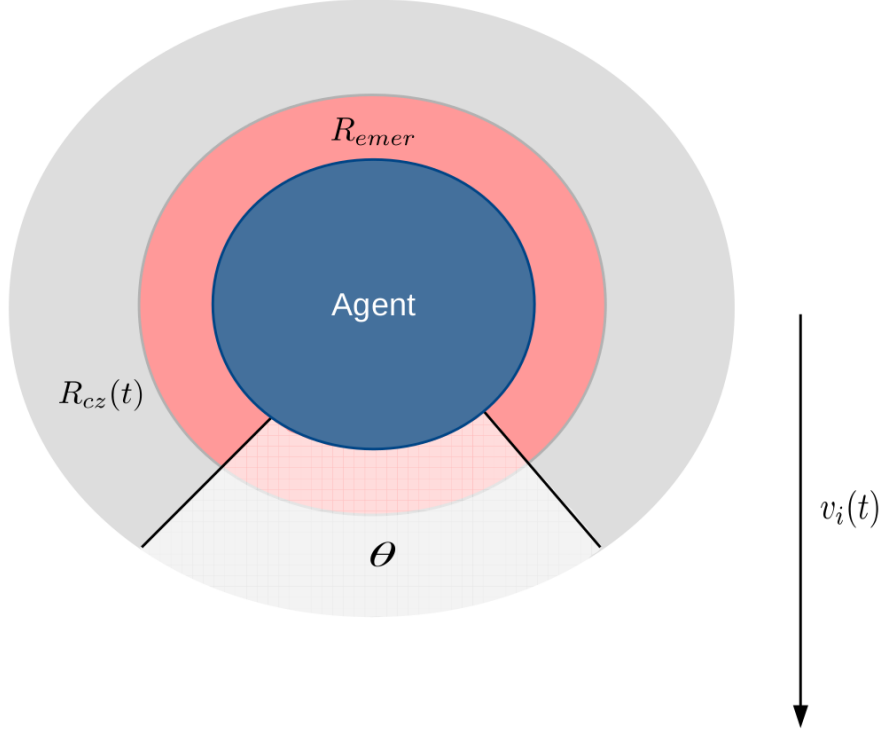


Figure 2-3: A 2-D holonomic agent with two distinct collision detection regions.

Figure 2-3 shows the two collision regions that every agent has. The outermost region, $R_{cz}(t)$, is shown in grey with a specific cutout, θ , which represents the agent's field of view and is the only portion of the outer region in which the agent can detect obstacles. The window always faces the agent's direction of motion and can be thought of intuitively as a camera or Light Detection and Ranging (LIDAR) for sensing obstacles. The radius of this circular region is time-varying, scaling with the agent's velocity. Both the field of view and the scaling rate of the region are tunable parameters. The time-varying nature of this region is agent-specific and defined as:

$$R_{cz}(t) = c + \beta v_i(t) \quad (2-7)$$

Equation (2-7) describes how the outer region scales with the agent's velocity, $v_i(t)$, additionally, c and β are both tuning parameters which are discussed in more detail later. An important consequence of the time-varying outer region is this allows the agent to respond appropriately to obstacles when traveling at higher velocities.

The second region, R_{emer} , is much smaller, 360 degrees in sweep width, and not time-varying. This is the agent's emergency detection region and its radius is also a tuning parameter. The

physical implementation of this region is more difficult to imagine, however it will be shown later that sensing in this region is required for successful implementation of the collision avoidance algorithm and this can be accomplished through network-decentralized estimation.

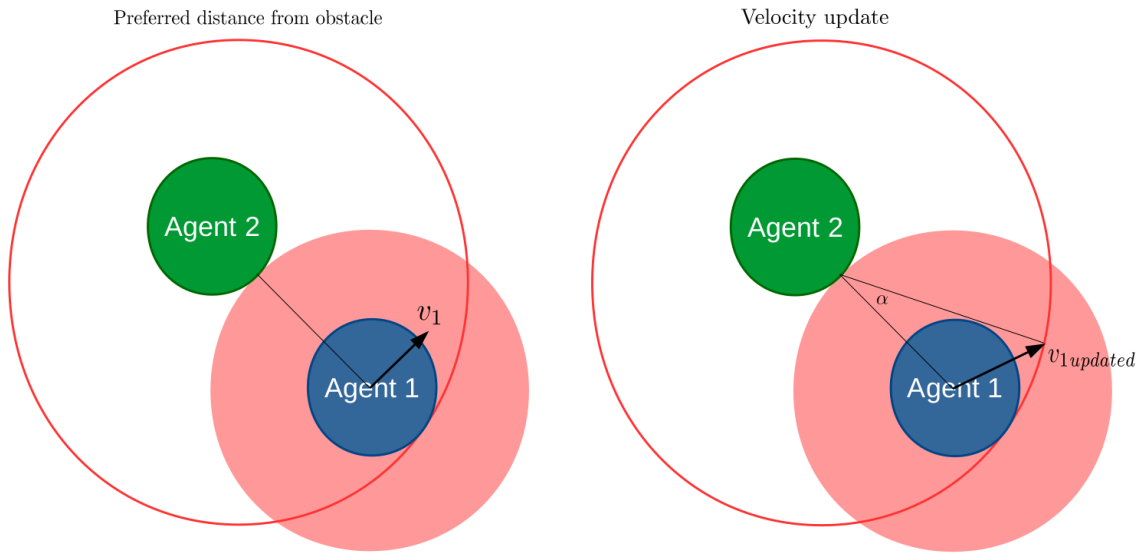


Figure 2-4: The 2-D agent collision avoidance velocity update.

Now the actual avoidance maneuver will be detailed. It is important to note that the agents assume that all other agents will avoid by maneuvering to the right. This is required so agents do not turn into each other under the premise of avoidance. Figure 2-4 shows a brief summary of the avoidance maneuver. This is also outlined below where Agent 2 is referred to as the obstacle and Agent 1 is simply the agent.

1. The agent detects an obstacle in one of its 2 sensed regions (outer or inner)
2. The intersection point between the closest edge of the obstacle and the agent's avoidance region is calculated
3. A vector is drawn from the center of the agent to the intersection point with the obstacle
4. A preferred avoidance region circle of radius R_{pref_dist} is drawn around the intersection point
5. The agent-obstacle vector is rotated counterclockwise by the avoidance angle α
6. The rotated vector is projected onto the preferred avoidance region circle, this results in a single point
7. The vector made from this projected point and the center of the agent becomes the agent's new velocity setpoint

Each agent needs to know the position of obstacles that fall within a detection region. On one hand, this can be accomplished via on-board sensors for the agent. This would require no information sharing. On the other hand, if the agents can share information there is a potential for reduced sensor payloads, agent-to-agent coordination, and reduced avoidance times.

In this thesis it will be assumed that the agents can communicate with nearby agents, defined inside a circle of radius R_{comm} , and the communication times are taken as instantaneous and lossless. The selection and rejection of certain position information inside a given communication radius is examined in this chapter. To model communication between agents it is common to rely on algebraic graph theory [22, 21, 20, 41, 42, 28, 29, 33, 19, 40].

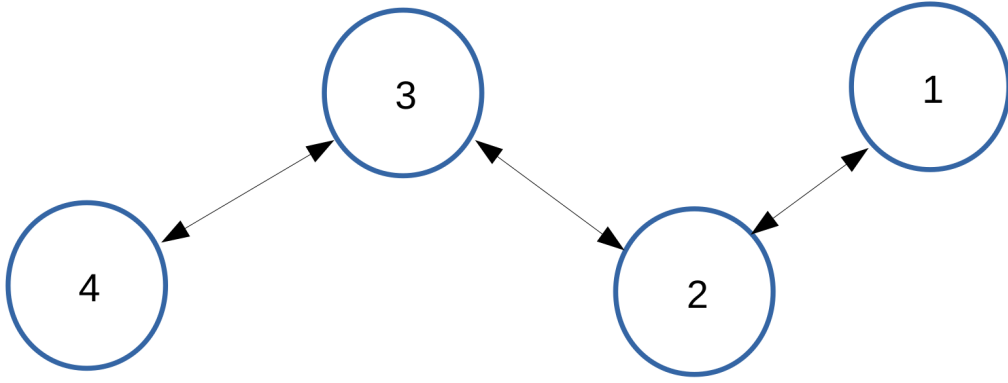


Figure 2-5: A basic undirected communication graph.

The communication graphs are assumed to be undirected, this means that each inter-agent connection is bidirectional. With this in mind, an incidence matrix is one way to describe the connectivity, where the columns are the graph edges (lines), and the rows are the graph nodes representing agents.

$$H(t) = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \quad (2-8)$$

Equation (2-8) shows a matrix representation of Figure 2-5. The undirected nature of the communication is represented by the fact that all rows and columns of the matrix sum to zero. It is natural that this matrix is time-varying in both values and dimensions. H is of size $N \times M_t$ where N is the total number of agents and M_t is the number of graph edges.

The multi-agent system can be written as a single state-space equation which encapsulates the inter-agent dynamics (or lack thereof in this case), and the current communication graph $H(t)$.

$$\begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_N(t) \end{bmatrix} = \underbrace{\begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_N \end{bmatrix}}_A \begin{bmatrix} x_1(t) \\ \vdots \\ x_N(t) \end{bmatrix} + \underbrace{\begin{bmatrix} B_{1s} \\ \vdots \\ B_N \end{bmatrix}}_B u(t) \quad (2-9)$$

$$y = \mathcal{C}x \quad (2-10)$$

In Equation (2-9) A is a block diagonal matrix denoting that the agents are not physically coupled. The pairing representing a single agent, A_1 and B_1 for example, is the same as seen in Equation (2-2). For the output matrix of a non-network-decentralized case, $C = \text{diag}(C_1, \dots, C_N)$ where in an ideal setting each agent would be able to measure its absolute position and velocity making C_i a diagonal matrix. However, what if the agent can only measure its relative position? This results in a different definition of C , denoted $\mathcal{C}$ and described below.

In this thesis, the goal is to implement network-decentralized estimation and sharing of position information. This enables each agent to estimate its absolute position even in the absence of its own sensor data. Therefore, in this approach each agent must know its own velocity for avoidance maneuvers, and be able communicate relative position error with agents inside a certain communication radius.

With inter-agent communication a Luenberger observer can be realized that estimates each agent's absolute position. This is shown for the whole system as:

$$\dot{z}(t) = Az(t) + Bu(t) + L(t)(y(t) - Cz(t)) \quad (2-11)$$

In Equation (2-11) z is the estimated absolute position, $\hat{\hat{r}}_x, \hat{\hat{r}}_y$ of all of the agents. $L(t)$ is chosen such that the error dynamics between the estimated state and actual state, $\dot{e} = \dot{z} - \dot{x}$ converges to zero. Since the goal is to share relative position information in a network-decentralized way both the output matrix, C , and the observer gain matrix, $L(t)$, must take on a specific form. This is efficiently accomplished through the use of the incidence matrix and algebraic graph theory.

$$\underbrace{\begin{bmatrix} d_{12} \\ d_{21} \\ d_{23} \\ d_{32} \\ d_{34} \\ d_{43} \end{bmatrix}}_d = \underbrace{\begin{bmatrix} -C_1 & C_2 & 0 & 0 \\ C_1 & -C_2 & 0 & 0 \\ 0 & -C_2 & C_3 & 0 \\ 0 & C_2 & -C_3 & 0 \\ 0 & 0 & -C_3 & C_4 \\ 0 & 0 & C_3 & -C_4 \end{bmatrix}}_C \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad (2-12)$$

$$L(t) = \begin{bmatrix} L_{11} & -L_{11} & 0 & 0 & 0 & 0 \\ -L_{22} & L_{22} & L_{23} & -L_{23} & 0 & 0 \\ 0 & 0 & -L_{34} & L_{34} & L_{35} & -L_{35} \\ 0 & 0 & 0 & 0 & -L_{46} & L_{46} \end{bmatrix} \quad (2-13)$$

Equation (2-12) describes the distance between agents in a network-decentralized manner. This is shown because the output matrix has the same block matrix form as the incidence matrix. The network-decentralized limitations are also expressed in (2-13) where the observer gain matrix, again, has the same block form as the incidence matrix.

If it is assumed that in (2-12) $C_1 = C_2 = C_3 = C_4$, where C_1 is the portion of the output matrix of each agent that encapsulates relative position information, then a more compact representation can be presented. Note, the agents are not sharing velocity information. Algebraic graph theory allows for $\mathcal{C}$ to be written for the time varying case as:

$$\mathcal{C} = H(t) \otimes -C_1 \quad (2-14)$$

where a Kronecker product is used; intuitively, Equation (2-14) maps the matrix $-C_1$ onto each location in $H(t)$ changing the sign appropriately. Since $H(t)$ is time-varying and describes the inter-agent communications the matrix $\mathcal{C}$ is also time-varying.

It also must be noted that all agents need to be part of a connected set of agents, a connected graph, where at least one agent can measure its own absolute position reliably. This requirement is notated as:

$$r_{x,i}, r_{y,i} = \bar{r}_{x,i}, \bar{r}_{y,i} \quad i \in \{1 \dots N | i \geq 1\} \quad (2-15)$$

where in (2-15) r_i is the i -th agent's measured position and $\bar{r}_i$ is the actual position of the agent. Note that the graph architecture itself does not need to be fixed, only the connectedness property must hold.

2-3 2-D Network-Decentralized Collision Avoidance

The information sharing in a network-decentralized way was defined above. Now multi-agent control utilizing this information will be detailed. The specific control problem that is solved here is the "formation producing" problem [19]. The actual obstacle avoidance is handled on an individual agent basis, and is similar to a disturbance preventing the agent from achieving its formation goal. For formation control there are two general research areas: formation producing and formation tracking. Formation tracking is the case where multiple agents maintain a formation while the formation itself tracks a reference signal. This is in contrast to the approach here. In the formation producing case the agents maintain a formation while each individual agent tracks its own trajectory, a subtle but important difference [19].

There are two important factors that are considered in the formation producing control used here. The first is the fact that each agent tracks its own reference, and the second is the absolute error between each agent and its reference position. Refer to Equations (2-5) and (2-6) to see the backstepping control used by each agent. With network-decentralized estimation properly implemented, each agent will know the absolute position error between its current position and reference position. This is defined as: $e_{r,1} = \bar{r} - r(t)$. If each agent also shares its absolute position error with the agents inside its communication radius then a new reference can be tracked by each agent, shown as:

$$\begin{aligned}\bar{r}_{new,1} &= e_{r,1} - e_{r,2} \\ \bar{r}_{new,2} &= e_{r,2} - e_{r,1}\end{aligned}\tag{2-16}$$

If the error between two agents in (2-16) is equivalent then $\bar{r}_{new,i}$ will go to zero and the agents will have achieved a formation. This can be written for the time-varying network-decentralized case in a similar manner as seen in Equations (2-12) and (2-13).

$$\bar{r}_{new} = \underbrace{\begin{bmatrix} I_2 & -I_2 & 0 & 0 \\ -I_2 & 2I_2 & -I_2 & 0 \\ 0 & -I_2 & 2I_2 & -I_2 \\ 0 & 0 & -I_2 & I_2 \end{bmatrix}}_{\frac{1}{2}G^T(t)G(t)} \underbrace{\begin{bmatrix} \bar{r}_1 - r_1(t) \\ \bar{r}_2 - r_2(t) \\ \bar{r}_3 - r_3(t) \\ \bar{r}_4 - r_4(t) \end{bmatrix}}_{\bar{r}-r(t)}\tag{2-17}$$

In Equation (2-17) the matrix $G(t)$ is constructed similarly to Equations (2-12) and (2-11) and can be written as:

$$G(t) = H(t)^T \otimes I_2\tag{2-18}$$

Equation (2-18) maps the 2×2 identity matrix I_2 onto every location in $H(t)^T$ where the dimension of I is directly related to the number of dimensions of the space where the agent moves (in this case, 2-D). It is important to recognize that the result of (2-17) only drives the agents to create a formation determined by their collective reference positions. To actually drive the formation of agents to their respective reference positions a weighted offset must be applied to the indexed agent.

$$\frac{1}{2}G^T(t)G(t) + \rho I_{2N} = \begin{bmatrix} I_2 + \rho & -I_2 & 0 & 0 \\ -I_2 & 2I_2 + \rho & -I_2 & 0 \\ 0 & -I_2 & 2I_2 + \rho & -I_2 \\ 0 & 0 & -I_2 & I_2 + \rho \end{bmatrix}\tag{2-19}$$

The weight, ρ , on the diagonal in (2-19) determines the selfishness of each agent. A low ρ value leads the agent to prioritize the formation, whereas a higher value places emphasis on the respective agent's individual reference position.

$$\bar{\dot{r}} = sat_{v_{lim}}[k_1(\frac{1}{2}G^T(t)G(t) + \rho I_{2N})(\bar{r} - r(t))]\tag{2-20}$$

Equation (2-20) updates the inner loop of the individual agent backstepping control method detailed in (2-5). The outer loop, shown in (2-6), remains the same producing an acceleration input for each vehicle's dynamics.

2-4 Analysis and Identification of Key Algorithm Parameters

The previous sections have laid the foundation for the work presented in this thesis, describing a novel collision avoidance algorithm that is implemented in a network-decentralized way. In this section an analysis of the algorithm is presented, as well as limitations that were discovered, and a brief discussion about extensions this thesis provides is included. The intention here is to clearly frame the purpose and usefulness of the network-decentralized work presented in the rest of this thesis.

The previous discussion considered a 2-D holonomic agent which resulted in state-space dynamics of four dimensions, two for position and two for velocity. The input into the state-space dynamics is a 2×1 dimension of acceleration, one for each component: x and y. This input is determined via backstepping control which, in this case, is a nested proportional controller that maps vehicle position error to an acceleration input. Even looking only at the individual agent case, there are many tuning parameters. These are outlined in Table 2-1.

Parameter	Description
α	Rotation angle of the agent-obstacle avoidance vector
R_{cz}	Circle radius of outer velocity-varying collision detection region
c	Constant defining R_{cz} , determines minimum avoidance region
β	Constant defining R_{cz} , determines scaling rate of avoidance region
θ	Window size of outer collision detection zone
R_{pref_dist}	Avoidance circle radius defining preferred distance from obstacle intersection
R_{emer}	Circle radius of inner fixed emergency detection zone

Table 2-1: Individual agent tuning parameters

It is clear that there is a substantial amount of complexity that must be discussed when choosing parameters. In Chapter 3 a basic minimal set of parameters will be compared to the tuning method presented in this thesis. This approach to tuning is presented to address collisions found using uninformed parameter selection. Additionally, a modification from one rotation angle for both avoidance regions to the use of two independent rotation angles is discussed.

Along with the individual agent tuning parameters found in Table 2-1 there are tuning parameters for the network-decentralized estimation and control aspect. These are summarized in Table 2-2.

Parameter	Description
R_{comm}	Homogeneous agent communication circle radius. Forces bidirectionality
$L(t)$	Network decentralized observer gain matrix, has time-varying structure
ρ	Scalar that weighs agents' tendency to meet formation goal vs. individual reference
k_1	Scalar proportional gain that controls the virtual reference velocity

Table 2-2: Network-decentralized tuning parameters

Similar to the parameters in Table 2-1, the tuning of the parameters in Table 2-2 is discussed in Chapter 3. The method in which communication is performed between the agents is important and there are a few different ways this can happen. A fully dynamic R_{comm} would

allow the radius to actively scale up online until each agent communicates with a pre-specified number of other agents. On the other hand a fully static radius can be used, as well as a hybrid between the two which uses a fixed radius on R_{comm} , but only allows communication with a maximum number of vehicles n_{conn} . This agent communication number would become another tuning parameter. The initial thought is to include information from all available agents using a fixed communication radius, because each agent will have to process the information from agents within a certain radius anyway, and this occupies bandwidth, processing power and time. Furthermore, the most common method to determine communication distance is through a Received Signal Strength Indicator (RSSI) value. This is not a highly reliable metric, especially indoors, and is subject to tuning parameters of models, most commonly: Fries' transmission formula, and the log-distance path loss model [43]. However, the use of relative distance information provided by the network-decentralized estimation portion of this thesis yields another approach independent of RSSI measurements. The fixed and hybrid approaches are explored and the difference between the approaches yields additional insight into the apparent "speed-up" effect network-decentralized control provides in collision avoidance situations. This is discussed in detail in Chapter 3.

The parameter $L(t)$ shown in Table 2-2 is a matrix of observer gains. The selection of these gains directly affects the convergence rate of each agent's position estimation and subsequently their ability to detect obstacles inside their emergency detection region. The setting of these gains online and in a network-decentralized way that enables successful collision avoidance is novel. For the algorithm to be truly network-decentralized it is important to only use information provided by the network. It also must be assumed that each agent's estimated position will converge quickly enough to provide accurate position information for the purpose of avoiding collisions. In Chapter 3 of this thesis the selection of observer gains under completely network-decentralized conditions is discussed.

For collision avoidance from a practical perspective, the all-around emergency detection region for each agent is reliant on an accurate absolute position estimation of every nearby agent. Recent technology, such as the RPLidar A3, can provide an onboard 360° 2-D scanned region using LIDAR laser sensing [44]. A unit like this can provide a position solution at up to 20 Hz, with 10 m range all-around accuracy, and an angular resolution of 0.225°. However, this comes at a cost. The unit consumes up to 2.5 Ah, weighs 190 g, and requires additional on-board computing power to process 256 Kbps of information in a meaningful way [45]. While this may be acceptable for a ground vehicle operating in 2-D, this thesis extends the network-decentralized avoidance algorithm to 3-D. The network-decentralized position estimation approach provides more information in a higher dimension than a LIDAR system can. All without the need for additional agent payload. A detailed explanation of the system engineering choices for the quadcopter platform used in this thesis is presented in Chapter 5 and the selection of observer gains is discussed in Chapters 3 and 4.

The parameters ρ and k_1 shown in Table 2-2 are scalars and they are applied to all agents identically. It makes intuitive sense that ρ is applied in a homogeneous manner because the network-decentralized representation of the agents, $\frac{1}{2}G^T(t)G(t) + \rho I_{2N}$, relies on symmetry for moving as a formation. The control gain k_1 is also a scalar, but the implications of altering it in an agent-dependent manner are less obvious. It seems that the gain may be important in instances where the agents are heterogeneous, but altering k_1 has secondary effects on the formation behavior of the agents. Another way to compensate for the heterogeneous agent case is to alter the individual agent's avoidance regions. While this latter idea is explored in

Chapters 3 and 4, this thesis does not explore the implications of applying a matrix of either ρ or k_1 gains, which is left as future work.

Chapter 3

2-D Holonomic Simulations

The network-decentralized network-estimated collision avoidance and formation producing algorithm used in this thesis was introduced in Chapter 2. An analysis of this approach is now presented where in this chapter a set of simulation environments are introduced and used to stress-test the approach. Through this various improvements to the algorithm are proposed, as well as a look at important interconnections when setting the algorithm's parameters.

3-1 Baseline Parameter Evaluation via Synchronous Simulation

In Chapter 2 an approach to collision avoidance and formation production was discussed. In this thesis a set of simulation environments is designed in MATLAB with the focus that they are modular, scalable, and readable. A specific simulation environment is designed for the three separate test cases: the individual-agent case, the full-information network-decentralized control case, and the network-decentralized estimation network-decentralized control case. Respectively, each environment increases in complexity with regard to how each agent determines the information it needs. A detailed guide to the simulations can be found in [46] and the code itself is included in the physical lab environment detailed later. Note that the MATLAB simulation environments built for this thesis, detailed in Chapters 3 and 4, are evaluated synchronously. In short, the sensor data, vehicle dynamics, and control calculations are all updated each time step. A simplified diagram of the system is shown in Figure 3-1:

Figure 3-1 shows some possible delays in the physical implementation of a 2-D holonomic agent. In general, all of the time delays are lumped together for the whole simulation under the discrete-time step-size. This is $t_d = 0.01$ seconds, much faster than the dynamics of the actual physical quadcopter system shown in Chapter 6. Additionally, each of the time delays in Figure 3-1 also represent a data flow, and not all are relevant for each simulation. For example, in the fully network-decentralized case all associated data is utilized, however in the individual agent case no inter-agent communication, t_{comm} , exists. Also note that $\bar{y} = \hat{y}$, which indicates that there is no measurement noise in the update.

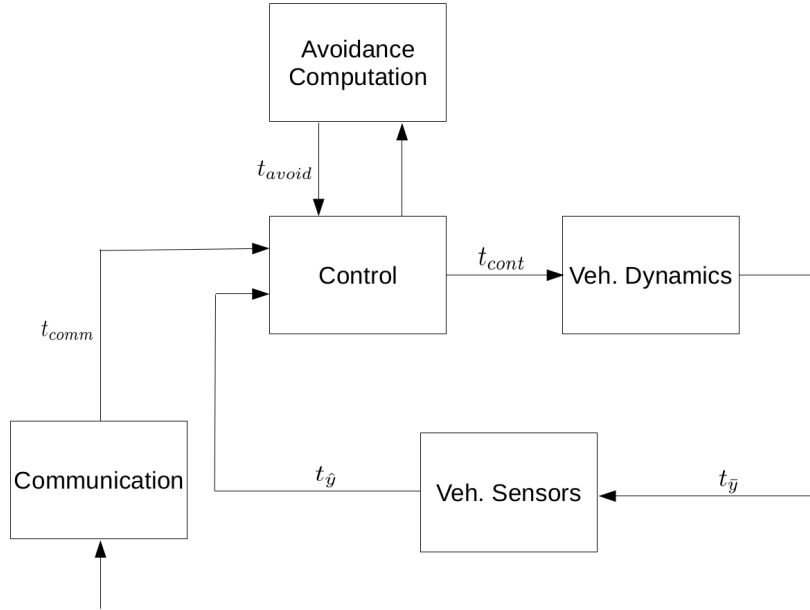


Figure 3-1: A simplified diagram of the system, with the indication of some delays that are present in a physical implementation of the simulation.

Using the simulation environment created for this thesis and a specific stress-test detailed below, a preliminary set of parameters modeled after the results in [40] can be run in a Monte-Carlo simulation, perturbing the agents' initial positions, to obtain an understanding of their robustness. In this test there are a few assumptions that must be noted. The communication between agents is performed dynamically in an instantaneous and lossless way. There is a communication region of theoretically infinite radius that is time-varying in radius to ensure that a minimum number of agents are communicating. For now, at any given moment three agents are in communication. This is presented to perform a baseline analysis of previous work, and an alternate approach will be discussed in Section 3-2. The observer gains are set in a network-decentralized manner with only one agent being externally connected. The tuning of these parameters is discussed in the next section. In this test there is no livelock detection or breaking. This is done to show how livelock can cause issues and that this can be overcome without implementing specific livelock breaking methods. It is also assumed that the agents will converge within thirty seconds or less, otherwise the test logs a failure for convergence. Lastly, the agents are unable to scan their entire outer region once a collision is detected. Instead, only the inner emergency detection region has 360° visibility. Given the assumptions above, the parameters are evaluated in a manner displayed in Figure 3-2 and Tables 3-1 and 3-2:

Agent #	1	2	3	4	5	6	7	8	9	10
Init. points	(6,4)	(4,4)	(3,4)	(0,4)	(0,2)	(0,0)	(2,0)	(4,0)	(6,0)	(6,2)
Setpoints	(0,0)	(6,0)	(4,0)	(6,0)	(6,2)	(6,4)	(4,4)	(2,4)	(0,4)	(0,2)

Table 3-1: Agent initial points and setpoints

Figure 3-2 shows a successful iteration of the test that was run in the simulation environment

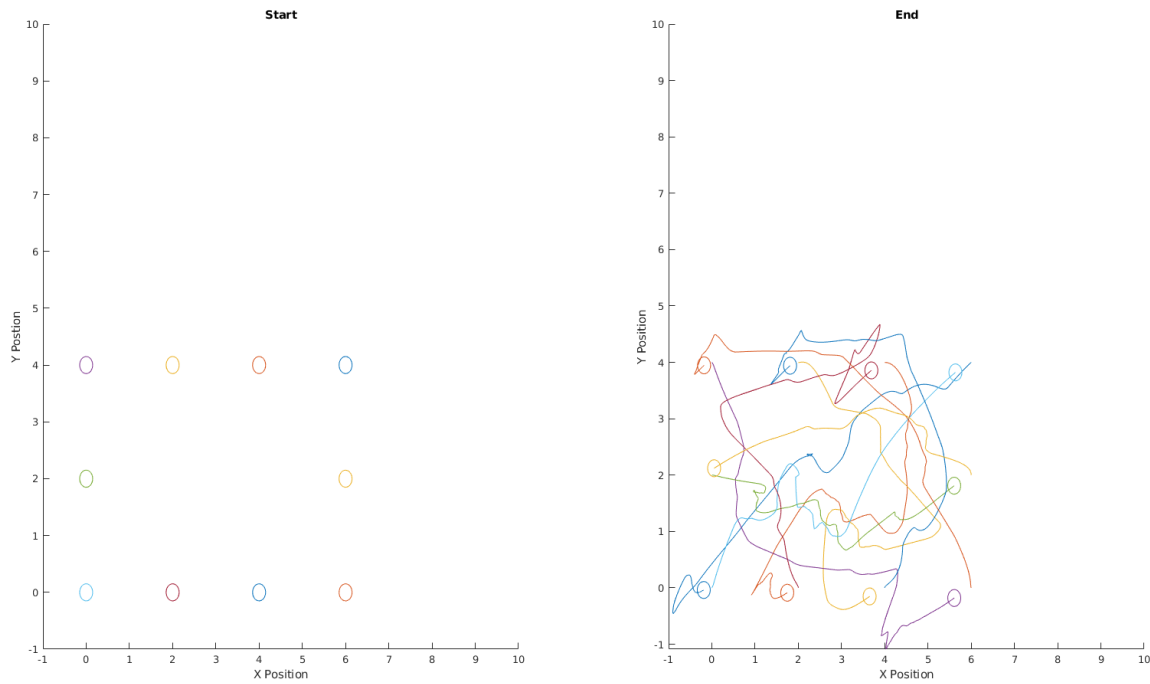


Figure 3-2: A 2-D network-estimated network-decentralized test.

Parameter	Value
m	1
k_1	20
k_2	12
α	5
c	0.36
β	0.48
v_{max}	1.0
θ	120°
R_{veh}	0.15
R_{pref_dist}	$R_{cz} - 0.18$
R_{emer}	0.36

Table 3-2: Baseline test parameters

	Trials	Percentage (%)	Ave. Time (s)
Collisions	386/500	77.2	N/A
Convergence	0/500	0	N/A

Table 3-3: Breakdown of baseline test results without livelock breaking

created for this thesis. This is not indicative of the results shown in Table 3-3. This figure only serves to help visualize the desired result. The agents are initialized in a rectangle, their initial positions are perturbed by some uniform random value, ± 0.30 rounded to the second decimal place, and their reference positions are the unperturbed initial position vector circularly

shifted. The initial positions and setpoints can be seen in Table 3-1. Table 3-3 shows the results of 500 simulations using this methodology and the parameters from Table 3-2. Note that the parameters shown in Table 3-2 are taken from [40] where the high percentage of collisions and non-existent convergence rate were compensated for via add-on solutions eg: live-lock breaking, deadlock avoidance, agents scanning their full outer region for obstacles, fully informed setting of observer gains. Removing these extra factors yields the somewhat surprising results seen in Table 3-3. In the next section it will be shown that these additions are unnecessary and that successful results can be achieved purely through appropriately tuned parameters.

Observing the collision avoidance case, it can be noted that the result of an avoidance calculation is a new reference velocity vector for the agent. In close quarter situations between agents, a small avoidance angle α will produce a much smaller velocity vector. As can be seen in the test above, this produces many livelock situations where agents cannot maneuver around one another. Additionally, the velocity of each agent is reduced in avoidance situations. A quick test of this impact can be performed by splitting the outer and inner regions into two separate avoidance angles. Leaving the outer region an avoidance angle of 5° and setting the inner region to an avoidance angle of 45° the Monte-Carlo test is run again.

	Trials	Percentage (%)	Ave. Time (s)
Collisions	328/500	65.6	N/A
Convergence	484/500	96.8	18.47

Table 3-4: Basic dual region test results without livelock breaking

Given the results in Table 3-4 it can be seen that a simple alteration of the avoidance behavior results in vastly improved results without the need for livelock breaking. The convergence rate goes from 0% to 96.8%, and the number of collisions falls by 11.6%. This result shows the importance of the tuning parameters in the algorithm, especially the avoidance detection zones. The average convergence time is calculated using only successfully converging tests, and it is noted that the required time, 18.47 seconds, is a baseline for which further improvements can be compared against.

While the results of these basic tests show promise, it is clear that improvements can be made to the parameter tuning. Additional analysis is required for this to be realized.

3-2 Improving Collision Avoidance

Given the results shown in the previous section, there are some significant features that must be detailed. First, the circular regions around each holonomic agent are represented in continuous time in a way that scales neatly to three dimensions, of which the scaling is detailed in Chapter 4. Another feature is in the avoidance velocity calculation. In this thesis two separate angles are chosen, one for the outer avoidance region, and one for the inner region, and the selection of the angle is done purposefully. The selection of the inner avoidance angle is done to maximize the distance between the agent and an obstacle. The third feature is the limited field-of-view for the outer detection region. This moves closer to a practical implementation and simplifies the transition from simulation to a physical model with on-board image based detection of obstacles. Lastly, this thesis restricts the information available

to each agent, using only what is available locally and from nearby agents in accordance with the algorithm shown in Chapter 3. The effects of having network-decentralized estimation in the critical path for successful collision avoidance is detailed further in Chapter 4.

3-2-1 Continuously Defined Detection Regions

As mentioned above, in this thesis the avoidance regions are defined continuously. This is accomplished in 2-D through the use of vector projections.

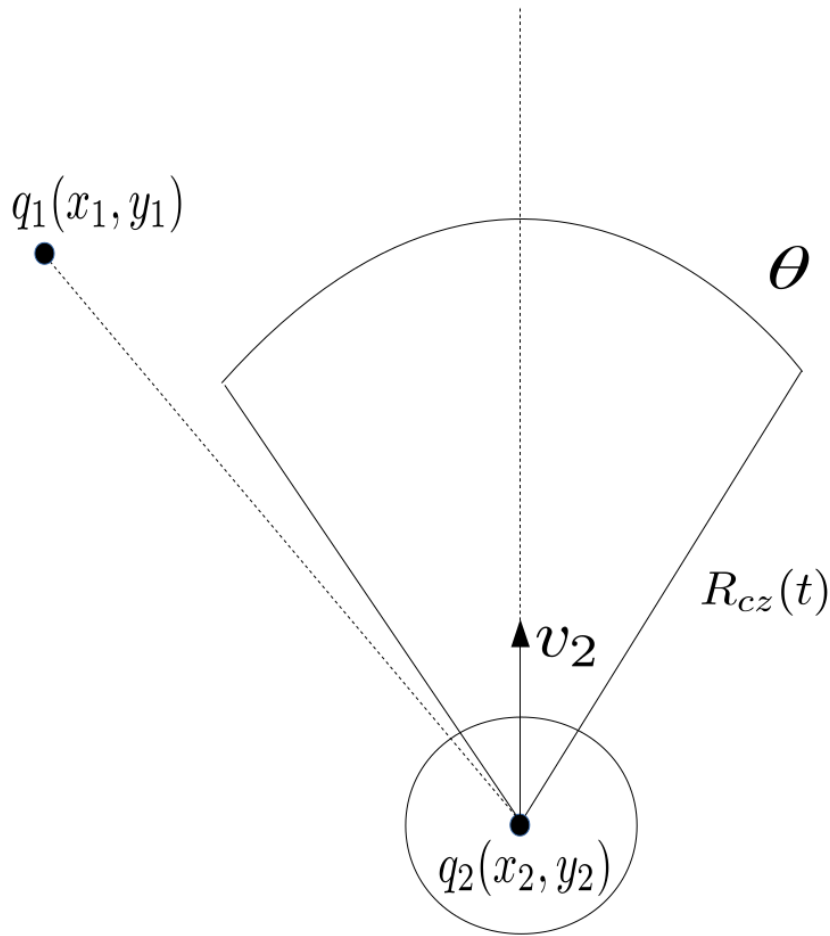


Figure 3-3: Continuous projection definition of the collision avoidance window.

Figure 3-3 shows a typical situation where the agent is evaluating potential obstacles that need to be avoided. $q_2(x_2, y_2)$ is the position of the center of the agent and $q_1(x_1, y_1)$ is the leading edge of a potential obstacle. v_2 is the velocity vector of the agent, θ is the agent's visual window, and R_{cz} is the radius of the outer avoidance circle, as mentioned in Table 2-1. To see if q_1 falls within the visual window the following calculation is performed:

$$\cos^{-1} \left(\frac{(q_1 - q_2) \cdot v_2}{\|v_2\| \cdot \|q_1 - q_2\|} \right) < \frac{\theta}{2} \quad (3-1)$$

If the condition in Equation (3-1) holds true, and the obstacle is close enough, then it is handled by the agent and the agent's velocity is updated for avoidance. The sensing of q_1 by the agent can be done in one of two ways: either by assuming the agent has on-board image processing or by the network-decentralized estimation of the obstacle's position. The image processing approach makes more sense here because, if the obstacle is non-communicating, then there will be no estimation information of its position. Additionally, the inner avoidance region of the agent uses the estimated position of an obstacle, so this information is used. If the outer detection region is scanned via image recognition then there is no way for a non-communicating stationary obstacle to collide with the agent without first being identified. This of course leaves one case that can lead to collisions, the non-communicating mobile obstacle. In 2-D it is more reasonable to assume that 360° sensing can be used to detect obstacles. For example, an on-board Light Detection and Ranging (LIDAR) system could be utilized. However, this is only a solution in 2-D and does not scale well for the 3-D case. Therefore, in this thesis the non-communicating mobile obstacle case is not covered.

3-2-2 Selecting an Appropriate Avoidance Angle

Another change presented in this thesis is the analysis and selection of an appropriate avoidance angle. Now an analysis of the avoidance angle parameter α , as seen in Table 2-1, will be performed with attention given to related parameters also found in the aforementioned table. The following Figures, 3-4 and 3-5, illustrate a simple test of the algorithm.

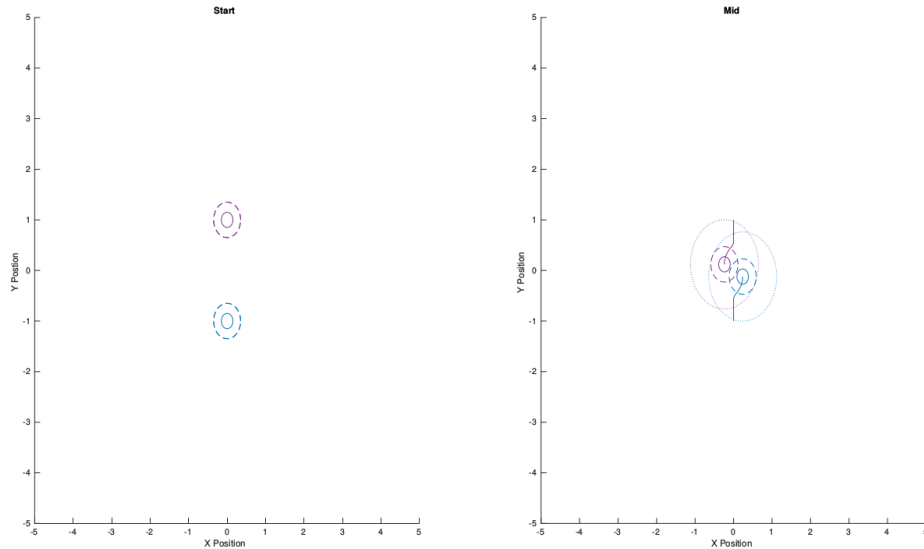


Figure 3-4: Full information collision avoidance in 2-D with holonomic agents, pt. 1.

Figures 3-4 and 3-5 show a basic full-information test with no communication sharing. This simulation environment case is selected because it isolates the effects of the avoidance angles as much as possible. In this test, the agents swap places without colliding. The two collision

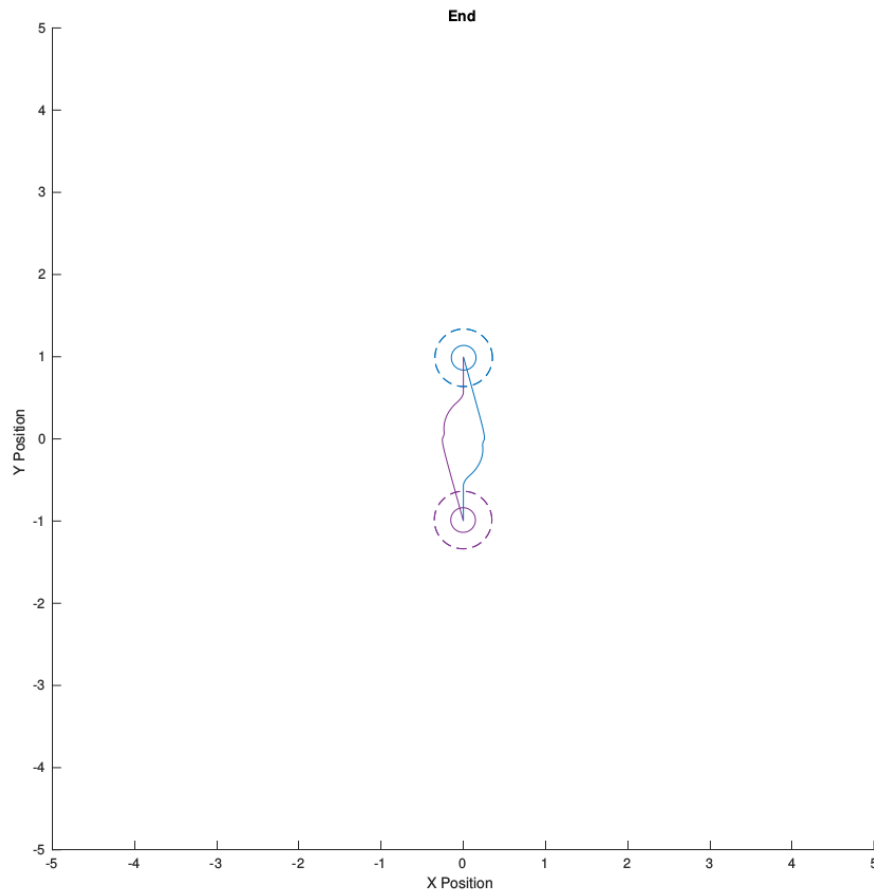


Figure 3-5: Full information collision avoidance in 2-D with holonomic agents, pt. 2.

Parameter	Value
α	(variable, based on tangent line)
c	0.35
β	0.65
v_{max}	1.0
θ	120°
R_{veh}	0.15
R_{pref_dist}	$R_{outer} - 0.05$
R_{emer}	0.35

Table 3-5: Agent test parameters

regions can be seen, where the emergency region is made of dashed lines, and the outer time-varying region is made of dotted lines. The collision avoidance algorithm runs on each agent as mentioned in Chapter 2. The parameters used in the test are shown in Table 3-5.

One intuitive way to select avoidance velocities is to set α dynamically to the tangent point between the avoiding agent and the R_{pref_dist} region. This provides an easily repeatable, consistent process that maximizes the distance between the two agents while staying consistent

with the algorithm concept presented in Chapter 2.

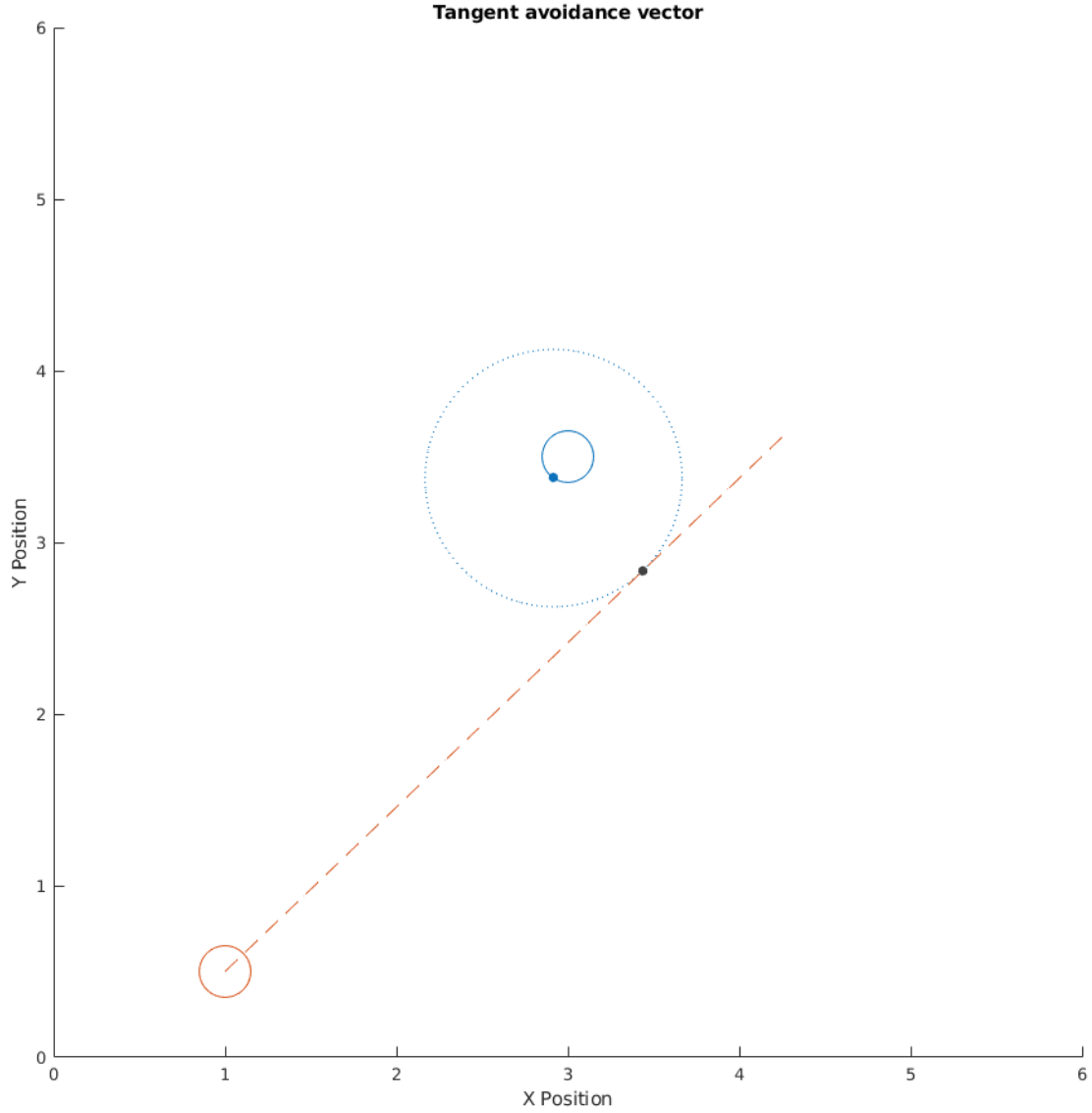


Figure 3-6: Example of an avoidance vector tangent to R_{pref_dist} .

Figure 3-6 shows two agents with reciprocal velocity vectors. In this snapshot the red agent is avoiding the blue one, and the dotted blue ring is R_{pref_dist} drawn around the collision intersection point that the red agent detected. Observing this graphic, it is shown that the tangent point to the right of the obstacle provides the maximum avoidance distance consistent with the algorithm. Realizing a maximum distance between agents is important, but not always desirable. If the agent and obstacle meet in close quarters, or have little time to react, then it is important to take the largest avoidance action possible consistent with the algorithm. However, at larger distances a much smaller avoidance angle can be used to keep agent heading deviations small. The parameters in Table 3-5 are interconnected, with some of the relationships visualized below.

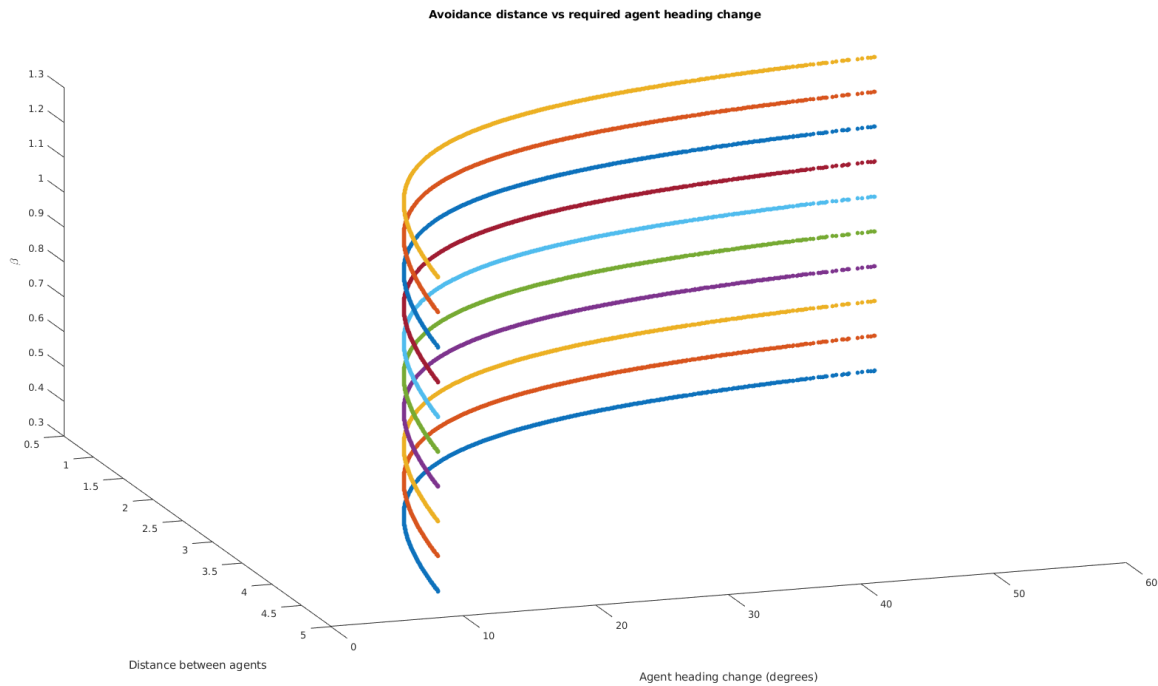


Figure 3-7: Agent required heading change for avoidance at different distances.

Figure 3-7 shows that as the distance between agents or agent-obstacle increases, the required heading change of the avoiding agent decreases. According to the plot, the required heading change of the agent reaches a minimum at approximately 10° at a distance of 5 meters (the units for distance are arbitrary). This of course requires the agent to detect the obstacle, which has strong dependence on Equation (2-7). This relationship can be visualized in Figure 3-8 as:

Figure 3-8 displays the relationship described by Equation (2-7). The vertical axis is the scaling parameter β which is linearly related to the radius of the outer detection zone by the agent velocity. The "required agent velocity" axis describes the minimum vehicle velocity needed to detect an obstacle at a given distance. Essentially, larger values for β allow earlier detection of obstacles at lower velocities. However, this comes at the price. There is the potential for an agent to avoid moving obstacles that are not actually a threat. Unlike the well studied velocity obstacles approach, and related variations, the approach in this thesis does not have access to obstacle velocity information [47, 48, 49]. The consequence of this is that the maneuvering agent cannot reliably project the obstacle ahead in time and therefore may incorrectly identify obstacles as hazardous, when in fact they do not present any actual danger. That being said, the approach in this thesis presents successful collision avoidance while also showing formation producing capabilities in a network-decentralized, position-estimated environment. All of these features cannot be found for current velocity obstacle work, however the network-decentralized estimation of velocity information for collision avoidance may be an interesting extension of the work presented here. On the other hand, a larger β value can lead to earlier obstacle detection with smaller required heading deviations. This would allow an agent to maintain a larger average velocity. Of course the example analyzed in Figures 3-4 and 3-5 is a simple one. Once more vehicles are included the collective consequences of

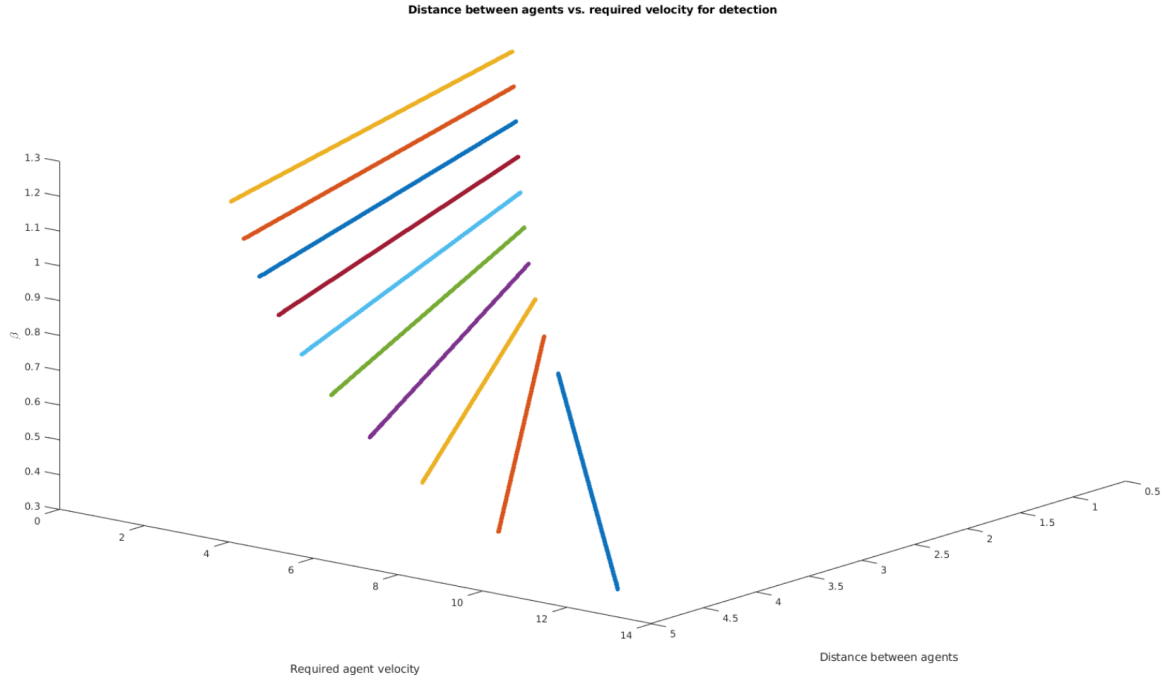


Figure 3-8: How changes in β impact agent responsiveness.

necessary avoidance actions becomes much more complex.

3-2-3 Tuning Network-Decentralized Estimation Parameters

Before additional tests are presented it is important to describe the estimation procedure used in this thesis. The process follows Equation (2-11), but it is the selection of the gains in the time-varying matrix $L(t)$ that is important. $L(t)$ changes both its dimensions and which indices of the matrix are expressed based on the time-varying connectivity of the graph. The selection of these gains in an optimal way is a current area of research not explored here. However, it was found in this thesis that forming the matrix as $L(t) = H(t) \otimes C_1$ causes the network-decentralized position estimation to converge, where C_1 is the output matrix containing only position information. Once this was discovered, the gains of $L(t)$ were scaled up by a factor of 2, which provided sufficiently strong performance to test the algorithm. More advanced techniques are possible, but this is beyond the scope of the work done here.

Using the maximum avoidance angle approach described above, as well as the concept of expanding the detection regions, the test in the previous section is performed again.

	Trials	Percentage (%)	Ave. Time (s)
Collisions	0/500	0	N/A
Convergence	500/500	0	13.52

Table 3-6: Expanded detection zones, maximum avoidance maneuver box test

The results shown in Table 3-6 validate the ideas presented in this section. It is clear that the size and scaling of the avoidance regions plays a crucial role in successful avoidance. The

Parameter	Value
α	(variable, based on tangent line)
c	0.45
β	1.3
v_{max}	1.0
θ	120°
R_{veh}	0.15
R_{pref_dist}	$R_{outer} - 0.05$
R_{emer}	0.35

Table 3-7: Updated agent test parameters

respective parameters used in the test can be seen in Table 3-7. Some additional insight can also be gained by viewing the observer performance of a couple of the agents.

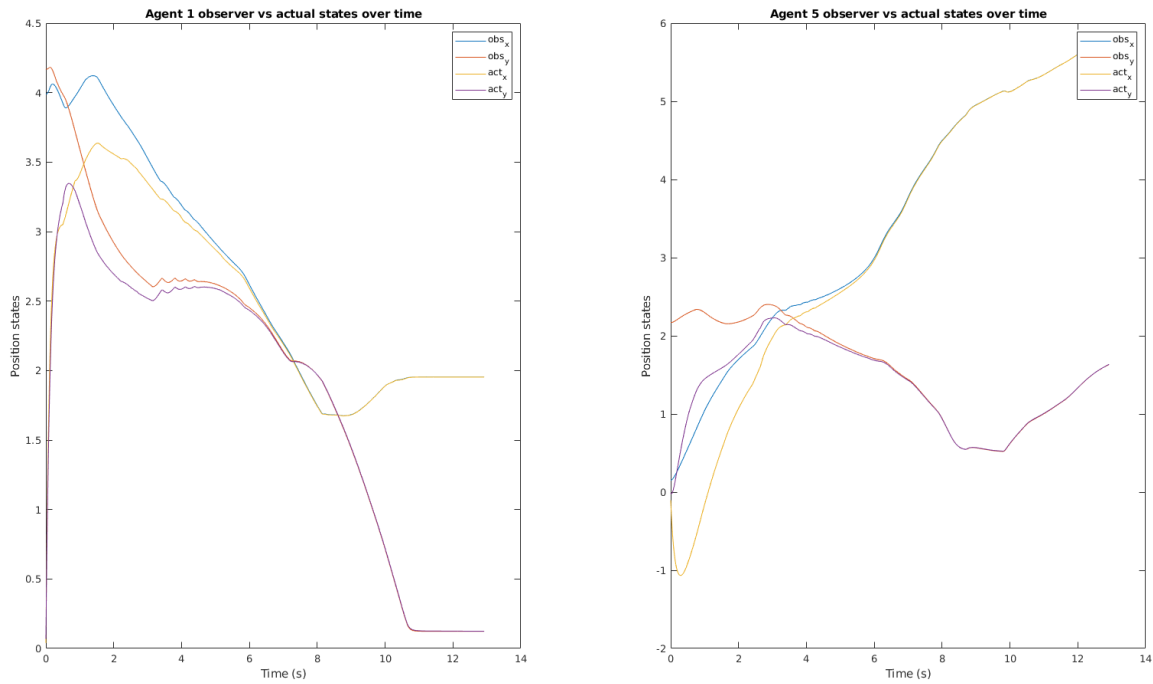
**Figure 3-9:** Observer performance of two agents.

Figure 3-9 shows two agents' observer positions plotted against the actual positions of the agents. It is clear that the convergence can take some time, as much as 7 seconds in this case. However, the "good enough" estimation occurs around the 2-3 second mark and the performance is reflected in the results shown in Table 3-6. The error and error rate performance of the observer for all agents is plotted in Figure 3-10

For all agents, shown in Figure 3-10, the full convergence can be seen at approximately the 6-7 second mark. The sharp cutoff in the figure on the left signifies the point at which all agents achieved their goal positions. This is consistent with the convergence time of 13.52 seconds seen in Table 3-6. These results are determined as adequate for extension and further application in this thesis.

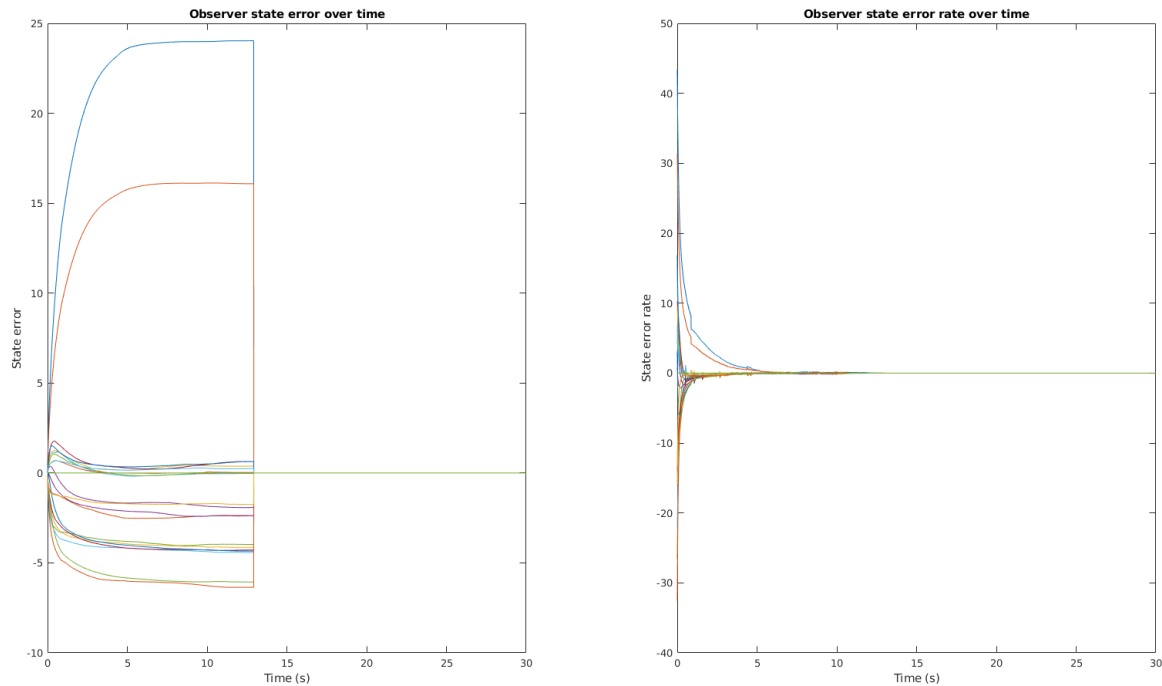


Figure 3-10: Observer performance of all agents.

Up until this point the communication radius of each agent has been scaled dynamically until at most 3 agents are communicating. However, this approach may have practical issues for indoor implementation such as determining how far away communicating agents are. Additionally, if the extra information from a fixed communication region is available intuition leads to the conclusion that it may as well be used. However, when the communication radius of an agent is fixed, interesting things begin to happen. The two methods, variable and fixed communication radius, are shown in Figure 3-11.

In the box test presented each agent starts with a separation of 2 meters. The communication radius in this case must be at least this large for the agents to initialize as a connected graph, one of the assumptions this thesis makes. When the fixed communication radius is scaled up to 4.2 meters, the result is as shown in the right panel of Figure 3-11. The agents have more information available than the dynamic communication case, shown on the left, and choose to interact at the center of the box. Additional insight can be gained by looking at the observer behavior of the dynamic communication radius case.

By looking at Figure 3-12 the estimation convergence time can be seen as approximately 2 seconds. Compared to Figure 3-9 the convergence time is much quicker, by almost 5 seconds. It seems that a reliable source of information about an agent's own relative position, in addition to calculating the formation producing control equation with information from more agents leads to specific behavior. In general, less available information results in more local interaction first, and more available information causes the agents to head directly toward their goal position, and sometimes this will result in more avoidance maneuvers. As shown here, the lack of information available to the agents has a secondary benefit, a so called "speed-up" effect. This effect is not a primary result of the algorithm, as there are initial conditions where no speed-up will be achieved. Regardless, recognizing why this effect occurs

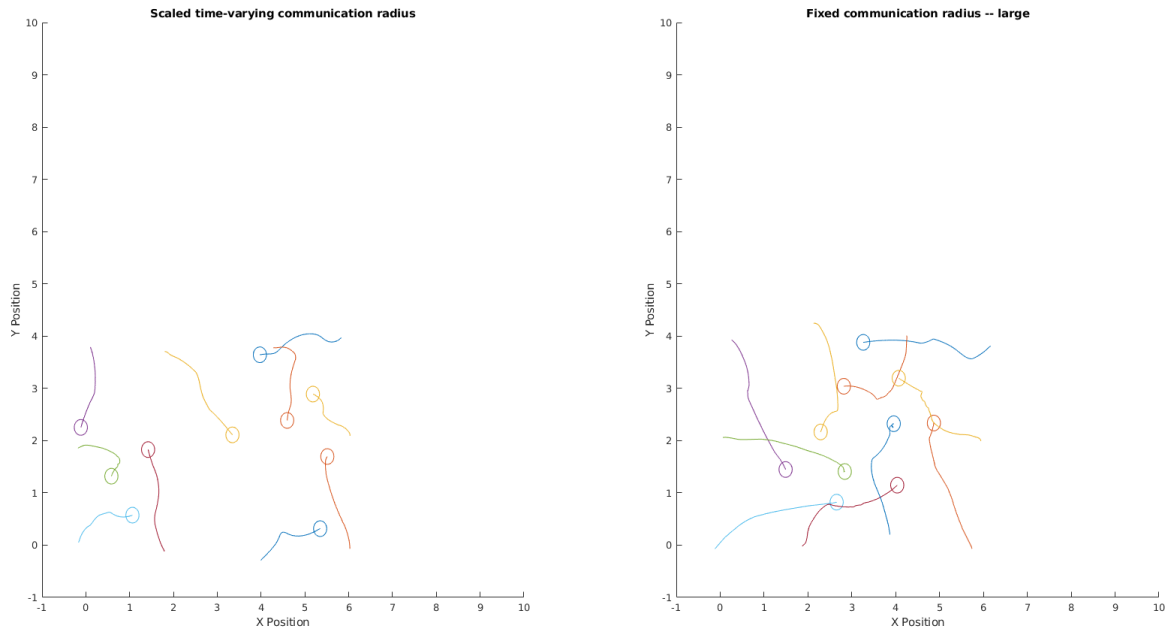


Figure 3-11: Agent behavior comparison of dynamic vs fixed communication radius.

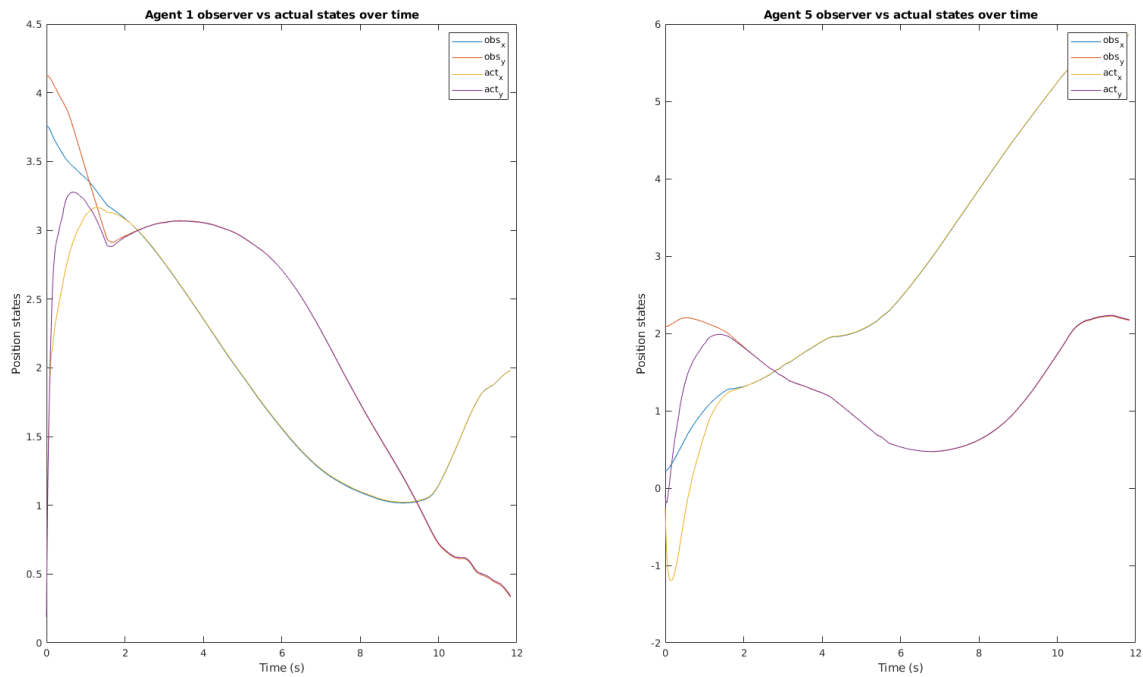


Figure 3-12: Fixed communication radius, $R_{comm} = 4.2$, observer behavior.

can lead to more intelligent formations and less avoidance maneuvers.

The fixed communication radius can be scaled down further to emphasize this conclusion. By scaling the radius down to $R_{comm} = 2.2$ the agent behavior is initially more local than in the larger radius case. This comes at the cost of observer convergence, as seen in Figure 3-13.

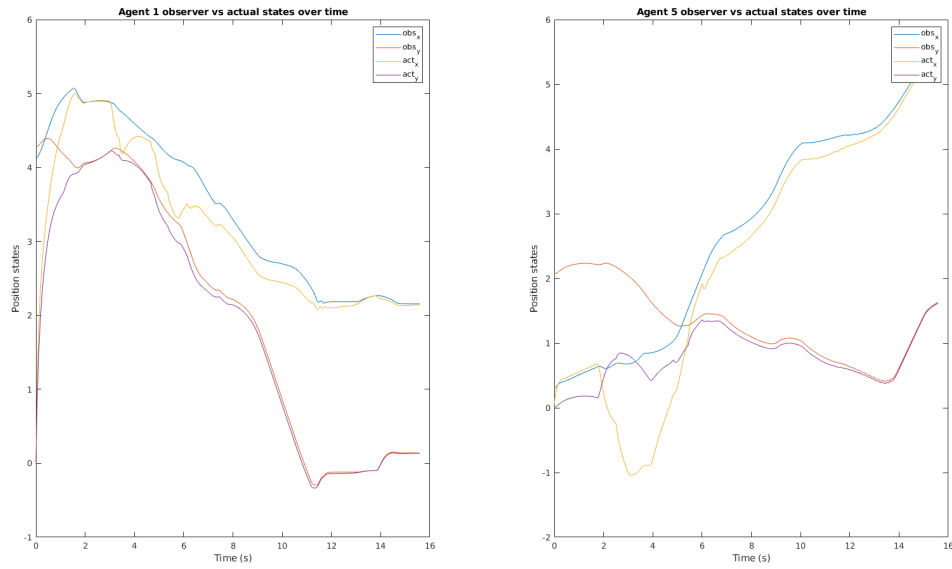


Figure 3-13: Fixed communication radius, $R_{comm} = 2.2$, observer behavior.

Figure 3-13 shows that the estimation convergence time for the box case can dramatically increase as the availability of information decreases, as can be naturally expected. It is worth examining what could be considered the "worst case scenario" for the agents, while also showcasing the formation producing behavior of the algorithm.

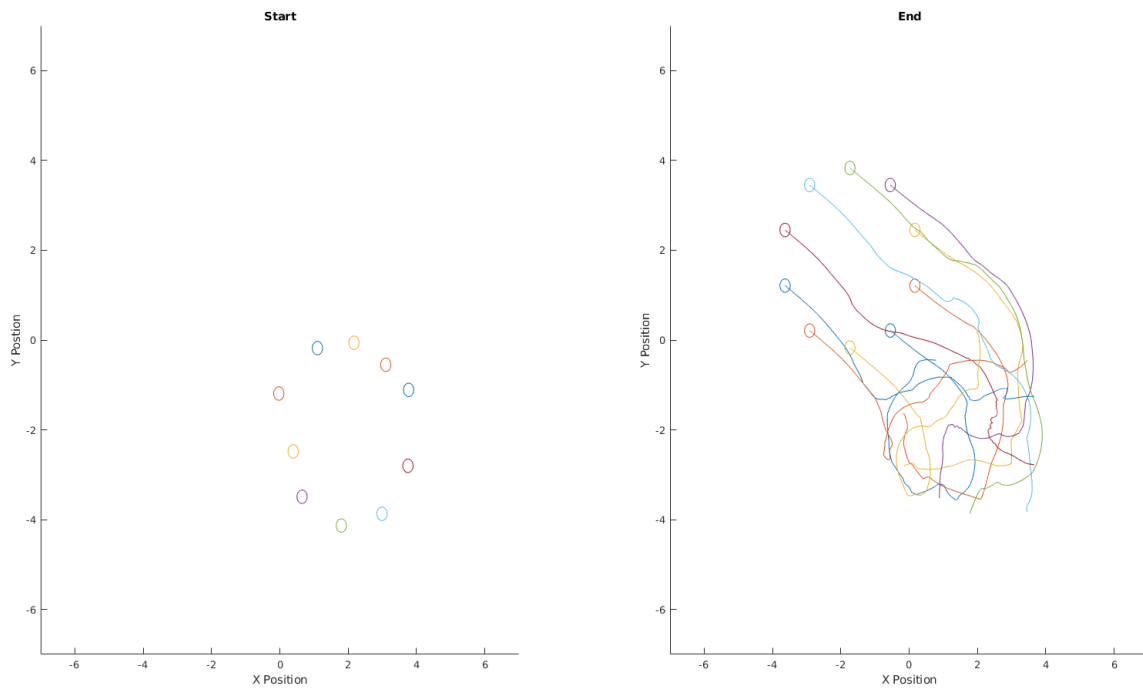


Figure 3-14: Traveling circle formation, proposed "worst case."

In Figure 3-14 a circle of radius 2.0 meters is initialized with the same uniform random disturbance as the box test. The number of communication connections each vehicle has is restricted to 1, their closest neighbor. To enforce this each agent must have a dynamically scaling communication radius to ensure there is always another agent to share information with. The setpoints for each agent require them to swap places with the agent directly across the circle while also traveling as a formation. This setup is run in a Monte Carlo simulation as before, with the same parameters chosen as in Table 3-7. Because of the more complex behavior required, and lack of information available to the agents, the maximum convergence time is extended to 50 seconds.

	Trials	Percentage (%)	Ave. Time (s)
Collisions	280/500	56.0	N/A
Convergence	307/500	61.4	35.93

Table 3-8: Worst case communication, $n_{conn} = 1$, results

Table 3-8 highlights the difficulty the agents have in a low information environment. A large part of the collision avoidance, the emergency avoidance region, relies on an accurate position estimation of the agent's own position and that of the obstacle. It is because of this result that this thesis uses a hybrid communication approach, where the region scales dynamically with a fixed upper bound on the communication radius. There is already significant complexity in the network-decentralized collision avoidance parameters without the addition of variable network-estimation performance. In order to more accurately evaluate the network-decentralized avoidance tuning and the network-decentralized estimation tuning an analytic optimal, or near-optimal approach to tuning the network decentralized observer must be found. This is beyond the scope of this thesis and is left as a recommendation for future work.

3-3 Conclusions

In this chapter a network-decentralized network-estimated approach to multi-agent collision avoidance and formation control was presented. Three simulation environments were designed from the ground up to be modular, scalable, and readable. This allows for rapid testing of various algorithm and vehicle configurations, including Monte-Carlo style analysis. A continuous and scalable approach to obstacle detection that does not rely on discretization was presented in Equation (3-1). At its core this approach relies heavily on each agent being tuned to have appropriate sized collision avoidance regions and avoidance angles for its respective vehicle radius. The relationship between the avoidance regions and respective maneuvers were shown in Figures 3-6 - 3-8 and Table 3-6. This showed that early detection of obstacles is critical to formation convergence and successful collision avoidance. An outcome of this was to define two separate avoidance angles: a smaller one for the outer region to keep agent deviations small, and an optimized angle for the inner region to provide maximum distance from the obstacle. The outer region is re-defined to only detect the exact distance of obstacles within a scaled 120° window. On the other hand, the inner emergency avoidance region is scanned using the estimated position of communicating obstacles. By setting the individual agent avoidance parameters this way and by increasing their radius the requirement

for livelock breaking is removed. Additionally, the requirement for an agent to scan the entire outer region upon detection of an obstacle is also relaxed.

In addition to the analysis of individual agent tuning parameters the network-decentralized portion of the algorithm was also explored. It was shown in Table 3-6 that the network-decentralized estimation of agent positions is not only possible, it can be reliable. Regardless, the current determination of observer gains is not analytical nor optimal, and therefore it is difficult to characterize where a set of parameters for the network-decentralized avoidance algorithm breaks down. This adds undefined complexity to the approach. Another aspect that was addressed was the communication capability of each agent. The network-decentralized algorithm was designed under the assumption that the communication graph would stay connected. There is no attempt, in this thesis, to characterize its performance as a connected union of graphs over time. That being said, the performance of the algorithm for a fixed set of parameters changes based on the communication methodology. It was shown that a fixed communication radius can have both better and worse performance than a dynamic approach. Large fixed radii, relative to the formation, result in collision situations with higher agent density, but with faster observer convergence. Smaller fixed radii have more local interaction, but much slower convergence characteristics. On the other hand, the hybrid dynamic scaling shows a middle ground between the fixed approaches when $n_{conn} = 3$. For a "worst case" communication graph, when $n_{conn} = 1$, the performance dropped dramatically. The observer convergence is important when considering that the emergency detection region of each agent relies on that estimation, and a poor estimation can result in both collisions and lack of convergence. With this analysis and aforementioned alterations the approach can be scaled to 3-D.

3-D Holonomic Simulations

Chapter 3 showed an analysis of a network-decentralized network-estimated collision avoidance algorithm that was also formation producing. An analysis of the baseline algorithm outlined in Chapter 2 was performed and various improvements were added and shown through simulation. At the heart of this thesis is how the algorithm, originally designed for holonomic vehicles, performs with non-linear vehicles. Chapter 3 focused on a holonomic agent in 2-D, however, the vehicle of choice in this thesis is a quadcopter. Therefore it is necessary to scale the approach to 3-D. The implementation, and subsequent analysis, of the algorithm on a holonomic vehicle in 3-D is the focus of this chapter.

4-1 Network-Decentralized Network-Estimated Collision Avoidance in 3-D

At the heart of the avoidance algorithm is the definition of avoidance regions, and the assumption that all moving obstacles will avoid by maneuvering to the right. Both of these features were improved in Chapter 3, and remain important when scaling to 3-D. The avoidance regions are no longer circles, they expand to spheres. This leaves the question of how to detect potential collisions in 3-D. Fortunately, the continuous projection method discussed in Chapter 3 and noted by Equation (3-1) can be scaled nicely. To determine if an obstacle falls within the outer window, the obstacle's position is simply projected onto the XY plane. This results in a loss of information, and some situations where agents will avoid each other even when they are "safely" altitude separated. For the eventual application of this algorithm, indoor flight of multiple quadcopters, this is actually a desirable behavior. The problem of compensating for the propwash of another quadcopter is still an active research topic and is outside the scope of this thesis. So for now, the agents will avoid passing overtop of one another. A quick fix for this issue in the future would be to ignore vehicles outside of a certain relative altitude envelope. However, appropriate disturbance rejection may still need to be applied.

With the collision detection portion scaled to 3-D, the actual avoidance maneuver must be discussed. The concept of "right" and "left" is inherently a two dimensional concept, therefore a 2-D plane must be defined for each collision situation. This problem must also be constrained enough to yield a single solution reliably and the resulting plane must be able to express avoidance vectors with components in all three axis. To constrain the problem enough to form a new plane the XY plane is used, along with the vector from the obstacle intersection point to the agent, and a vector normal to the XY plane.

To perform the avoidance maneuver the 2-D plane that the velocity vector is rotated in must be defined. The vector is rotated relative to this new coordinate system and then transformed back.

$$y = AR(\theta)A^{-1}v_1$$

$$A = \begin{bmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{bmatrix} \quad (4-1)$$

Equation (4-1) shows the rotation of the avoidance vector in a 2-D plane defined relative to the agent-obstacle interaction. The matrix A is defined with v_1 as the vector between the intersection point and the agent, $v_2 = (y_1, -x_1, 0)$ is the XY plane, and $v_3 = v_1 \times v_2$ is the orthogonal vector between them. $R(\theta)$ is the 3-D rotation matrix about the z-axis.

It can be easily deduced that the state vector in Equation (2-2) now expands to 6×1 in dimension to accomodate the agent's position and velocity in the z-axis. This extends speed limiting shown in Equation (2-3) to accommodate the extra dimension. Where some other alterations are needed is in the network-decentralized portion of the algorithm. Equation (2-18) is updated to make the identity matrices three dimensional, shown as:

$$G(t) = H(t)^T \otimes I_3 \quad (4-2)$$

This impacts Equation (2-19) for formation production and tracking of the agents' reference positions. This is done to include the z-axis in the formation control. It is also necessary to increase the dimesion of $\mathcal{C}$ as it appears in Equation (2-12).

4-2 Evaluation of the Extension to 3-D

The previous section showed how the algorithm scales into 3-D. Apart from defining what it means to "turn right" in 3-D, the algorithm is fairly straightforward to scale analytically. Now an analysis will be performed to determine what impacts this may have on algorithm performance. For stress-testing here, the simulation environments built for this thesis, as discussed in Chapter 3 and outlined in [46], work nicely. In the 3-D case a test must be designed in the higher dimension as well. One way to do this is by evenly distributing points on a sphere using the Fibonacci sphere algorithm [50]. Then, similarly to Chapter 3, the agents can swap places with the agent across from them.

In Figure 4-1 one hundred agents are spawned about a sphere of radius 2 meters. Projections onto the XY and YZ plane of this initialization can be seen in Figure 4-2. The sphere

Fibonacci sphere algorithm agent initialization

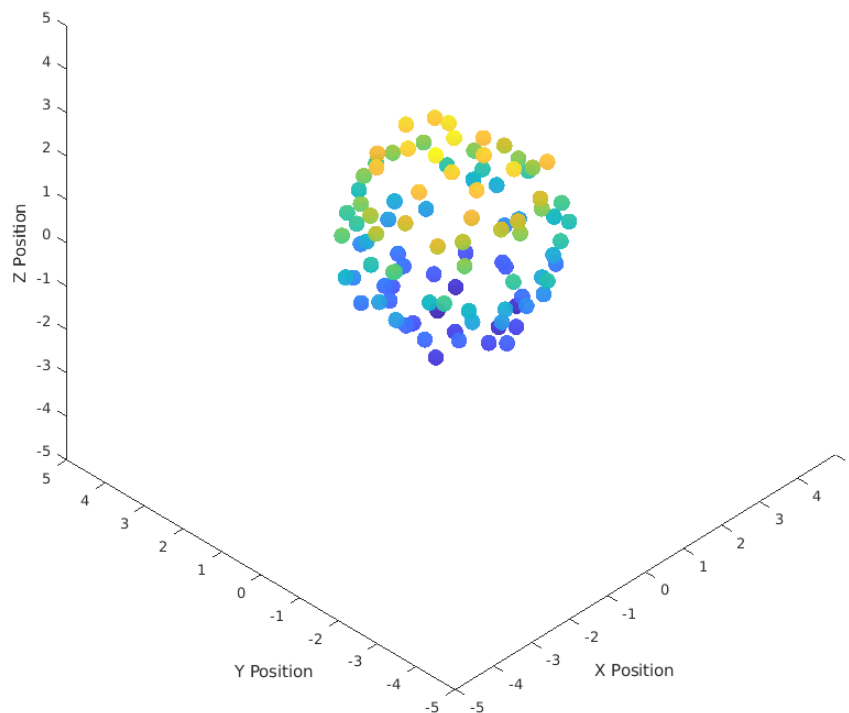


Figure 4-1: Fibonacci sphere algorithm initialization of 100 agents.

initialization places the agents such that there is approximately equal surface area between them. The agents' initial positions are then perturbed by a uniform random variable, ± 0.30 rounded to the second decimal place.

	Trials	Percentage (%)	Ave. Time (s)
Collisions	0/500	0	N/A
Convergence	500/500	100	7.35

Table 4-1: Results for 10 agent 3-D sphere test

Figures 4-3 and 4-4 show one potential outcome of the test that was run. Using the successful parameters from Chapter 3, the tuning proved equally successful in 3-D. This is shown by the results in Table 4-1. Observation of the test graphic gives the impression that there is less chance for collision than that shown in the 2-D test in Figure 3-2. At the same time, some of the agents are actually initialized closer together in the sphere test. By looking at the interagent distance of the unperturbed initial conditions, the agents in the square test are all 2.0 meters from their closest neighbor, but in the sphere test the some agents are as close

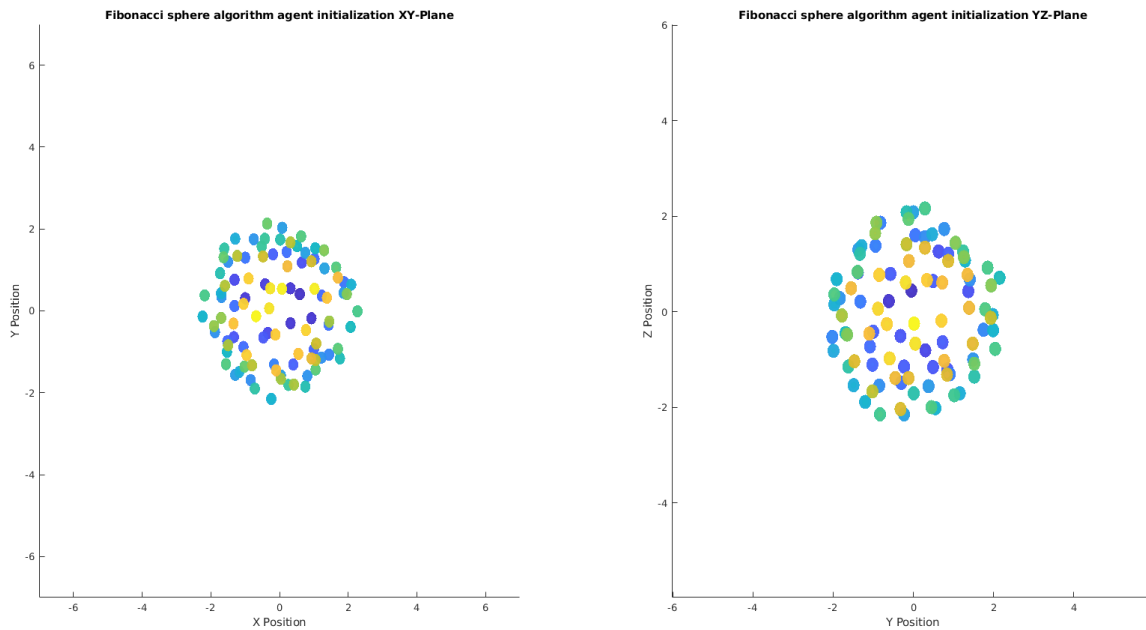


Figure 4-2: Fibonacci sphere algorithm projections.

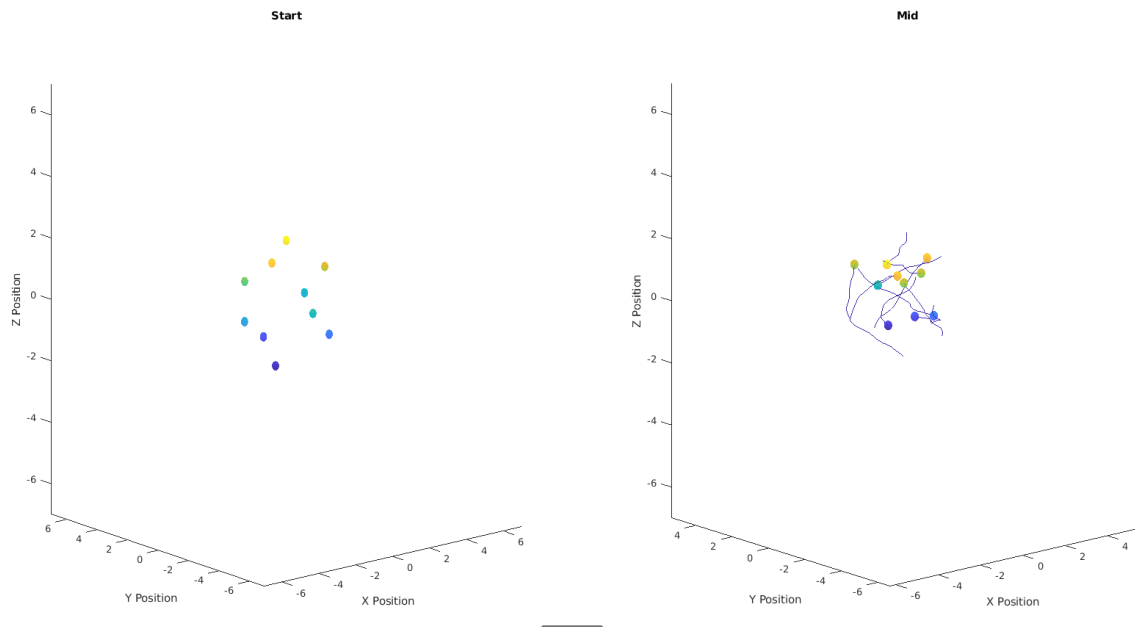


Figure 4-3: Fully network decentralized traveling sphere test with 10 agents, part 1.

as 1.79 meters. This distance is important because the agents are estimating their positions and that estimate is required for successful detection of potential collisions inside their inner detection region. If the agents are initialized too close to each other, they may collide because they do not realize the true position of an obstacle or another agent. However, compared to 2-D, in 3-D the agents have more free space to move about. It is clear that there needs to be some way to define the closeness of the agents at initialization, as well as the local free space

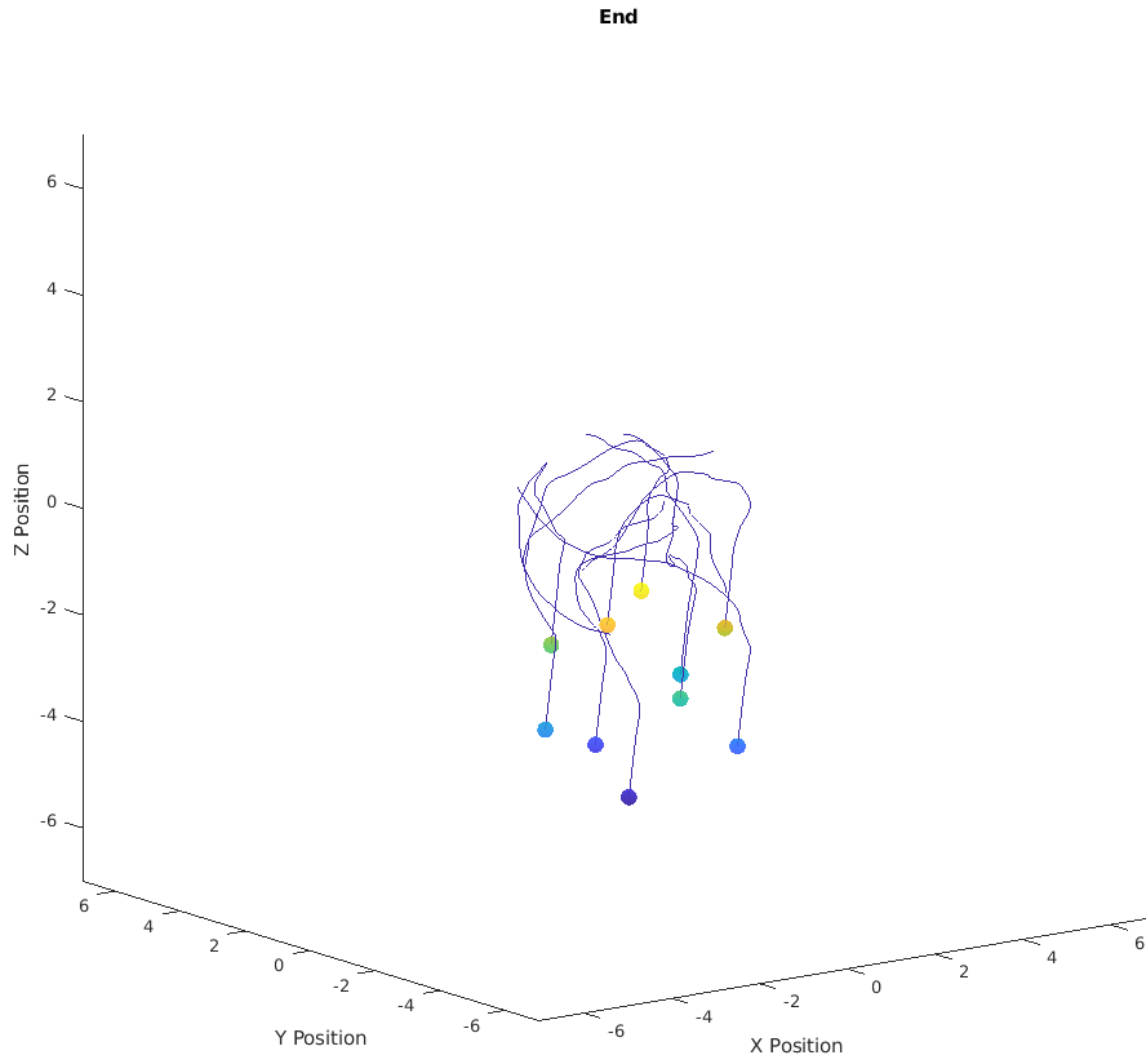


Figure 4-4: Fully network decentralized traveling sphere test with 10 agents, part 2.

respective to each agent. This can be accomplished through the use of two separate metrics: average initialization distance, and average obstacle free-space.

$$\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \|x_i - x_j\|_2 \quad (4-3)$$

Equation (4-3) shows a way to describe the closeness of agents at initialization. This provides useful information that can indicate the cause of collisions. The equation describes the average distance from each agent to all other agents and calculates the average of this distance for all agents. This information is global in nature. Another metric is introduced that is local and requires only information available to each agent. Shown as two separate equations in (4-4) and (4-5) the idea is the same: take the n-dimensional area/volume of communication for an agent and divide that by the sum of the area/volume of all obstacles inside the region.

$$\frac{\pi R_{comm}^2}{\sum_{j=1}^{\mathcal{O}_i} \pi R_{j_{obs}}^2} \quad \forall x_i \quad i \in \{1 \dots N\} \quad (4-4)$$

$$\frac{\frac{4}{3}\pi R_{comm}^3}{\sum_{j=1}^{\mathcal{O}_i} \frac{4}{3}\pi R_{j_{obs}}^3} \quad \forall x_i \quad i \in \{1 \dots N\} \quad (4-5)$$

	10-agent box	10-agent sphere
Ave. init. distance	4.19	2.88
Ave. obs. free-space	9.82	22.91
Ave. travel dist	5.87	4.34
Conv. time (s)	13.52	7.35

Table 4-2: Measure of agent free space in 2-D and 3-D

In Equations (4-4) and (4-5) note that R_{comm} is the communication region of a single agent and $R_{j_{obs}}$ is the smallest n-sphere region fully encompassing the j-th obstacle. $\mathcal{O}_i$ is the set obstacles visible to the i-th agent x_i , taken for all agents N when finding the average as seen in Table 4-2. This makes the metric non-dimensional, allowing it to compare results between 2-D and 3-D tests.

Table 4-2 shows the result of Equation (4-3) in the row "Ave. init. distance". This shows that at initialization the agents in the sphere test are closer to one another than in the box test. The result of Equations (4-4) and (4-5) are shown in the row "Ave. obs. free-space". More testing is presented to determine the usefulness of this metric. However, at first glance the free-space is correlated with the agent convergence time. The average travel distance calculation is shown as more of a reassurance that the tests are demanding similar performance from the agents, e.g. travel some distance to replace the agent across the formation while avoiding other agents and producing a traveling formation with them.

To further test the parameters chosen in Chapter 3 the test can be scaled up to 25 agents. For the 25 agent test the maximum convergence time was increased to 50 seconds to account for extra collision avoidance interactions that may occur. Using the same Fibonacci initialization technique of a sphere with radius 2.0 meters the performance is shown as:

	Trials	Percentage (%)	Ave. Time (s)
Collisions	0/500	0	N/A
Convergence	497/500	99.40	19.13

Table 4-3: Results for 25 agent 3-D sphere test

The performance in Table 4-3 is high, with a very small loss of convergence and is comparable to the 10 agent test shown in Table 4-1. Additional strain can be placed on the tuning parameters by increasing the test to 50 agents. The performance under these conditions is shown in Table 4-4:

With the 50 agent test shown in Table 4-4 initialization of randomly perturbed agents already in collision started to become an issue. Many of the tests need to be re-initialized to a valid

	Trials	Percentage (%)	Ave. Time (s)
Collisions	0/500	0	N/A
Convergence	359/500	71.80	36.50

Table 4-4: Results for 50 agent 3-D sphere test

state. However, the collision avoidance performance remains excellent with 0 recorded collisions. There is a noticeable drop in convergence, which can be attributed to livelock situations and additional complexity due to an increased number of required avoidance actions. An overall look at the performance of different tests of this tuning can be seen below.

	10-agent box	10-agent sphere	25-agent sphere	50-agent sphere
Ave. init. distance	4.19	2.88	2.76	2.72
Ave. obs. free-space	9.82	22.91	7.94	3.57
Ave. travel dist	5.87	4.34	4.56	4.93
Conv. time (s)	13.52	7.35	19.13	36.50

Table 4-5: Comparison of test characteristics and corresponding performance

Table 4-5 allows the value of the metrics presented in Equations (4-3) - (4-5) to be seen clearly. A decreasing trend in the average initialization distance between agents is shown, which alone indicates that the agents will demand faster convergence times from the network-decentralized estimation of their positions in order to avoid collisions. The average obstacle free-space metric, when tied to convergence time, allows the 2-D and 3-D tests to be compared. The difference between a 10-agent test in 2-D and 3-D is clearly displayed and also shows that the 3-D case is utilizing the extra free-space to reduce its convergence time. In fact, when it comes to utilizing free-space to decrease convergence time, interpolating between the tests shows that the 10 agent box test in 2-D is roughly equivalent to a 20 agent test in 3-D. In addition to free-space, the test convergence times are also related to the average travel distance of each agent. It can be seen that as the average obstacle free space decreases the average travel distance increases. This relationship does not convert as nicely across dimensions as average obstacle free space, however, since avoidance actions generally increase the travel distance of the agents, this metric provides an indicator as to the number of avoidance actions each agent must make before converging.

With the relationships between parameters in Table 4-5 defined above, it is now possible to further examine the collision/convergence metrics of the previously shown tests. The tuning of the algorithm's parameters to the values shown in Table 3-7 has thus far been done experimentally. It can be clearly shown that the average obstacle free-space metric indicates where the performance of a particular parameter tuning will break down. The parameters were originally tuned for the 10-agent box test case which yields a free-space metric of 9.82. Given the free space metric of the 10-agent sphere case as 22.91, it can be expected that the performance will be equal with respect to collisions and convergence cases, but with faster average convergence time. This holds, and can be seen by comparing Tables 3-6 and 4-1. For the 25-agent sphere test the free-space metric is 7.94 which is lower than the 9.82 for the box test the parameters were tuned for. Since the original tuning was not optimal there is still a chance that the 25-agent sphere test has similarly successful performance, but there is also a chance that the parameter performance begins to break down. The initial indication

of a break down in parameter performance is exactly what is seen in Table 4-3. This is noted by a small drop in convergence of 0.60%, and also by an increase in convergence time. The break down in parameter performance is further demonstrated by the 50-agent sphere test where the free-space metric falls to 3.57. The difference between the free-space metric of the 10-agent box and 50-agent sphere tests is displayed by another drop in convergence, down to 71.80% as shown in Table 4-4. Lastly, it is worth pointing out that the continued performance of the collision avoidance aspect can be attributed to a good definition of the collision regions and adequate initial spacing between the agents relative to their observer convergence rates.

4-3 Conclusions

The tests shown above consistently result in no collisions, however the tuning is sub-optimal. The current approach relies on the user defining a "worst case scenario" and tuning successfully for that situation. This approach in 2-D has shown good results when scaled to 3-D. A set of new metrics were defined that can help predict, across dimensions, when the performance of a particular set of parameters will break down. Regardless, the static tuning method results in less than ideal performance in low obstacle density environments. If there are not a lot of other agents or obstacles around, then the agents should be able to maintain a higher average velocity. Note that in 3-D there is more "safe-region" volume to work with and this results in a different set of, still undiscovered, optimal parameters than with the 2-D case. A reliable estimation of free-space may allow the vehicles to dynamically alter their tuning online, but this is left for future work. It is also important to note that the simulations here utilize the same observer tuning as in Chapter 3 and the vehicle dynamics and the avoidance algorithm are still running synchronously with each other. This is not practical, and the additional error this introduces when applied to an asynchronous non-linear vehicle will be discussed in Chapter 6.

Lab and Vehicle Design

This chapter addresses the design of an indoor autonomous quadcopter lab. A brief look at other academic lab setups is included. The focus is on the Delft Center for Systems and Control (DCSC) Distributed Robotics Laboratory, its original setup, and a complete redesign this thesis proposes and implements that allows for the better testing of quadcopter swarms. The re-design centers around the elimination of redundant communication channels and the consolidation of lab computers into one server hosted bare-metal hypervisor. A high level overview of the original lab configuration is shown, along with the newly implemented redesign and how this re-design improves upon existing literature with virtualization. Hardware and software considerations are covered, along with appropriate vehicle selection. Lastly, the vehicle selection is discussed in parallel with on-board electronics and the concept of scalability in an indoor lab environment.

5-1 Lab Environment: Closing the Loop

5-1-1 High Level Overview

When designing an autonomous vehicle lab from a control theoretic perspective there is an "outer loop" of information flow that must be defined. This is the high level view of what is happening, and encapsulates a real-time vehicle positioning solution, calculation of trajectories or position setpoints, safety checks, and various status indicators. More commonly, it is not until the perspective shifts to a specific singular vehicle that the attitude control and estimation fusion algorithms become important. Before looking at the DCSC lab setup and proposed changes an overview of other successful multi-vehicle indoor quadcopter labs is performed. This includes, but is not limited to: Massachusetts Institute of Technology (MIT)'s Real-time indoor Autonomous Vehicle test ENvironment (RAVEN) lab, University of Pennsylvania (UPenn)'s general robotics, automation, sensing, and perception (GRASP) lab, and ETH Zurich's Flying Machine Arena (FMA) lab.

The RAVEN lab is MIT's indoor autonomous vehicle testbed. It is 10x8x4 meters in dimension with a structural column in the middle of the space. The lab dimensions are covered with a

moveable configuration of motion capture (MoCap) cameras [51, 52, 2, 53]. The lab can be seen in Figure 5-1 below.

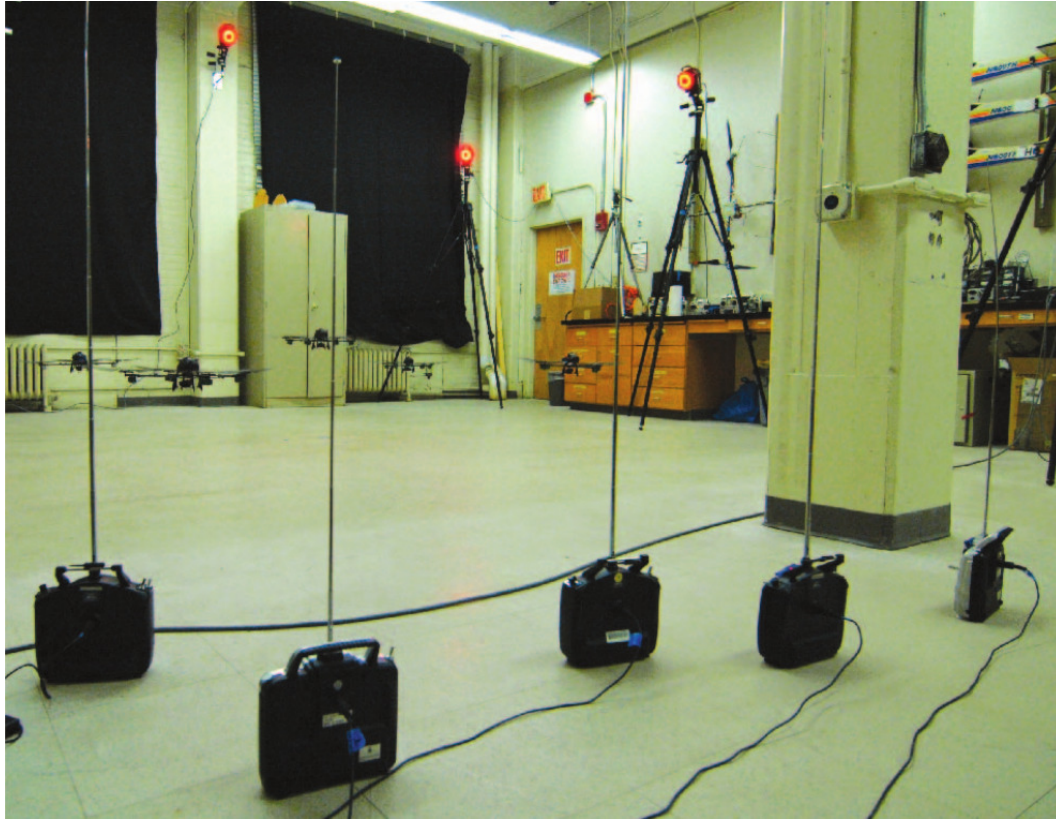


Figure 5-1: MIT's RAVEN lab with RC transmitters and MoCap system [2].

Note the use of RC transmitters in Figure 5-1; the lab setup requires a one-to-one pairing of a transmitter to a vehicle. Each transmitter is attached to its own dedicated central processing unit (CPU). In this setup a CPU is an independent, networked computer. The benefit of this approach is that, theoretically, any RC enable vehicle can be used directly out-of-the-box without modification as long as an acceptable model and controller have been designed in software. This comes with a significant scalability issue, as the number of vehicles able to operate in the space is directly tied to the number of physical transmitters and computers. For each vehicle an RC pairing must be established and a physical computer must be brought online. This becomes more of an issue the more vehicles that are required to fly simultaneously, and with the scalable implementation of higher level monitoring and guidance software. An additional issue is the quadcopter platform the lab implements, the Draganflyer V Ti Pro. This vehicle measures 700 mm along its diagonal, can lift a 100 g payload, and has a flight time of 13-17 minutes with a 2000 mAh battery [2]. The RAVEN lab was built to test long endurance missions and this platform works adequately for that purpose. However, control of quadcopters when they fly overtop of one another is a difficult and still open research topic. With this constraint, the lab is not large enough to support swarming operations utilizing this vehicle.

Another indoor quadcopter testbed is UPenn's GRASP lab. This lab measures 9x5x3 meters

in dimension and has safety netting surrounding the space. This lab space is also covered with a MoCap system, however this setup is fixed [12, 54, 55]. The lab can be seen in Figure 5-2.

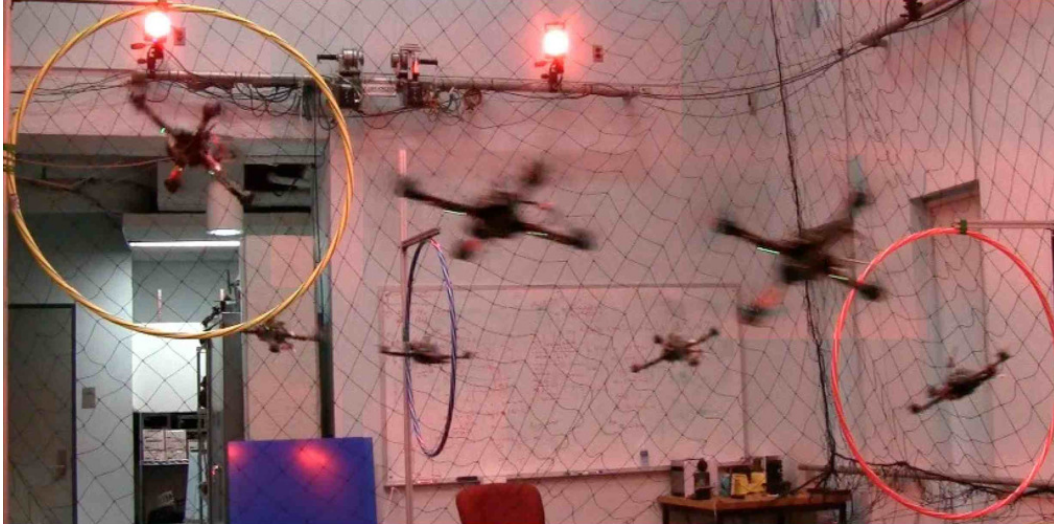


Figure 5-2: UPenn's GRASP lab quadcopter acrobatics [3]

The GRASP lab setup is more centralized than that of the RAVEN lab. A single computer interfaces with the MoCap system, the data is processed using MATLAB with custom Robot Operating System (ROS) bindings, and a set of data is sent to each vehicle via frequency hopping using an Xbee Series 1 over the Zigbee communication protocol [54, 3]. This allows multiple quadcopters to be controlled via a single transmitter, allowing for more scalability. The vehicle of choice here is the, no longer produced, Ascending Technologies Hummingbird [56]. It measures 550 mm along the diagonal, can lift a 200 gram payload, and has an approximately 20 minute flight time at hover state. A downside of this setup is the technical expertise needed for the configuration. Custom ROS bindings for MATLAB; programming of the on-board microprocessor for communication and control; and the parsing of ROS information can be technically challenging. The setup offers the choice of moving ROS off-board or keeping it on-board. The on-board option requires the programming and interfacing of an additional microcontroller to run the necessary libraries. Since the platform is no longer available, additional difficulty would be presented by needing to build an adequate System on a Chip (SoC) for the vehicle. This would also present difficulty for future users, unless additional effort was put into designing a user-friendly interface. This lab also runs into the issue of overlapping vehicles because of their size relative to the total lab volume.

With the GRASP lab it was shown that a nested feedback controller designed around the vehicle hover envelope yields adequate performance. Different sets of gains are used depending on the desired tradeoff between performance and robustness. It was found that the vehicles can achieve a velocity of at least $3.6 \frac{m}{s}$ and that the hover envelope performance does not increase dramatically over 35 Hz [3].

The last indoor quadcopter testbed that will be discussed here is ETH Zurich's FMA. This lab is the largest seen in literature so far, at 10x10x10 meters, and has safety netting and padded flooring. A fixed MoCap system is used to track vehicle positions [57, 58]. The lab

can be seen in Figure 5-3 below.



Figure 5-3: ETH Zurich's FMA testbed. [4]

The FMA lab setup can be seen in Figure 5-3. This setup is the most scalable compared to the other two. A networked computer receives MoCap data which determines vehicle desired attitude and thrust values. This information is sent to an off-board "copilot" computer which checks vehicle commands against pre-defined safety settings. Every piece of information is broadcasted over multicast User Datagram Protocol (UDP) using a custom designed "protocol bridge" which allows for rapid scaling of both the off-board infrastructure and the number of vehicles that can fly in the lab. The main vehicle used in the lab is the same as in the GRASP lab, the Ascending Technologies Hummingbird. Again, measuring 550 mm along the diagonal and with the capacity to lift a 200 gram payload. The strength of this approach is the use of multicast UDP, where this is the first lab in literature to implement the protocol. The use of UDP allows for a higher sampling rate and lower latency than other presented approaches. The size of the testbed coupled with the scalability of the infrastructure overcomes most of the issues seen in both the RAVEN and GRASP labs. The major downside of this approach is the technical expertise required to build the testbed and to implement new control algorithms. Custom software/hardware is needed for the protocol bridges and copilot computer, as well as custom communication parsing to be implemented on the vehicle. This all requires a team, time, and specific skillsets. Additionally, since the vehicle is no longer in production the difficulties expressed in the description of the GRASP lab still hold.

With the above descriptions of three labs found in literature a general diagram of a typical design is shown in Figure 5-4:

Figure 5-4 shows a vision based MoCap system that interfaces with a computer or multiple computers. This generates inputs for the inner attitude control loop running on each

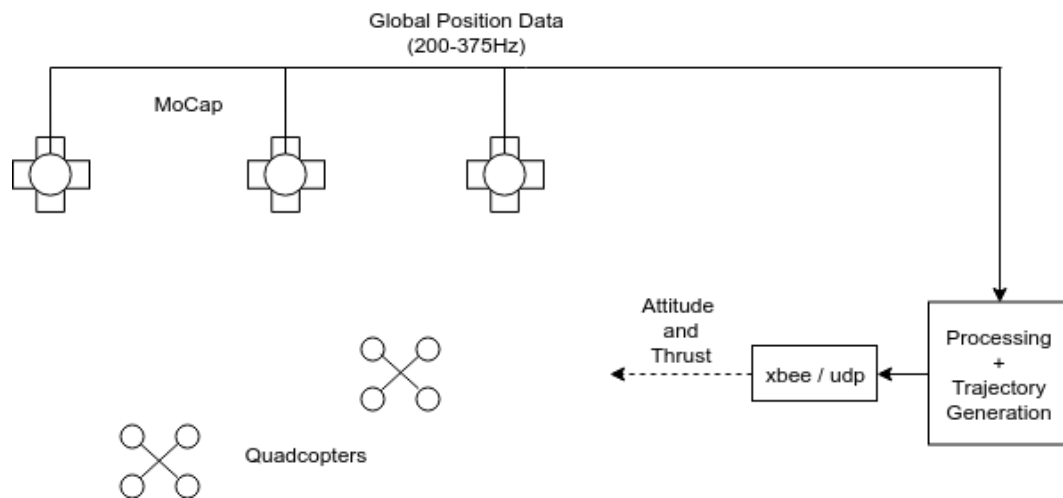


Figure 5-4: A typical indoor quadcopter lab for autonomous control testing.

quadcopter. The setpoints are sent wirelessly via Xbee modules or UDP transceivers. The approach used in the RAVEN lab was omitted here due to its lack of scalability and other viable options.

5-1-2 DCSC Distributed Robotics Lab: Original Design

With an analysis of other labs presented above and a general outline of an indoor quadcopter lab shown in Figure 5-4, a look at the original DCSC Distributed Robotics Lab is presented.



Figure 5-5: Robotics test area with protective netting.

Figure 5-5 shows the Distributed Robotics Lab which has a protective net cage, a fixed MoCap system, and measures 9x4x3 meters in physical space. This makes the space similar to those described above, but with a below average volume.

In Figure 5-5 the lab's MoCap system can be seen mounted above the floor on a metal frame. The utilized system is the Optitrack 17W with a 10-camera setup. Each camera has a 70°

field of view (FOV) with a sampling rate of 360 frames per second. Because of the overlapping FOV cones produced by the cameras only a reduced subset of the space is usable. For accurate position solutions only an approximately 9x4x2.2 meter region can be used. These position solutions require four retroreflective markers to be placed on a vehicle to produce a unique rigid 3-D body in the corresponding Motive software. This software can then solve for the absolute position of the vehicles at 120 Hz.



Figure 5-6: Position and trajectory computation setup.

The MoCap cameras are wired via ethernet cable to a dedicated physical computer which runs the Motive software [59]. The computing setup is shown in Figure 5-6. Here one computer receives the position solution which is connected to a router. The user administering the test has the option of using their own laptop or another fixed computer to connect over WiFi to the router broadcasting the position solution. The use of a personal laptop results in an incorrect timestamp. The desired algorithms are run on this information and then sent wirelessly again to the physical vehicles in the lab environment. All wireless communication is done via Transmission Control Protocol (TCP).

5-1-3 Lab Re-design

Given the description of the original DCSC Distributed Robotics Lab setup a number of improvements will now be proposed.

The first issue is with redundant communications. Currently two wireless pathways are required for relevant information to be passed to a quadcopter. Each path relies on a TCP based connection, and every TCP connection requires a 4-way handshake before information is passed, as opposed to UDP which is send-and-forget. This adds significant overhead to vehicle operation, where a 100 Hz quadcopter command rate is desired. Combining the Motive software and base-station on one computer would cut the current communication costs in half.

Additional issues arise if multiple vehicles need to be tested simultaneously using TCP based communication. To perform tests with more than one vehicle a separate wireless network interface card (WNIC) is required on the base-station side for each controlled vehicle. This

can lead to a situation where a laptop base station has a one-to-four USB-A splitter where each of the four ports is a wireless network adapter. This only allows four vehicles to fly simultaneously, quite a small swarm, but will be used as an illustrative example. To get an idea of what happens on the computer side it will be assumed that both the base-station and the computer with Motive are running a user version of a Linux based operating system (OS). For the base-station the OS will acknowledge that a USB device is attached to the computer and requests to-and-from that serial port will be cataloged by the OS alongside all other housekeeping operations the computer must perform. The kernel manages this scheduling and the user is unable to alter this behavior. This results in highly variable latency with even just one network adapter [60]. In this scenario four separate network adapters are sharing the same serial port which would further increase the variance and overall latency. It is quite clear that TCP is not an ideal protocol for swarming operations.

Another problem occurs because of the redundant communications and that is with the assignment of a timestamp by the ROS libraries. When vehicle information is transmitted wirelessly from the Motive computer to the base-station any further operations that are performed using the ROS libraries alter the timestamp. This timestamp is an important part of many built-in packages, such as the Point Cloud Library (PCL) library. Having an incorrect time can lead to difficult to diagnose bugs, issues for trajectory design, estimation, position solutions and other time sensitive operations.

Lastly, protection of the information present on the computers should be considered. The computer running the Motive software is a local administrator account with no active backup. If the physical hard drive crashed it could take a while for the software to be reinstalled and calibrated. When it comes to the base-station, one of two computers can be used: a desktop fixed in the lab, or a personal laptop. The computer fixed in the lab runs a user image that is pulled down from a local active directory. This active directory has limited user permissions and is blocked from installing 3rd party software. The alternative is to use a personal laptop, which depends on the user to determine (and use!) a suitable backup method. There is also an additional router and N network adapter dongles that present additional points of failure.

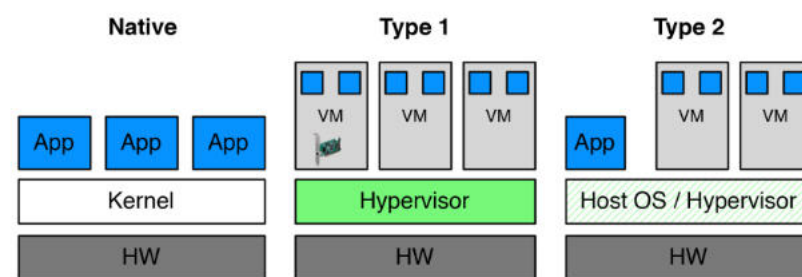


Figure 1: Hypervisor Designs

Figure 5-7: Typical OS vs hypervisor types [5].

A suitable alternative to the present issues is to reimplement the lab back-end on a Dell PowerEdge r740 server [61]. A typical configuration may include 8 TB of solid-state drive (SSD) memory with 128 GB of double data-rate 4 (DDR4) random-access memory (RAM), 4 gigabit ethernet ports, and 4 USB-A 3.0 ports. With these server resources VMware ESXi can be implemented as a Type-1 "bare-metal" hypervisor. A Type-1 hypervisor runs directly

on the server in place of an operating system and spins up virtual machine (VM)s of full OSs. A brief comparison of a traditional "native" OS, Type-1 hypervisor, and Type-2 hypervisor can be seen in Figure 5-7.

The VMs can have fixed resources or additional resources can be applied dynamically by the hypervisor. Virtual gigabit ethernet connections can exist between VMs, with the added capability for designing complex virtual networks, eliminating the latency of an additional router and the physical hardware. This also allows both the Motive software and the base-station to exist on the same physical machine, effectively clearing the table in the lab shown in Figure 5-6. The VMs can be easily backed up by creating "snapshots" in the hypervisor management screen and reverting to previous snapshots takes a minute or two. Students in the lab can spin up personal VMs to act as development environments and easily create backups with their own snapshots. A real strength of this approach is the ability for anyone on the lab network to secure shell (SSH) or virtual desktop infrastructure (VDI) into a hypervisor VM from their own personal laptop. This allows multiple people to access the same centralized and compact set of resources simultaneously.

Documentation of lab resources also becomes more efficient and allows for growth over time. A local wiki can be spun up in the hypervisor as an independent web server VM. Any VM in the virtual network can pull down and potentially edit the documentation for the lab, adding their own work and referencing the work of previous students. This allows for the accumulation of generational knowledge in the lab.

With the centralization a server setup provides it becomes possible to remove the redundant router in the current lab setup. A single Ubiquiti Unifi AC SHD access point (AP) can be used with 800 mbps at 2.4 GHz or 1733 gbps at 5 Ghz [62]. Over multicast UDP this supports as many vehicles as needed in the lab. An additional feature is the 4x4 multi-user, multiple input, multiple output (MU-MIMO) capability which allows for four simultaneous TCP connections to be processed, as opposed to the traditional first in, first out (FIFO) pipe. Regardless, the use of UDP is preferred. The AP has power over ethernet (PoE) capability, removing yet another cable from the setup.

As an accessory, a set of Unifi G3 Pro cameras can be mounted in the lab. These devices are also PoE and can be configured to continuously log test video in 1080p on a dedicated VM on the hypervisor. A centralized hypervisor environment provides a scalable all-in-one back end solution for an autonomous vehicle lab. The approach described above was implemented in the DCSC lab and the server setup is shown in Figure 5-8.

	ESXI Passthrough	Laptop Passthrough
Radio Packet Loss	9.4 %	11.3 %
Radio Latency	4.63 ms	21.92 ms
Radio Sampling Rate	639 pkt/s	75 pkt/s
Motive Latency	0.6 ms	0.6 ms
Motive Sampling Rate	120 Hz	120 Hz

Table 5-1: Performance data for two different methods of accessing the lab environment.

In Figure 5-8 the implementation of the virtualized lab environment described above can be seen. In the figure a rack mounted Dell r740 server plus a Unifi switch is configured with a Unifi AP, and G3 Pro camera. The keyboard and monitor placed on top of the server rack are



Figure 5-8: The entire virtualized lab environment hosted on a Dell r740 server.

to access the integrated Dell remote-access controller (iDRAC) environment which allows for configuration of the server itself, a level of abstraction below the hypervisor. This replaces the entire table of computers shown in Figure 5-6. Table 5-1 shows performance for two different

ways that the virtualized lab environment can be setup. Note that the vehicle used here is the Crazyflie 2.1 which will be discussed in more detail in Chapter 5-3 and Chapter 7. The first method, noted as "ESXI Passthrough", has the vehicle communication radio attached directly to the server which is then passed through to the hypervisor environment. It is shown that this setup has the least latency and the highest sampling rate when communicating with a vehicle. The second way that the lab can be configured is noted as "Laptop Passthrough" where the radio is connected directly to a student's laptop computer. In this case the personal laptop must be on the same network as the virtual lab environment, then the student can use a VM client, such as VM Workstation 15 [63], to access the server and pass the radio commands over the network. While the performance is worse with the laptop passthrough approach, it allows the communication radio to be placed in different locations in the lab. Additionally, it allows for the server to be moved out of the immediate lab space.

5-2 Vehicle Selection

With a back-end for the lab environment described in the previous section, vehicle choice will now be discussed. The vehicles detailed in the previous section were the Ascending Technologies Hummingbird, for the GRASP and FMA labs; and the Draganflyer V Ti Pro, for the RAVEN lab. Neither of these vehicles are still in production as of January 2019, but their features can still be analyzed. One method to perform this analysis is by looking at size, weight, and power (SWaP) characteristics [64]. However, a modification will be made to this to include "complexity" as an additional parameter since the end goal of vehicle design in an academic setting is widespread utilization by the student body and researchers.



Figure 5-9: Ascending Technologies Hummingbird platform [6]

The Hummingbird platform measures 550 mm on the diagonal with a maximum payload of 200 grams, maximum takeoff weight (MTOW) of 710 grams, and a 20 minute (no payload) battery

life using a 2200 mAh 3S Lithium polymer (LiPo) battery [65]. The lift capabilities are good, and this allows for a number of peripheral devices. A typical stereo vision camera is similar to the BlackBird 2 which is 25 grams [66]. A 360° Light Detection and Ranging (LIDAR) system like the RPLidar A3 described in Chapter 2 is 190 grams. So this platform is capable of implementing either sensor. In addition to lift capability it is important to see if the platform can actually house the electronics. Many modern commercial off the shelf (COTS) ready-to-fly vehicles have configurations that do not allow for peripheral sensors, or restrict the user to that specific brand for auxiliary equipment. Figure 5-9 shows that the vehicle has a bottom mounted camera bay and can be built up vertically as well. The battery life is also quite long which is related to the size of the vehicle. Larger props produce higher thrust at lower rotations per minute (RPM)s, require lower KV motors, and are overall more efficient than smaller vehicles [67]. However, the benefit of a larger vehicle comes with reduced vehicle free-space. If a 10 vehicle swarm is assumed and vehicles cannot pass over top of one another then a 550 mm frame in the DCSC lab results in $12.3 \frac{m^3}{veh}$ of maneuverability, using the free-space metric in Equation (4-5). In a best case scenario, where the quadcopters have similar performance to holonomic agents, this aligns with the performance of an approximately 18-agent sphere test shown in Chapter 4. In practice the performance will be worse. Lastly, since the vehicle is no longer in production the electronics complexity must be considered. To implement a similar platform the avionics would need to be entirely rebuilt. The Hummingbird has 3 main pieces of electronics: the AscTec AutoPilot Board, an Intel Atomboard v3, and a Xbee Series 1. This adds a high level of complexity to the design of which this will be discussed more in the next section on flight controller selection.



Figure 5-10: Draganflyer V Ti Pro [7]

The RAVEN lab utilized the Draganflyer V Ti Pro as their primary quadcopter platform. This vehicle measures 700 mm on the diagonal, with a 100 gram payload, and a flight time of 13-17 minutes. The exact weight of the vehicle is not know, but it is less than 500 grams. A standard out-of-the-box configuration of the vehicle can be seen in Figure 5-10. The vehicle is sparse but has strong symmetry which makes modeling and control simpler. If the goal is to test higher order algorithms then a simple to control vehicle is a good feature. The ability for

custom electronics is limited because of the skeleton style frame, but an additional platform can be mounted below the flight controller seen in the center of the vehicle in Figure 5-10. Similar with the Hummingbird this vehicle has good battery life due to larger propellers and motors. This comes with the similar free-space consequences as with the Hummingbird. In a 10 vehicle swarm, where the vehicles cannot pass over the top of one another, and where the total volume is the same as the DCSC lab the free-space is $7.3 \frac{m^3}{veh}$. This is approximately the same performance as the 25 agent test. The onboard electronics consist of a custom made flight controller interfacing with a RC receiver circuit. As discussed earlier, the use of raw RC commands for control does not scale well so that is not a desired approach. Even without the RC receiver circuit replacing the flight controller would require a custom built circuit and programming low level control at the microcontroller level. This adds complexity to the design

With the complexity and SWaP characteristics in mind a platform is presented that focuses on modularity and user convenience. The Lumenier QAV-R2 is a frame 250 mm along the diagonal, that focuses on symmetry and free space for custom payload configurations. This vehicle frame is shown in Figure 5-11:



Figure 5-11: Lumenier QAV-R2 frame [8]

The symmetry is proven in Chapter 6 and allows for implementation of a hover envelope controller as the principle axis aligns with the body frame of the vehicle. Otherwise there would be non-linear coupling along the axis with roll, pitch, and/or yaw creating an additional unintended roll, pitch and/or yaw moment. Additionally, the vehicle material is carbon fiber

with upper and lower plates separated with standoffs. The additional holes in the body allow a user to secure any manner of peripheral devices, including third-party flight controllers. The mounted motor combination is a 2206-2300KV brushless direct current (DC) motor with 30 A electronic speed controller (ESC)s and 5x4.3x3V1S polycarbonate propellers. The propellers are tri-blade which reduces the overall size of the vehicle while producing less vibrations and more thrust per revolution. Currently the lab utilizes one Parrot Bebop vehicle [68], but the platform has a minimal amount of modularity. It also does not have the capacity for dynamic movements because of its motors. Although the size of the vehicle is slightly larger than the QAV-R2, at 250mm, and utilizes tri-blade propellers, its motors only operate in a range of 7500-12000 RPM. This is half that of the 2206-2300KV motors mounted on the QAV-R2 frame. The consequence is a smaller thrust-to-weight ratio. While the Bebop platform will operate well within its hover envelope it cannot take on demands that would test a more non-linear controller, for example trajectory tracking through an inversion.

The chosen combination exhibits modularity, accessibility, and dynamism. It is for these reasons that the pairing of the QAV-R2 platform, 2206-2300KV motors, and 30A ESCs was chosen.

5-3 Flight Controller Selection

The vehicles implemented in the RAVEN, GRASP, and FMA laboratories were discussed in the previous section. As mention before neither the Hummingbird nor the Draganflyer are still in production. Regardless, the frame size of these vehicles is inadequate for the available test space in the DCSC lab. The on-board electronics for both vehicles were briefly discussed. It was found that the avionics were mostly custom implemented. For example, the Draganflyer uses a vehicle specific flight controller circuit. The on-board microcontroller is not intended to be re-programmed as seen in the implementation strategy used by the RAVEN lab. That lab used individual RC transmitters each paired with dedicated computers where all of the software was implemented. For the Hummingbird platform more advanced electronics were utilized. The heart of the on-board electronics is again a custom flight controller auto pilot board. This is where the low-level motor commands are calculated. Also on-board is an Intel Atomboard OS that interfaces with both the auto pilot board and an Xbee for communication. Again, the software running on both of these devices requires significant development time. The Atomboard will most likely also require a customized Linux distribution to meet power and computing requirements or a real-time operating system (RTOS) to limit variance in OS latency.

A search was done for COTS systems that meets the DCSC lab requirements. Only a few options were discovered, but were inadequate for different reasons. The first is the RC Benchmark Otus system [69]. This is a whole system that pairs with a MoCap system. It has pre-designed electronics that are configured with higher level libraries [69]. The choice to not use this system was due to the platform size, the unnecessarily large tracking marker, and resulting dynamics limitations placed on the vehicle.

Another platform that was considered is the Crazyflie 2.1 [9]. This is an all-in-one system is shown in Figure 5-12 and measures 127 mm on the diagonal. It has platform specific attachments including one for a local positioning solution via optical flow sensing. The



Figure 5-12: Crazyflie 2.1 vehicle [9]

bindings to interface the built-in electronics with the ROS libraries are all open source, but not currently supported by the developers and not significantly developed. Other limiting factors are the overall payload capacity of 15 grams, the inability to use other COTS products, and the dynamism of the vehicle. The available payloads were minimal at the time this thesis was conducted. However, as of 13 May development is being conducted to allow for an on-board ESP32 which would give the vehicle the ability to perform image processing and on-board real-time avoidance calculations [70]. Examining the shortcomings further, the thrust at 90% capacity is approximately 54 grams. This is two times the mass of an unburdened vehicle. When compared to the QAV-R2 setup proposed above it will be shown that the 2206-2300KV motors provide a thrust that is over five times the vehicle mass. This allows for highly dynamic maneuvers. Despite the shortcomings the major upside of the platform is that it is an all-in-one COTS product that has fast zero-to-flying time. For this reason in addition to acquisition issues for the desired QAV-R2 setup, described in more detail below, testing in this thesis is performed using the Crazyflie 2.1 platform.

This leaves the QAV-F2, the more modular approach which requires system integration of multiple COTS products to produce a final system. As mentioned earlier the frame combination chosen was the QAV-R2, 2206-2300KV motors, and 30 A ESCs. The on-board avionics chosen will now be discussed, along with their benefits and downsides.

Figure 5-13 shows the nested PID controller used to control a quadcopter about its hover envelope. This logic can be split a number of ways depending on how much computation is

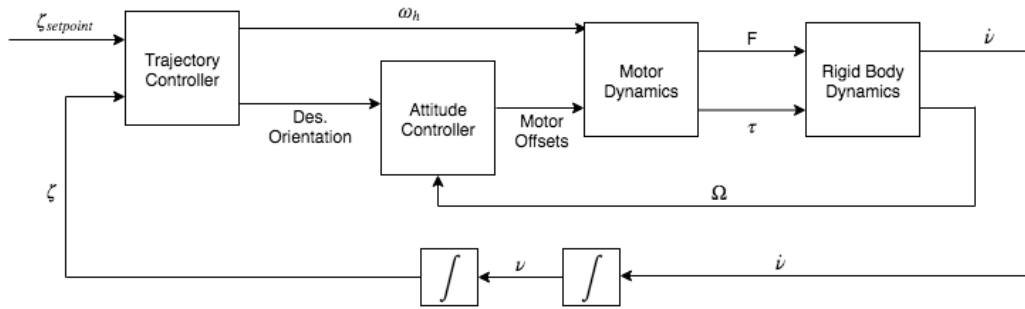


Figure 5-13: Quadcopter hover envelope control logic.

done on-board vs. off-board. This is shown in the lab discussion in the previous section. The RAVEN lab is entirely off-board. The FMA lab shifts slightly more computation on-board with a body-rate controller computation, and the GRASP lab changes the computation split based on the application. The split between on-board and off-board control is a spectrum, as long as the controller is implemented similarly to Figure 5-13 the hover envelope approach will work. One consequence of this split is the required vehicle payload, where the more on-board computation that is required the more payload may be required. Of course custom circuits can cut down on this payload requirement, but this increases complexity and user accessibility. That being said there are some COTS flight controller (FC)s that can implement this approach. The FC that was chosen is the Pixhawk 4 Mini, first released in December 2018 [71]. This is an all-in-one FC that fuses two separate inertial measurement unit (IMU)s along with an external position measurement using a dedicated STM32F765 microcontroller. The unit is designed to fit on 250 mm quadcopters. The FC runs PX4 firmware with the 1.9.0v development build as of May 2019 [72]. The firmware is developed in open-source with periodic stable releases and allows users to edit low level functionality. Additionally, a higher level interface is included that allows a user to set FC parameters instead of altering the firmware directly. The firmware can be built independently of the physical flight controller and has bindings in the Gazebo v8.0 simulation environment [73]. These bindings are through ROS topics which allows a user to run the exact same code in simulation before implementation on the physical vehicle. It must be noted that communications between the Pixhawk and a base station can prove challenging. The initial idea was to use a Raspberry Pi Zero W, which is 60x30x5.4 mm in size, only weighs 9 grams, and draws less than 1 A [74]. However, the libraries to create the Mavlink packets for communication require at least ARM7 instruction set architecture (ISA) and the Pi Zero W has ARM6 ISA. Numerous dependency workarounds were attempted without success. Without rewriting critical parts of the library, the communication would not work. Therefore the communications were implemented on a Raspberry Pi 3B+. The downside here is the 3B+ is much larger at 85.60x56x21 mm, weighs 45 grams, and can pull up to 2.5 Amps [74]. Therefore, the implementation of the companion computer communication library on a Pi Zero is left as future work.

When looking at the potential levels of complexity for a FC there are a few "classes" that describe the user interface. One way to think of this is by decreasing levels of user-complexity: pure SoC; open-hardware with open-source firmware; and a fully abstracted system. A pure SoC system offers the greatest power, size, and weight reductions at the cost of complexity. A developer will need to design a suitable circuit board to interface with and program the

microcontroller. This presents a steep learning curve for new users. The open-hardware, open-source firmware approach is what is described above with the Pixhawk 4 Mini, PX4 combination. This system already has working firmware and FC software, but allows the user to alter both. Additionally, the open-hardware has additional serial peripheral interface (SPI), universal asynchronous receiver-transmitter (UART), and inter-integrated circuit (I2C) interfaces that can be used by any manner of COTS sensors. Last is the fully abstracted system. This is similar to the Bebop or Crazyflie. In these cases the on-board electronics are fixed, vehicle-specific, and not editable. There are also limited, usually sold by the same company, peripheral sensors that can be integrated with the setup. This is usually the quickest way in which to test simple hover envelope 3-D algorithms, but does not scale well for non-linear applications. Since the desire is to implement a modular FC that allows for a wide range of quadcopter tests the open-hardware, open-software solution is the best fit.

Quadcopter Dynamics and Identification

A network-decentralized network estimated algorithm was presented in Chapter 2. A common assumption in this field of study is that the agents must be homogeneous and holonomic. The main goal of this thesis is to assess the ability of this algorithm to be scaled up to 3-D, and then be applied to a non-linear non-holonomic set of vehicles. Chapter 3 discussed tuning methods for the algorithm and various improvements that resulted in reliable collision avoidance and formation production. Chapter 4 scaled the algorithm to 3-D and analyzed the performance of the parameter tuning first presented in Chapter 3. The work up until this point has been focused on holonomic systems. This chapter seeks to further expand the use case of this algorithm. The quadcopter, a non-linear, non-holonomic vehicle, is detailed and the applicability of work previously presented is discussed.

6-1 Quadcopter Dynamics and Control

6-1-1 Dynamics

There have been many attempts to mathematically characterize the behavior of quadcopters [10, 75, 76, 77, 78, 3, 79]. What most approaches have in common is the assumption that the vehicle is a rigid body, symmetric, and that the center of gravity (CoG) coincides with the origin of the body frame. Variations in modeling tend to be which disturbances are modelled, how coordinate frames are defined, and the transformation between those frames. There is a subset of work, mentioned in Chapter 5, that tries to determine what modeling and control concepts are actually necessary to achieve an adequate level of vehicle performance. One group that has been successful at this is the general robotics, automation, sensing, and perception (GRASP) lab at University of Pennsylvania (UPenn) [76, 3, 79, 12, 55, 54, 80]. The work in this thesis follows their underlying assumptions. With that said, it is important to lay the foundation for the rest of this chapter, and that begins with reference frames.

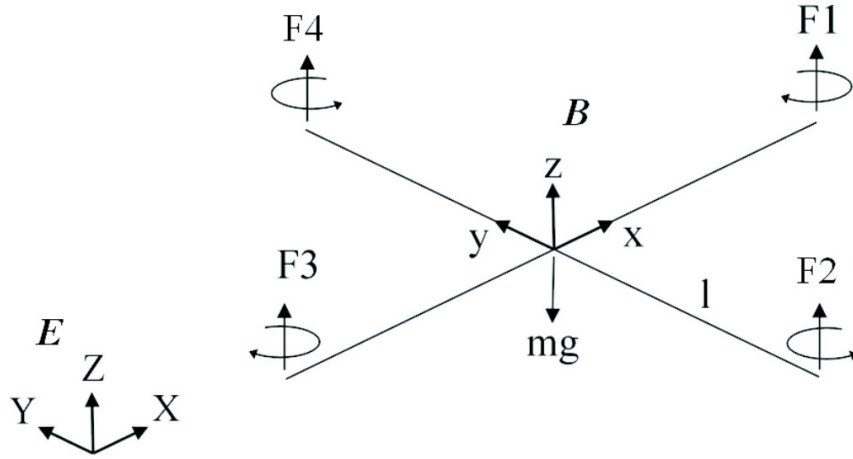


Figure 6-1: Diagram of a quadcopter with two reference frames: inertial (E) and body (B) [10].

Figure 6-1 shows two reference frames, the Earth fixed frame (EFF) $\mathbf{E} = \{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$, and the body fixed frame (BFF), $\mathbf{B} = \{\vec{b}_1, \vec{b}_2, \vec{b}_3\}$. For this application the EFF assumes the vehicle operates in a small enough area such that the Earth can be modelled without curvature. In actuality the EFF is a point in an indoor lab or simulation that is defined as the origin. It is also mathematically convenient to have a second reference frame, $\mathbf{B}$, because certain aspects of a quadcopter model exist nicely inside this frame. To measure vehicle orientation Euler angles are utilized and to transform between coordinate frames a Z-X-Y rotation matrix is used.

$$R_E^B = C_z(\phi)^T C_x(\phi)^T C_y(\psi)^T = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \quad (6-1)$$

Equation (6-2) shows the result of the multiplication of three Euler angle parameterized direction cosine matrices (DCMs), where "c" and "s" represent a shorthand notation for "co-sine" and "sine", respectively. The matrix R_E^B maps coordinates from the EFF to the BFF. Since the matrix is a member of special orthogonal group 3 (SO(3)), the transformation from the BFF back to the EFF can be accomplished with $R_B^E = (R_E^B)^{-1} = (R_E^B)^T$. Euler angles can also be used to convert the vehicle's orientation in 3-D to an angular velocity vector.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (6-2)$$

Equation (6-1) shows the conversion from Euler angle rates, $\{\dot{\phi}; \dot{\theta}; \dot{\psi}\}$, to an angular velocity vector, $\{p; q; r\}$ where the Euler angles represent typical data from an on-board inertial measurement unit (IMU). The conversion to angular velocity vector is important in simulation to

show the behavior of the quadcopter over time as well as in any estimators that may require information about the vehicle dynamics.

Further details about the characteristics of a quadcopter can be seen in Figure 6-1. The EFF and resulting BFF are defined using the East-North-Up (ENU) convention. Of the three dimensions, the x-axis is aligned with the quadcopter arm holding motor 1. Additionally, motor forces are defined. The thrust forces are shown as, $F_i \quad \forall i \in \{1, 2, 3, 4\}$ and the motor torquing forces, shown by the directional circles, are now defined as $B_i \quad \forall i \in \{1, 2, 3, 4\}$. The vehicle arm length is l , and downward force due to gravity is denoted as mg . Given these definitions and taking from both [55] and [76] the vehicle dynamics are defined as:

$$\dot{\xi} = \nu \quad (6-3)$$

$$m\dot{\nu} = mg\vec{e}_3 + RF \quad (6-4)$$

$$\dot{R} = R\Omega_{\times} \quad (6-5)$$

$$I\dot{\Omega} = -\Omega \times I\Omega + \tau \quad (6-6)$$

The first two equations, (6-3) and (6-4), describe the evolution of the linear dynamics and are expressed in the EFF. ν is the linear velocity in $\mathbb{R}^3$ ($\{\dot{x}, \dot{y}, \dot{z}\}$). Equation (6-4) is the expression of Newton's second law where R is the rotation matrix defined in Equation (6-2) and F is a vector of the sum of thrust forces expressed for each axis in the body frame such that $F = \{0, 0, \sum_{i=1}^4 F_i\}$. The second two equations, (6-5) and (6-6), describe the evolution of the angular dynamics. Equation (6-5) is expressed in the EFF and Equation (6-6) is in the BFF. $\dot{R}$ describes how the rotation of the quadcopter body in the EFF changes over time, where $\Omega_{\times}$ is a skew symmetric matrix (square matrix where the transpose equals the negative) of the vehicle's angular velocity in $\mathbb{R}^3$ ($\{\dot{p}, \dot{q}, \dot{r}\}$) taken in the BFF. A skew symmetric matrix in $\mathbb{R}^3$ describes the cross product operation such that $a \times b = [a]_{\times} b$. Equation (6-6) shows Newton's second law for rotation which relates angular acceleration to net external torque. Here, I is the 3x3 moment of inertia matrix which is taken as diagonal for simplicity. The diagonal of the matrix I describes principle moments of inertia, eg: I_{xx}, I_{yy}, I_{zz} . Since the quadcopter is taken as symmetric about its center of mass the off-diagonal components are assumed to be negligible. If the off-diagonal components were not negligible, then this would lead to coupling and non-linear behavior that would need to be compensated for in control. Lastly, in Equation (6-6) τ is a 3x1 vector of torques acting on the components of the vehicle's body frame.

Ultimately the equations of motion need to be mapped to relevant motor rates. Seeing as a quadrotor vehicle is underactuated with six degrees of freedom (DOF) in its configuration space and only four control inputs, understanding the tradeoffs leading to its non-holonomic behavior is important. Equation (6-3) contains the three dimensional force vector F . The third dimension representing the z-axis in the BFF can be related to the speed of the motors.

$$F_i = C_T \rho A_{r_i} r_i^2 w_i^2 \quad (6-7)$$

One way to do this is with the non-linear relationship shown in Equation (6-7). Here C_T is a thrust coefficient that depends on rotor geometry, ρ is the density of air, A_{r_i} is the disk area created by a spinning propeller, r_i^2 is the radius of the propeller, and w_i^2 is the angular

velocity of the motor. Fortunately most of the terms in Equation (6-7) are specific to the propeller used, and in practice different propeller types are rarely used on the same motor-vehicle combination. This allows Equation (6-7) to be simplified using a lumped parameter model.

$$F_i = k_F w_i^2 \quad (6-8)$$

As long as the same motor-vehicle combination is used, Equation (6-8) can accurately describe the relationship between rotor speed and resulting thrust force. The same sort of simplification can be used to describe the non-linear mapping from motor angular velocity to torque.

$$M_i = k_M w_i^2 \quad (6-9)$$

Equation (6-9) uses the same methodology to relate motor angular velocity to torquing force. These two relationships, thrust and torque, allow the quadcopter to position and orient itself in $\mathbb{R}^3$. By observing Figure 6-1 pitching moments about the y-axis are generated by opposite perturbations applied to the rates of motors 1 and 3. Rolling moments about the x-axis are similarly created using opposite perturbations applied to the rates of motors 2 and 4. Lastly, the yawing moments about the z-axis require unbalanced torque forces by perturbing either motors 1 and 3 or 2 and 4 together. What contributes to the non-holonomic behavior here is the coupled effects present such that when the vehicle yaws the opposite set of motors must spin down to maintain altitude. Additionally, the pitching and rolling moments will cause a net linear velocity along the axis perpendicular to the axis of rotation. The magnitude of these coupling effects is tied to the rate of the vehicle dynamics, and is explored in simulation later.

Given a brief description of the coupling effects in a quadcopter, the mapping from motor angular velocity to forces acting on the vehicle can be shown as a matrix.

$$\begin{bmatrix} T_\Sigma \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \Gamma \omega^2 = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & lk_F & 0 & -lk_F \\ -lk_F & 0 & lk_F & 0 \\ -k_M & k_M & -k_M & k_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (6-10)$$

Equation (6-10) is a compact way to express the relationships discussed above. One benefit of this representation is that the matrix Γ is invariant and has to be inverted only once to yield desired motor rates. It may seem that there are still four degrees of freedom, even after inverting Γ , however a common desired state for a quadcopter is hover. The hover state can be described as zero angular velocity about the center of mass and a sum of thrust forces equal to gravitational acceleration. This helps to constrain Equation (6-10) and highlights the independent variables of interest as: arm length l , thrust coefficient k_F , torque coefficient k_M , moment of inertia matrix $\mathbf{I}$, and vehicle mass m . The proper fitting of these parameters is discussed in the next section.

With the vehicle dynamics defined above, it is worthwhile briefly discussing potential additions to the model. One non-linearity that has been ignored is the rotor flapping effect. This behavior affects the propellers and arises at high speeds because the propellers are flexible.

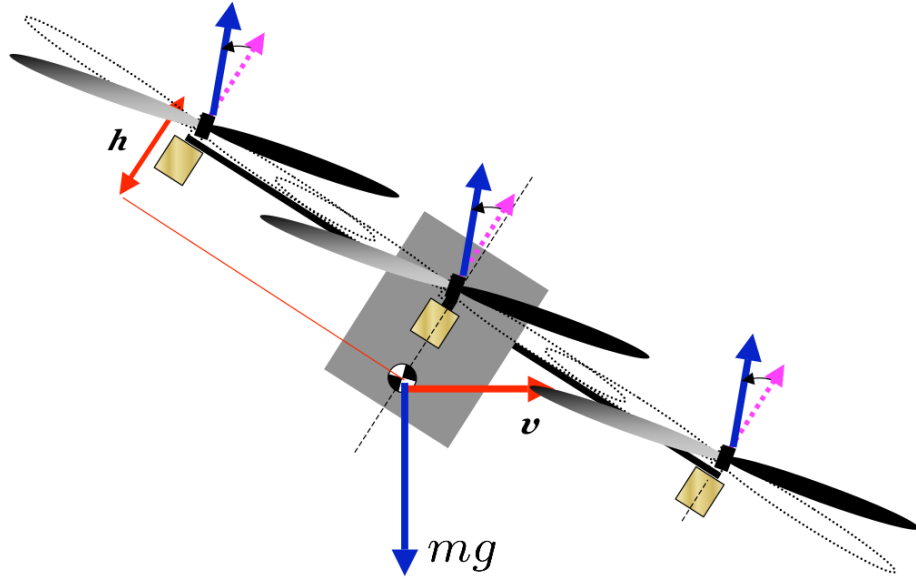


Figure 6-2: Graphic detailing the thrust force offset caused by the rotor flapping effect [11].

This effect is detailed in helicopter research and briefly in some foundational quadcopter work [11, 76]. The effect is visualized below as:

Figure 6-2 shows that there is a torquing force on the propeller blades that causes the lift force to shift back toward the neutral hover position causing a longitudinal moment on the vehicle. This is caused by a higher velocity at the tip of the leading edge of the propeller which generates more lift than the trailing edge. Following the variable notation in Figure 6-2 one way to model the total resulting moment is:

$$M_{b,long} = Th \sin(A_{1s}) \quad (6-11)$$

$$M_{b,s} = k_{\beta} A_{1s} \quad (6-12)$$

$$M_{b,total} = M_{b,long} + M_{b,s} \quad (6-13)$$

By observing the equations above it is clear that the rotor flapping effect impacts the orientation dynamics because of the resulting longitudinal moment. The magnitude of these moments are tied to the thrust force and size of the motors through variables T and h in Equation (6-11). At higher speeds and with larger motors this effect can cause significant disturbances. The total effect also relies on the propeller flexibility k_{β} and the perpendicular offset of the motor caused by the rotor flapping A_{1s} . There are few different ways to model this effect mainly differing in their levels of complexity. However, to test the network-decentralized algorithm in this thesis the quadcopters do not need to operate at velocities over approximately 2.5 m/s and do not require the aforementioned performance improvement. Remaining

within a hover envelope is adequate and this approach will be discussed more in the control section.

Another potential model addition is the ground effect phenomenon. This effect describes the increase in the lift-to-drag ratio of a vehicle when close to the ground. Essentially, for a given thrust force the vehicle will experience more lift when it is close to the ground as opposed to when it is in a free stream environment. Some limited research has been performed modeling this effect for quadcopters [81]. The main model used is shown below.

$$\frac{T}{T_\infty} = \frac{1}{1 - \left(\frac{R}{4z}\right)^2} \quad (6-14)$$

In Equation (6-14), T is the actual thrust the quadcopter produces at a specific rotor setting and T_∞ is the thrust produced by the quadcopter at that same rotor setting but in a free-stream environment. Under ideal operating conditions the ratio is equal to 1, however when ground effect is present the ratio will increase because more thrust will be produced under the same conditions. This is dependent on the rotor radius R and the rotor height above the ground z . Helicopter research points to $\frac{z}{R} > 2$ as the threshold for the influence of ground effect. However, some quadcopter research suggests the ratio may be $\frac{z}{R} > 5$ [81]. Regardless, for the 5.5 cm propellers suggested in Chapter 5 taken with the worst case shown in literature the ground effect disturbance is only present within 27.5 cm of the ground and would impact the vehicle thrust by 5% at approximately 20 cm. A takeoff routine with a sufficiently fast control loop should dominate this disturbance therefore it is not included in the model.

With the rigid body dynamical model presented above, control of the vehicle can now be discussed.

6-1-2 Control

The model for a quadcopter shown in Equations (6-3) - (6-6) is inherently non-linear. This is due to the rotation matrix present in Equations (6-4) and (6-5) as well as the relationship between BFF Euler angles and the vehicle's angular velocity vector. The Euler angle parameterization relies on sines and cosines which can be handled by a more comprehensive non-linear controller or through linearization. It so happens that good results can be achieved through linearization about the hover point of the quadcopter. This is where the roll and pitch are measured at 0 degrees with a variable yaw, and thrust is provided to cancel out the effects of gravity.

Figure 6-3 shows the linearized closed-loop control for a quadcopter. This is done in a way similar to backstepping control presented in Chapter 2, where the position controller is an outer PID loop that sets a virtual orientation that the attitude controller tracks via PD control. Deviations from this linearization point are used to orient and position the vehicle. This is shown by the variables with the Δ prefix. Taking the motor model from Equation (6-8) the motor rate for a single motor at hover is found to be:

$$\omega_{i,h} = \sqrt{\frac{mg}{4k_F}} \quad (6-15)$$

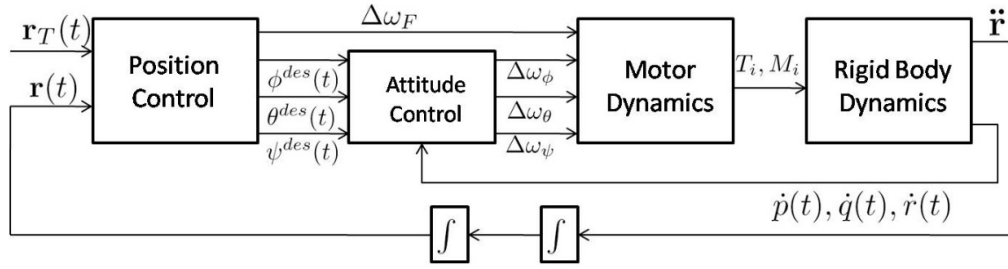


Figure 6-3: Nested PID control loop for quadcopter vehicle [12].

With the linear force to cancel gravity shown in Equation (6-15), Equation (6-6) representing orientation can be combined with Equation (6-10) to form three equations:

$$I_{xx}\dot{p} = lk_F(\omega_2^2 - \omega_4^2) - qr(I_{zz} - I_{yy}) \quad (6-16)$$

$$I_{yy}\dot{q} = lk_F(\omega_3^2 - \omega_1^2) - pr(I_{xx} - I_{zz}) \quad (6-17)$$

$$I_{zz}\dot{r} = k_M(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \quad (6-18)$$

The model is linearized about its hover point making roll and pitch components approximately zero. The roll, pitch, and yaw rates are also assumed to be zero. Since the vehicle is assumed symmetric the product of inertia components is assumed to be zero. The linearization required here is to deal with the sine and cosine terms and a small angle approximation is used. Observation of Figure 6-1 and the previous discussion of motor thrust and torque forces, allows the effects of each motor on the vehicle roll, pitch, and yaw to be defined compactly.

$$\begin{bmatrix} w_1^{des} \\ w_2^{des} \\ w_3^{des} \\ w_4^{des} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} w_h + \Delta\omega_F \\ \Delta\omega_\phi \\ \Delta\omega_\theta \\ \Delta\omega_\psi \end{bmatrix} \quad (6-19)$$

Equation (6-19) shows the effects of each individual motor on the vehicle orientation when the vehicle is assumed to be operating inside its hover envelope. Taking motor 1 from Figure 6-1 and Equation (6-19) it is shown that the motor rate creates a negative effect on the pitch angle which the opposite motor, motor 3, cancels out. However, both motors accumulate a yawing moment which is only canceled by motors 2 and 4. With this compact representation, Equations (6-16) - (6-18) are linearized to produce a new set of rotational equations:

$$\dot{p}^{des} = \frac{4k_F l \omega_h}{I_{xx}} \Delta\omega_\phi \quad (6-20)$$

$$\dot{q}^{des} = \frac{4k_F l \omega_h}{I_{yy}} \Delta\omega_\theta \quad (6-21)$$

$$\dot{r}^{des} = \frac{4k_M l \omega_h}{I_{zz}} \Delta\omega_\psi \quad (6-22)$$

Equations (6-20) - (6-22) show the linearized orientation dynamics where deviations from the hover envelope can now be set using a PD control law.

$$\Delta\omega_\phi = k_{p,\phi}(\phi^{des} - \phi) + k_{d,\phi}(p^{des} - p) \quad (6-23)$$

$$\Delta\omega_\theta = k_{p,\theta}(\theta^{des} - \theta) + k_{d,\theta}(q^{des} - q) \quad (6-24)$$

$$\Delta\omega_\psi = k_{p,\psi}(\psi^{des} - \psi) + k_{d,\psi}(r^{des} - r) \quad (6-25)$$

In combination with Equation (6-19) the desired motor speeds can be found. Referring back to the control architecture in Figure 6-3 it can be seen that the attitude controller described above is fed virtual orientation angles to track by the position controller. The position controller can be derived similarly to the attitude controller using Equation (6-4).

$$\ddot{r}_x^{des} = g(\theta^{des}\cos(\psi_T) + \phi^{des}\sin(\psi_T)) \quad (6-26)$$

$$\ddot{r}_y^{des} = g(\theta^{des}\sin(\psi_T) - \phi^{des}\cos(\psi_T)) \quad (6-27)$$

$$\ddot{r}_z^{des} = \frac{8k_F\omega_h}{m}\Delta\omega_F \quad (6-28)$$

In Equations (6-26) - (6-28) the rotation matrix in (6-4) is linearized based on a small angle approximation. The linear acceleration $\dot{v}$ is taken as $\ddot{r}$ to be more consistent with the previous single-agent definition given in 2. These equations can be solved for ϕ^{des} , θ^{des} , and $\Delta\omega_F^{des}$ and $\ddot{r}_i^{des}$ can be set via PID control similar to the methodology shown in the attitude controller. Some work has been done to set $\ddot{r}^{des}$ via 3-D trajectory tracking with a feed-forward acceleration term [12]. However, the network-decentralized algorithm in this thesis sets either the vehicle velocities or accelerations, so additional trajectory tracking methods are not explored here.

6-2 System Identification

The previous section laid out one approach for the modeling and control of a quadcopter. Based on the dynamics shown in Equations (6-3) - (6-6) the parameters required for a first principles system identification can be seen in Table 6-1.

Parameter	Description
m	Vehicle mass at CoG
$\mathbf{I}$	Moment of inertia matrix
k_F	Lumped parameter thrust constant
k_M	Lumped parameter torque constant

Table 6-1: List of parameters for first principles system identification

Table 6-1 shows the parameters that must be found to describe the motion of a specific quadcopter through free space while acting within its hover envelope. This section will utilize the vehicle described in Chapter 5. The mass of the vehicle can be found using a simple kitchen scale. For the QAV-R2 this is 482 grams.

The moment of inertia matrix is found using the computer aided design (CAD) software Fusion 360 [82]. Each component of the vehicle is weighed individually and assumed to be uniform. The motors and electronics are taken as static point masses with the electronics assumed to be symmetric about the CoG of the base plate. The respective component masses are set in software. The model used is shown as:

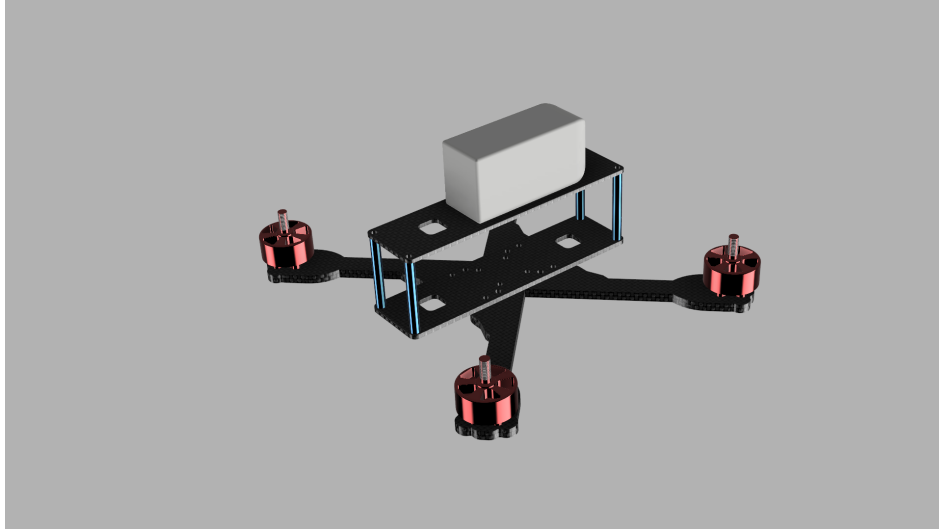


Figure 6-4: CAD design of the QAV-R2 quadcopter

Using the model shown in Figure 6-4 resulting properties are generated in software. The moment of inertia matrix is one of these. Taking the off diagonal components as negligible and $I_{xx} \approx I_{yy}$, this yields:

$$\mathbf{I} = \begin{bmatrix} 1.544 & 0 & 0 \\ 0 & 1.544 & 0 \\ 0 & 0 & 2.332 \end{bmatrix} gm^2 \quad (6-29)$$

The symmetric nature of the vehicle yields the similar values $I_{xx} = I_{yy}$. It can also be noted that the rotation about the z-axis requires more force for the same angular acceleration. The zeros in the upper and lower triangle are due the axes of rotation coinciding with the principle axes.

The thrust and torque parameters are dependent on the motor. As mentioned in Chapter 5 the vehicle motor is a 2206-2300Kv brushless direct current (DC) design. This interfaces with an electronic speed controller (ESC) which allows digital logic to generate a three phase continuous voltage signal which drives the motor. Using a testbench, torque and thrust data can be gathered along with current consumption, power requirements, rotations per minute (RPM), and respective ESC commands that relate the information.

Figure 6-5 shows a RCBenchmark 1580 dynamometer with an additional optical RPM sensor. The setup interfaces directly with a laptop computer with the motor powered by an external Lithium polymer (LiPo) battery and a dedicated 5 V DC power supply for the optical sensor. The optical RPM sensor relies on retroreflective tape applied to the motor's rotor to determine a successful rotation. The mounted motor and prop are the same as described in Chapter



Figure 6-5: Motor testbench to collect data for system identification.

5, a 2206-2300KV motor with a 5x4.3x3V1S polycarbonate propeller and the ESC interface between the motor and computer is rated for 30 Amps, but 20 A should work because peak current consumption is measured at approximately 16 A.

Using a ramp test the ESC input was varied from 1000 μs to 1800 μs over 30 seconds. Data was sampled at 40Hz and every 20 samples were averaged using a moving window. From this information Equations (6-8) and (6-9) were solved for their respective thrust and torque constants. This result is displayed in Figure 6-6.

Figure 6-6 shows the raw data in blue and an interpolated smoothing of the data in red. Measurement noise was expected and is seen in the data. Also note the scale of the axis for both the thrust and torque constants is small. The constants themselves are used to formulate controller gains, of which will need to be tuned further once implemented, so taking a simple average of this data is adequate. Doing this, the thrust constant is found to be $1.1e-08 \frac{N}{RPM^2}$

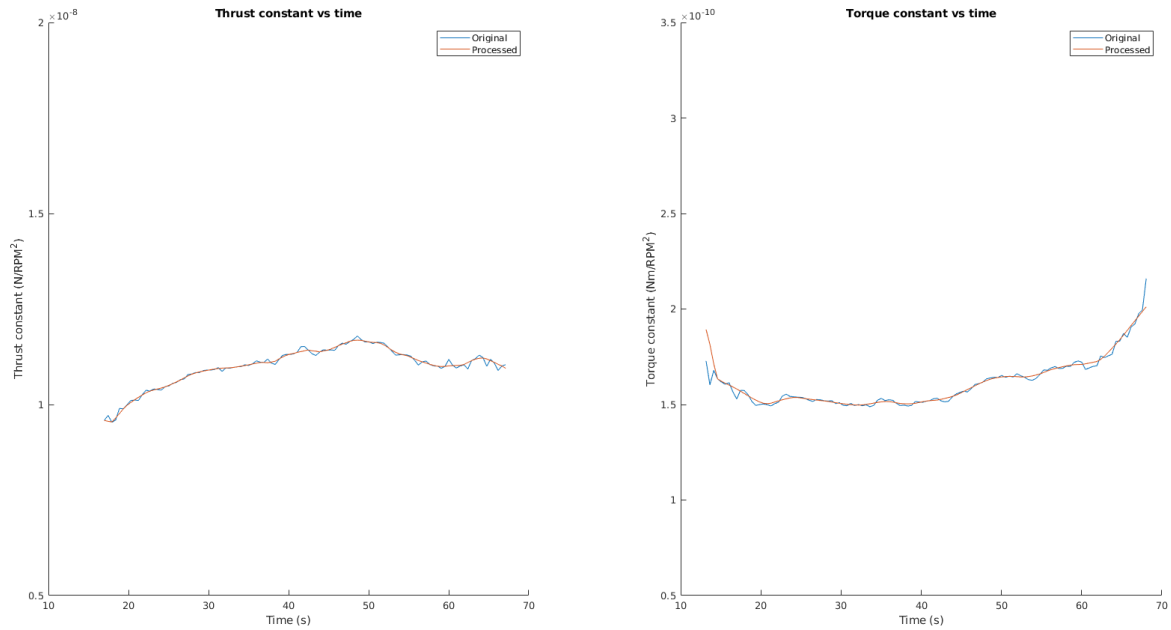


Figure 6-6: Thrust and torque constants during a ramp test of a 2206-2300KV motor.

and the torque constant is $1.6 \times 10^{-10} \frac{N}{RPM^2}$. Additional information can also be gained from the test data. A polynomial line of best fit is used to describe the relationship between motor RPM and both thrust and torque forces. This can be seen in Figure 6-7.

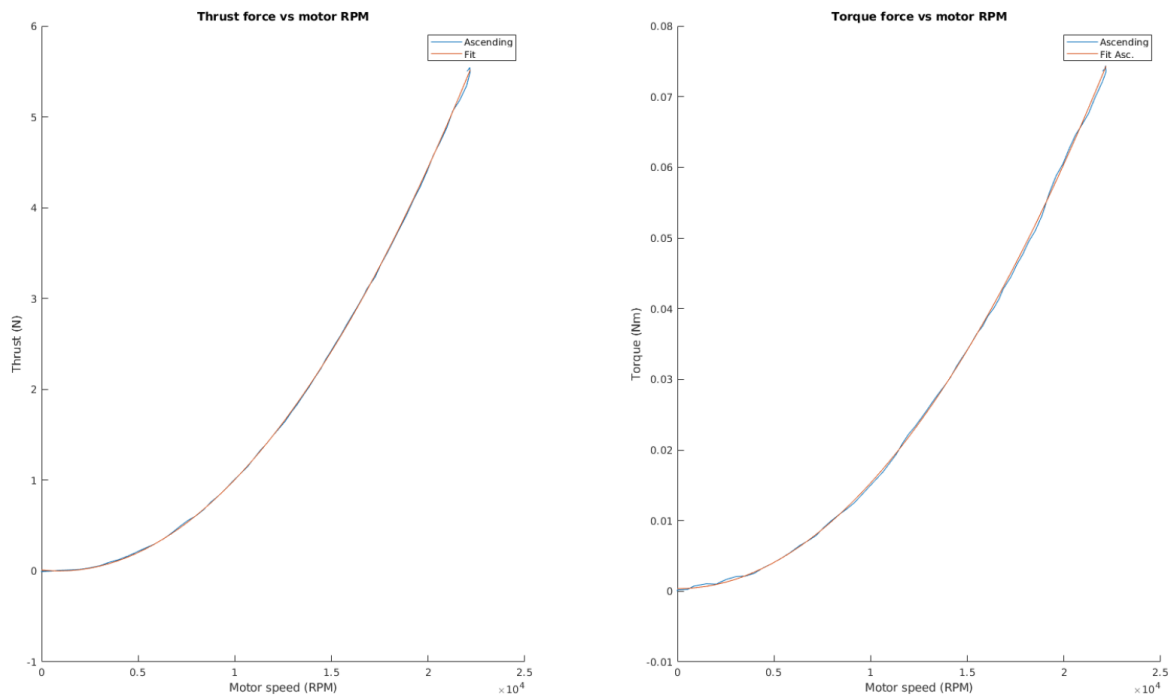


Figure 6-7: Thrust and torque forces vs motor speeds with a second-order polynomial of best fit.

A mapping between motor rates and thrust and torque is important for simulation and estimation. The polynomial approximations in Figure 6-7 are second order with the parameters rounded to the fourth decimal place to avoid floating point precision overfitting. This is coupled with a mapping from ESC commands to motor rates to provide a complete picture of how motor inputs affect the vehicle dynamics.

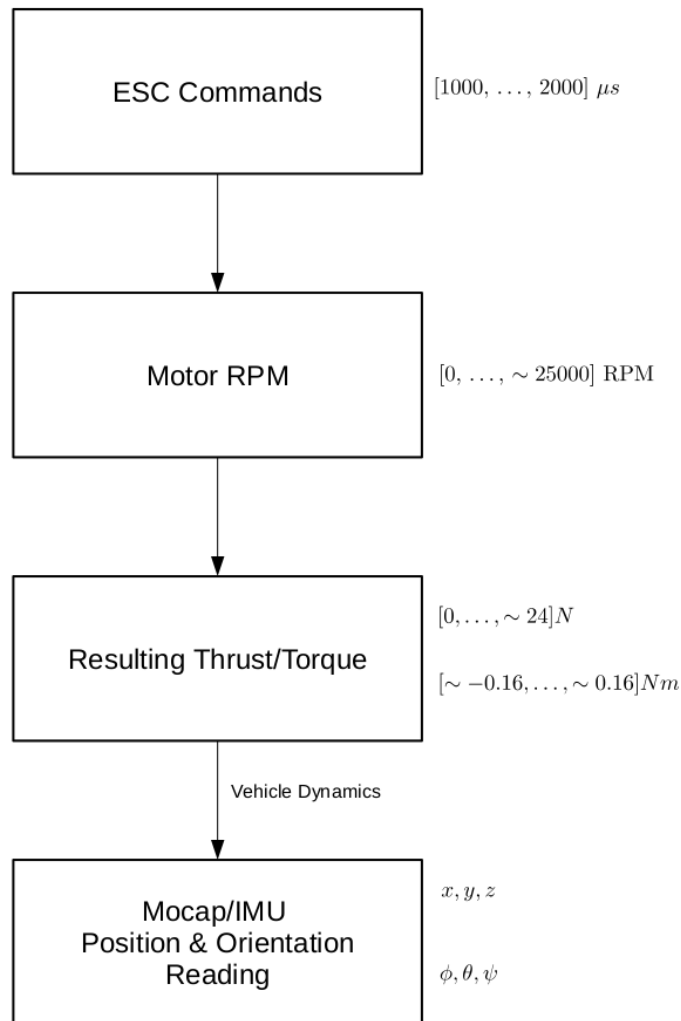


Figure 6-8: Information flow from the ESC input to the final position/orientation.

In Figure 6-8 the need for a mapping from ESC to motor rate and from motor rate to thrust and torque becomes clear. Without this mapping there is no way to measure this information online. It is important to note that the mapping from ESC input to resulting motor rate varies with time. As the battery voltage drops over the course of a flight a higher ESC input will be needed to maintain the same motor rate.

The ramp test conducted using the motor also yielded power and current consumption data. This is used along with thrust and motor rate data to generate a size, weight, and power (SWAP) plot.

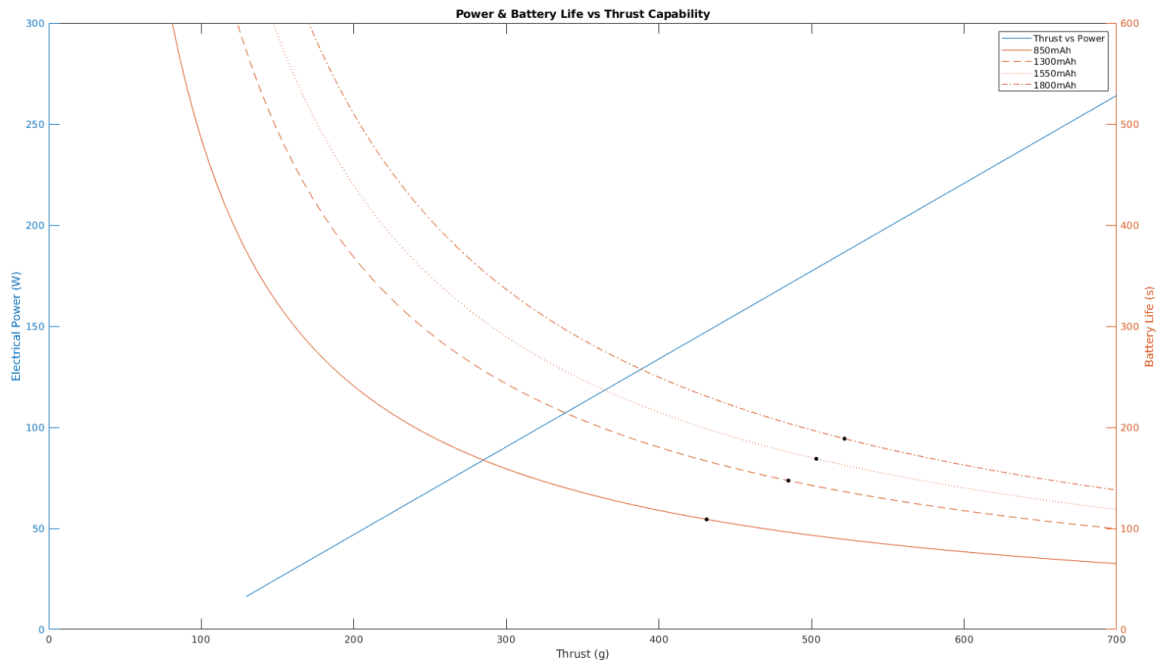


Figure 6-9: A SWAP plot denoting performance for the 2206-2300KV motor.

Figure 6-9 shows theoretical performance for the motor mounted on the QAV-R2 platform as detailed in Chapter 5. The red lines show theoretical battery life for different sizes of LiPo batteries. Ultimately the 1300 mAh setup was chosen, which is predicted to have 147 seconds of flight time.

With the quadcopter dynamics detailed, hover envelope control defined, and relevant system parameters identified the controller gains still need to be set. With a nested position-attitude controller in place there are five tunable gains. Since a poorly chosen set of parameters can result in a rapid unplanned dissassembly of the vehicle a simulation environment was developed using MATLAB for model validation before physical testing. This is described in more detail in Chapter 7.

Quadcopter – Testing

With holonomic simulations of the network-decentralized algorithm presented in Chapters 3 and 4 and the development of a new lab environment with new vehicle selection in Chapter 5, this chapter presents the performance of a physical vehicle. As stated in Chapter 5 it is recommended to implement the QAV-R2 platform, however the Delft Center for Systems and Control (DCSC) was unable to acquire this platform so the physical tests presented here are with a single Crazyflie 2.1. The tests with a single physical vehicle are done to validate the parameters that were experimentally fitted in Chapter 6 and to compare these results to the holonomic simulations in 3-D from Chapter 4. This indicates that the holonomic agents in 3-D have similar performance to a quadcopter controlled about its hover envelope. Lastly, a high fidelity quadcopter simulation in Gazebo is used to test the network-decentralized algorithm and this performance is compared to the holonomic agents in Chapter 4. These high fidelity tests indicate that the algorithm has decreased performance to the point of collisions in a real-time setting, where in the same case but with holonomic agents the results are successful.

7-1 Single Agent Performance

7-1-1 Takeoff and Hover

With the virtualized lab environment as described in Chapter 5 and a radio mounted remotely through VMWorkstation 15 [63], a hover test is performed to determine a baseline accuracy of the setup.

Figure 7-1 shows the Crazyflie in the lab environment takeoff from a stand, hover at 1.0 meters, and subsequently land in the same location. A plot of the vehicle's altitude over the duration of the test is shown in Figure 7-2. It is shown that the vehicle oscillates about the desired altitude in a sinusoid-like fashion with an amplitude of approximately 2 cm and a period of approximately 6 – 7 seconds. The accuracy of the system is summarized in Table 7-1.



Figure 7-1: Crazyflie 2.1 hovering at 1.0 meters.

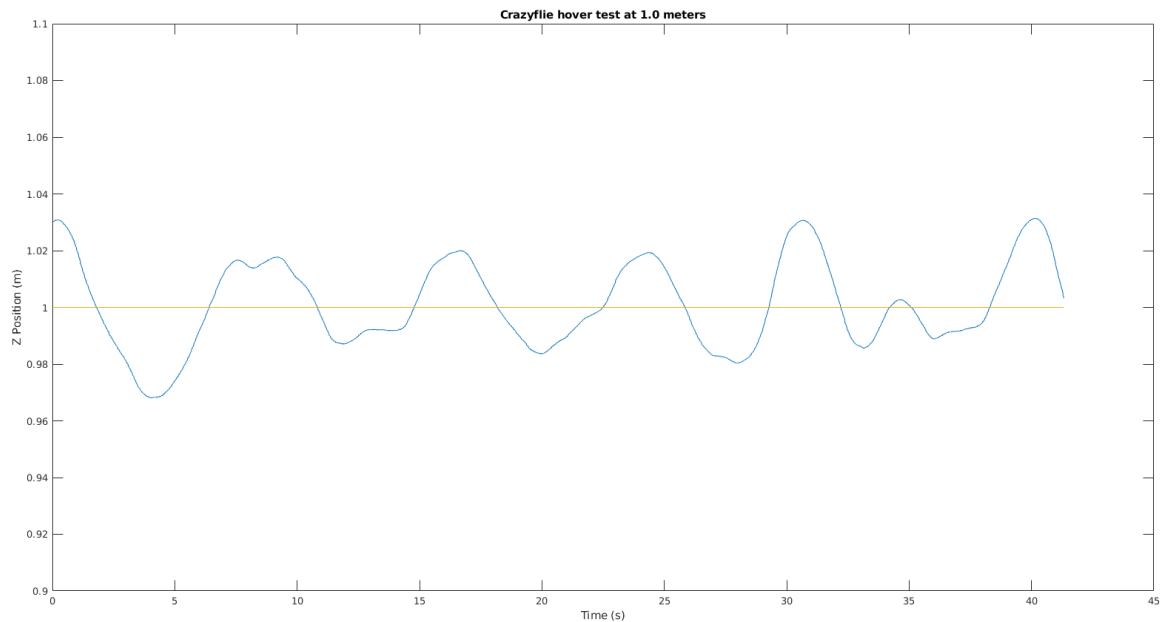


Figure 7-2: Plot of the Crazyflie position over time during the hover test.

	Mean	MSE	Variance
Sampling	99 Hz	—	17.8 Hz
X Pos.	0.0424 m	0.0019 m	6.3894e-05 m
Y Pos.	0.1146 m	0.0132 m	3.9233e-05 m
Z Pos.	1.0011 m	2.4040e-04 m	2.3936e-04 m

Table 7-1: Crazyflie 2.1 hover test in the virtualized lab environment.

In Table 7-1 it is shown that the vehicle has vertical variance of 2.3936×10^{-04} meters and a mean-squared error (MSE) of 2.4040×10^{-04} meters at hover. This indicates a reliable

positioning solution. The "Sampling" data field shows the rate at which the data is logged via Robot Operating System (ROS), not the packet rate from the radio to the vehicle. The packet rate from the radio to the vehicle is assumed to remain around 75 Hz as demonstrated in Chapter 5. Note that both the takeoff and landing routines for the vehicle are subject to ground effect forces, as described in Chapter 6. Therefore the takeoff and landing data are not included in this analysis.

7-1-2 Waypointing

In this test the vehicle establishes a hover at 0.35 meters and then flies a square with sides of 0.5 meters. This is done with velocity limiting of 1.0 m/s, as with the holonomic agent as described in Chapter 2.

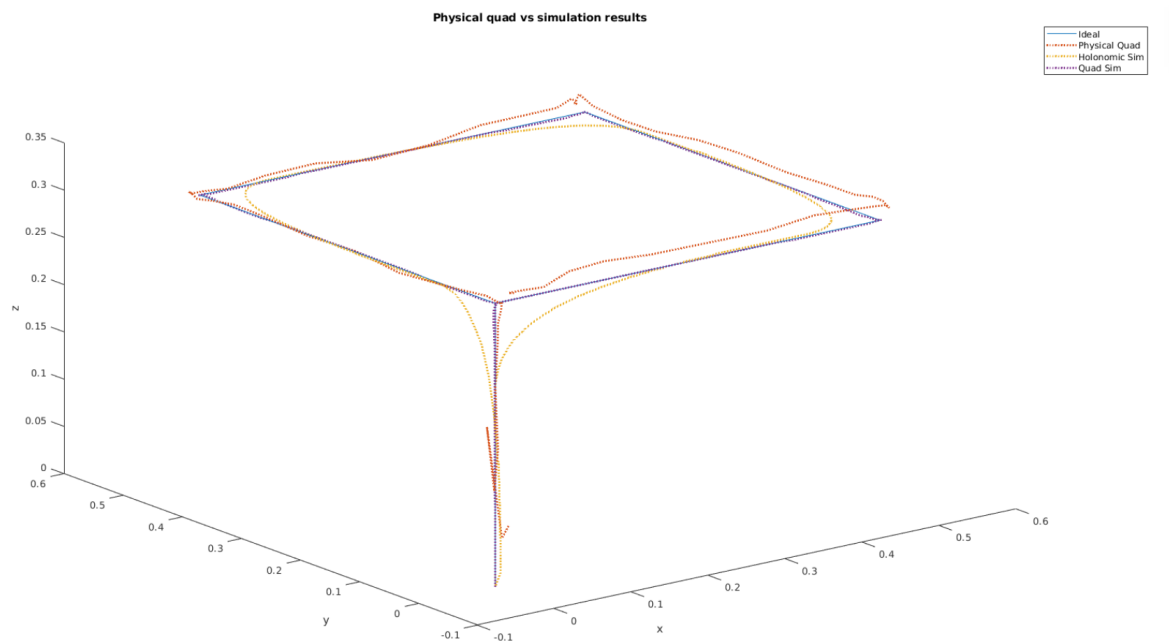


Figure 7-3: Comparison of physical Crazyflie 2.1 and various simulations in a square pattern.

	MSE
Physical	0.0471
Quad Sim	0.0211
Holonomic Sim	0.0910

Table 7-2: Comparison of mean-squared error between simulation and physical results in a simple square test.

Figure 7-3 shows multiple tests laid on top of one another. The "ideal" case is simply the trajectory that every other test is tracking. Then there are the physical, holonomic, and quadcopter simulation cases. The quadcopter simulation here is a MATLAB simulation with parameters fitted from Chapter 6 Table 6-1. A more comprehensive look at the various simulation environments developed for this thesis can be seen in the supporting documentation

also produced for this thesis [46]. The holonomic simulation is taken unaltered from Chapter 4, and the physical test is again with the Crazyflie vehicle. Table 7-2 shows how the physical, quadcopter simulation, and holonomic simulation tests compare to the ideal trajectory. Note that since the holonomic simulation is taken unaltered from Chapter 4 the successful waypoint threshold is unchanged at 0.10 m which leads to a MSE that is artificially high which can be misleading. By reducing the waypointing threshold the holonomic sim could be made to look more accurate, however that is not the focus here. Instead, it is more important to recognize that the physical performance is very similar to that of the quadcopter and holonomic simulation as can be seen visually in Figure 7-3 and numerically in Table 7-2. Further analysis is performed by observing altitude varying performance.

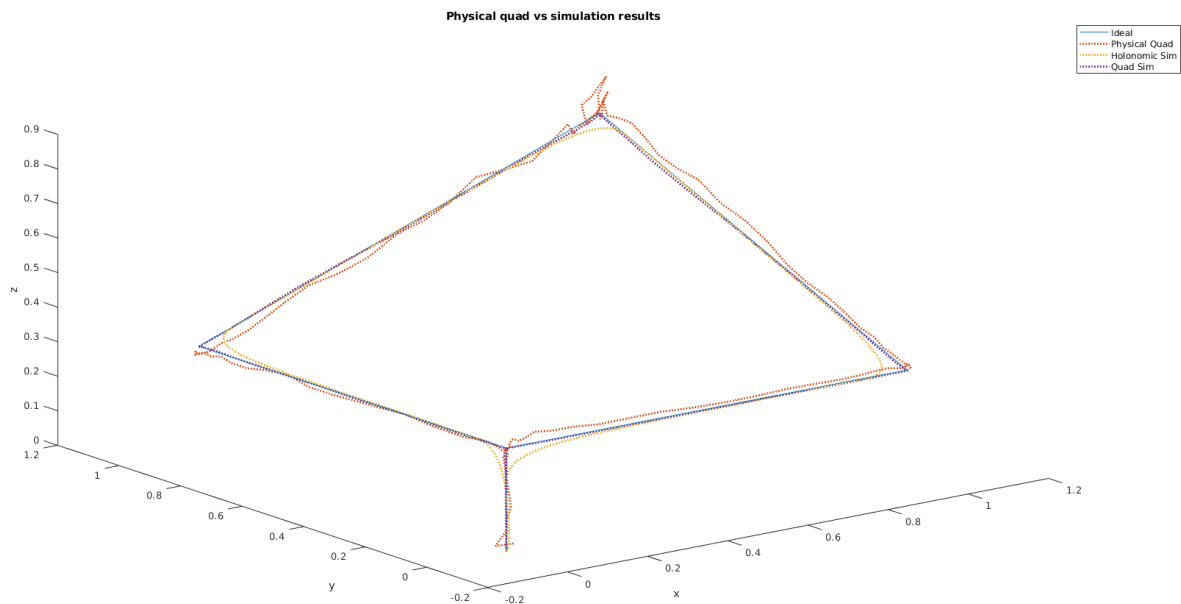


Figure 7-4: Comparison of physical Crazyflie 2.1 and various simulations in an altitude varying square pattern.

	MSE
Physical	0.1362
Quad Sim	0.0884
Holonomic Sim	0.3679

Table 7-3: Comparison of mean-squared error between simulations and physical tests in an altitude varying test.

By varying the altitude additional stress is placed on the physical and simulation based tests. This allows for a more comprehensive look at how the setups differ. It can be seen both visually in Figure 7-4 and Table 7-3 that the MSEs are about 3x what was seen in the altitude invariant square test. However the overall MSE remains quite low and it can be concluded that both the holonomic simulation and quadcopter simulation are good approximations of the behavior of a single quadcopter with a hover-envelope controller. Additionally, the system identification approach detailed in Chapter 6 was used in the quadcopter simulation to tune controller gains. This controller has been shown to yield reliable performance on the physical

vehicle that is similar to that of the simulation, thereby validating the system identification methodology.

7-1-3 Circling

The last of the single agent tests is performed using a circling trajectory. This test is useful because the trajectory that must be tracked is continuously changing across multiple components, mainly in the X and Y dimensions, and the vehicle must also maintain altitude while being continuously perturbed from its hover state.

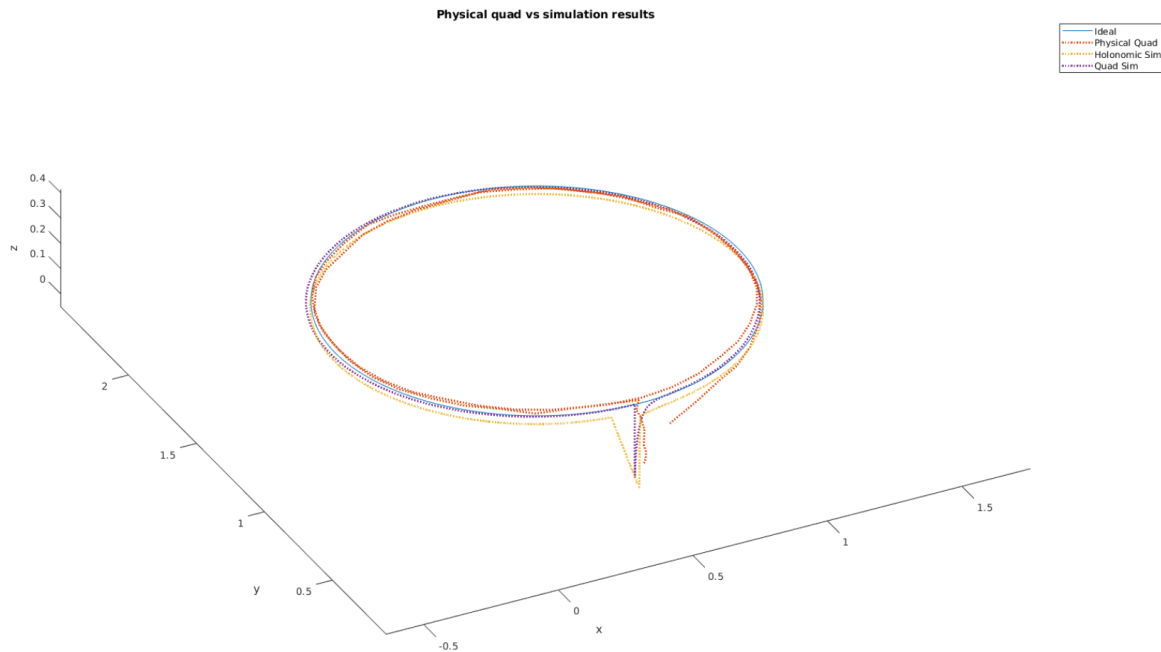


Figure 7-5: Comparison of physical Crazyflie 2.1 and various simulations in a circling pattern.

	MSE
Physical	0.3092
Quad Sim	0.2309
Holonomic Sim	0.1844

Table 7-4: Comparison of mean-squared error between simulations and physical tests in a circling test.

Figure 7-5 and Table 7-4 show the results of the circle trajectory test. The circle that is being tracked is of radius 0.5 m and it can be seen that the MSE is comparable to that of the previous tests with respect to the accuracy of the physical quadcopter. Note that the holonomic sim is more accurate than in the previous tests because there are no waypoints to track. Of the three tests this one demonstrates the highest MSE for the physical quadcopter, however the physical vehicle is still closely related to the simulations. This again indicates that the simulations, both holonomic and quadcopter, are accurate indications of the behavior of a quadcopter controlled about its hover envelope.

7-2 Multi-Agent Collision Avoidance

In the previous section it was shown that a single physical quadcopter closely mimics the behavior of both a more basic quadcopter simulation in MATLAB and that of a holonomic agent. This result was repeated across three different tests, however, this was only measured using a single agent. In the multi-agent real-time scenario there are more considerations that must be taken into account, including but not limited to: variance in the sampling rates; asynchronous sampling rates between agents; variance between vehicle dynamics and the control update; non-linearities in the motor dynamics; and the real-time override of a quadcopter's velocity. It has been shown in literature that these considerations can be mostly ignored in the single agent case [3, 79, 55, 80, 58, 51, 75]. For the multi-agent case, trajectories are usually calculated offline and then each vehicle is given its own trajectory to follow in real-time [54, 51]. The online avoidance case for real-time vehicles relies not only on the ability of each vehicle to track a trajectory, such as in Section 7-1, but also the ability for each agent to quickly deviate from that trajectory, and also recover from that deviation. It is shown in this section that, for the algorithm presented in this thesis, the collision avoidance performance of real-time quadcopters decreases rapidly in more obstacle dense environments.

This section utilizes the Gazebo simulator [73], a more high fidelity simulation environment with a focus on real-time performance. The vehicle implemented in this section is the QAV-R2, not the Crazyflie. It was shown in Chapter 6 the QAV-R2 vehicle parameters were identified, and in the previous section this system identification methodology was verified with the Crazyflie platform. The QAV-R2 platform, as designed in this thesis, is built around a flight controller (FC) running PX4 firmware which is built and run in the Gazebo simulation the same way it would be implemented on a physical FC. The firmware is built with ROS bindings to allow the user to access and control simulation information. A command and control node as well as a swarm management node were developed for the implementation of the algorithm presented in this thesis. These nodes are applicable in both Gazebo and the physical build.

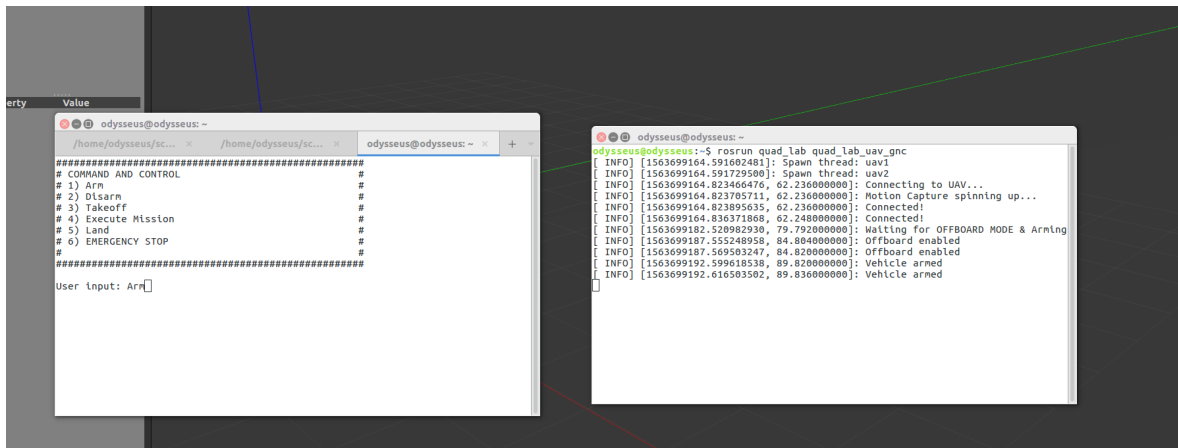


Figure 7-6: Developed user interface with Gazebo using ROS libraries.

Figure 7-6 shows the command and control interface on the left and the swarm management node on the right. The swarm management node spawns a new thread for each vehicle in the simulation and provides feedback as to what commands are received and executed by

the vehicles. Using the Gazebo bindings a mock motion capture environment is implemented to add further realism to the simulation. A detailed look at this simulation environment and how this thesis interfaces with it can be seen in the companion document produced alongside this thesis [46].

7-2-1 Two Agents Collision Avoidance – Planar

In this test two agents are initialized on the same plane and made to naively swap places. During the maneuver the vehicles avoid any detected obstacles.

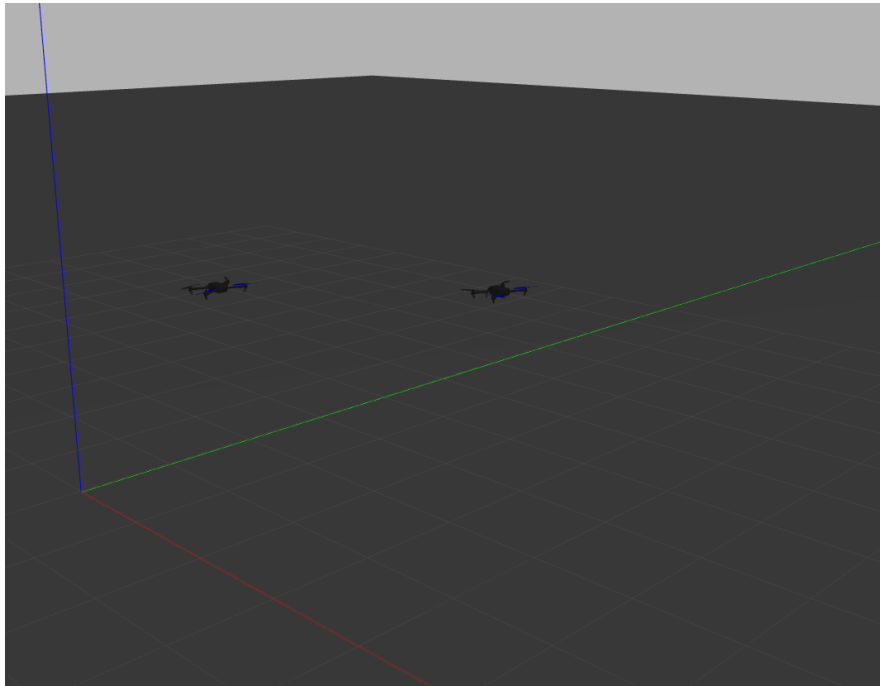


Figure 7-7: Gazebo environment of two vehicles that will avoid on the same plane.

	MSE
Agent 1	0.1755
Agent 2	0.1662

Table 7-5: Holonomic vs quadcopter simulation, 2-agent planar avoidance performance.

Figure 7-7 shows two agents initialized in the Gazebo environment on the same plane. The test results are overlaid with that of an equivalent test using the 3-D holonomic agent case and can be seen in Figure 7-8. The dashed lines indicate the holonomic agents and the solid lines are from the Gazebo simulation vehicles. Note the result of the quadcopter simulation is similar in behavior, but does not track the holonomic results closely. This is in part due to the difficulty of smooth velocity control of quadcopter dynamics, as well as the asynchronous update of the vehicle dynamics and control. The direction of the velocity vector of the quadcopter is defined as "forward" for the purpose of the collision avoidance algorithm and in Figure 7-8 it is shown that the vehicle behavior swaps between avoidance and reference tracking states

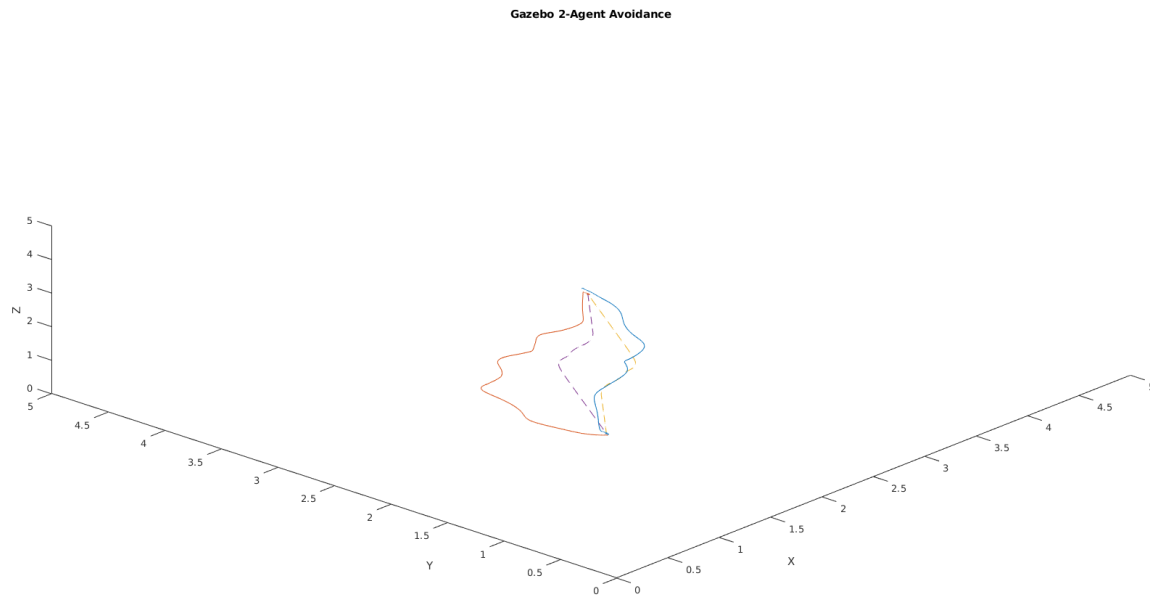


Figure 7-8: Two agents avoiding in real-time on the same plane.

multiple times. The MSE comparison between the quadcopter and holonomic simulation for each vehicle can be seen in Table 7-5. The MSE for each vehicle is multiple times larger than that seen in Table 7-2 where the vehicle traveled a similar distance. This indicates a significant divergence between the behavior of the quadcopter vehicle and holonomic agent, as can be easily visualized in Figure 7-8.

7-2-2 Two Agents Collision Avoidance – Altitude Varying

To place further pressure on the real-time quadcopter simulation the test from Figure 7-8 is repeated, but with the vehicles at different altitudes. The initialization distance between the agents is kept constant to ensure the focus is placed on the vehicles' ability to generate avoidance velocities in three dimensions.

	MSE
Agent 1	0.2353
Agent 2	0.2089

Table 7-6: Holonomic vs quadcopter simulation, 2-agent altitude varying performance.

In the altitude varying test two different perspectives can be seen in Figures 7-9 and 7-10. The plot schema is the same as above with the holonomic test as dotted lines and the quadcopter simulation as solid lines. Figure 7-9 shows the vehicles cutting through all three dimensions whereas Figure 7-10 is a projection onto the XY plane. The projection shows similar behavior to that seen in the previous test in Figure 7-8 where the vehicles swap between avoiding and reference tracking states multiple times. The MSE between the holonomic test and the quadcopter simulation can be seen in Table 7-6. As in the planar test above, the MSE here for both quadcopters is much higher indicating a degradation of performance.

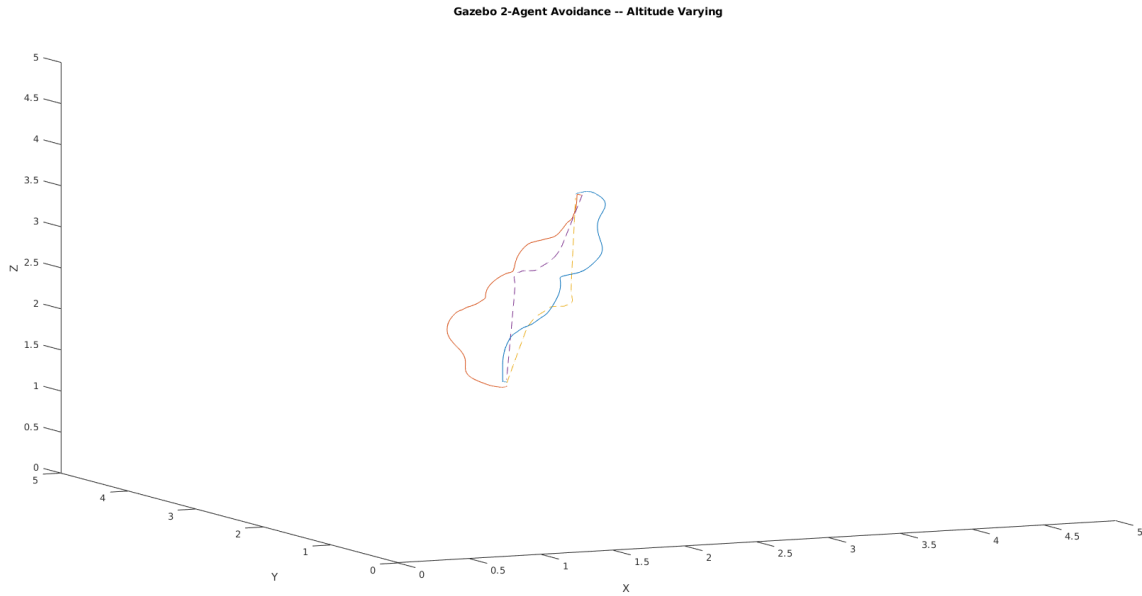


Figure 7-9: Two agents avoiding in real-time in an altitude varying setting.

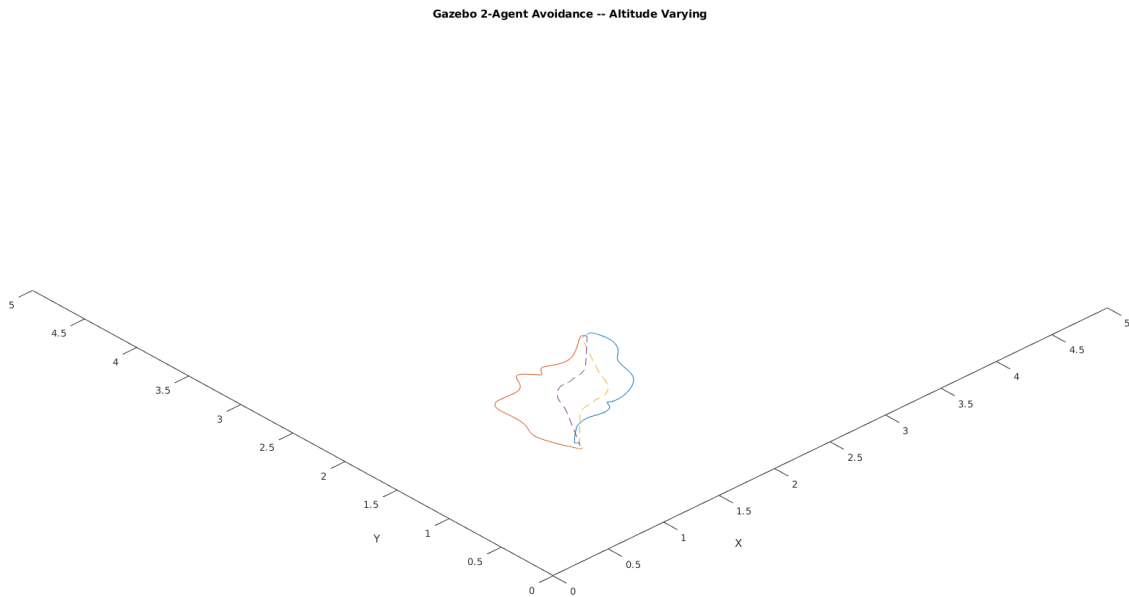


Figure 7-10: Two agents avoiding in real-time in an altitude varying setting, another perspective.

7-2-3 Six Agents Collision Avoidance – Sphere

Similar to Chapter 4 the vehicles can be initialized in a sphere and be made to swap places with other vehicles. This is done to show how much free space exists in 3-D for the vehicles to operate in, even with their non-linear constraints.

Figure 7-11 shows six quadcopters initialized in a sphere of radius 2.0 m. While Figure 7-13 looks to be a constrained environment the vehicles only have to make a few avoidance

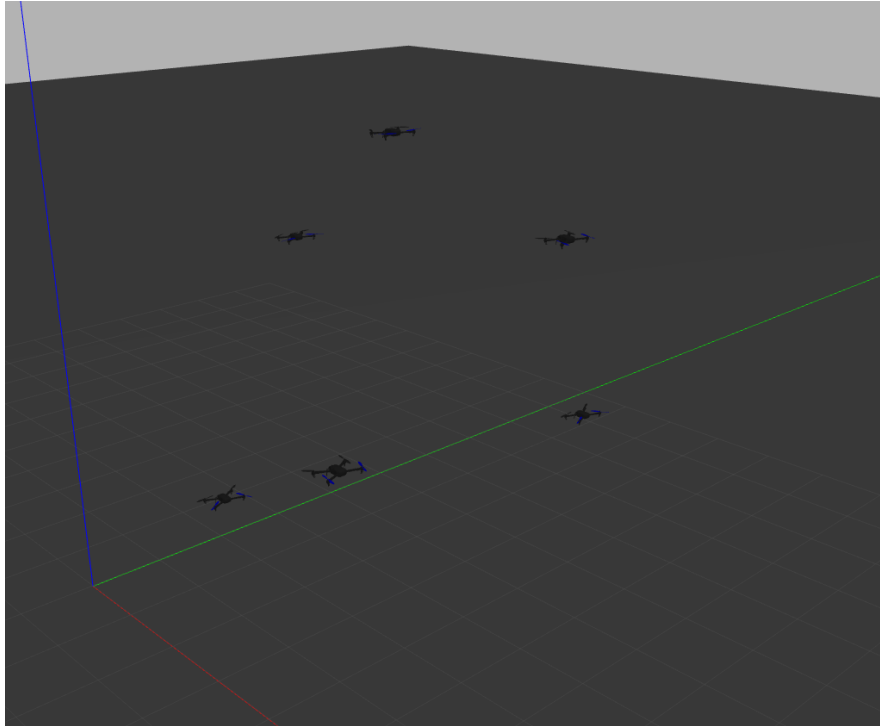


Figure 7-11: Initialization of a 6-agent sphere formation.

6-Agent Avoidance Simulation

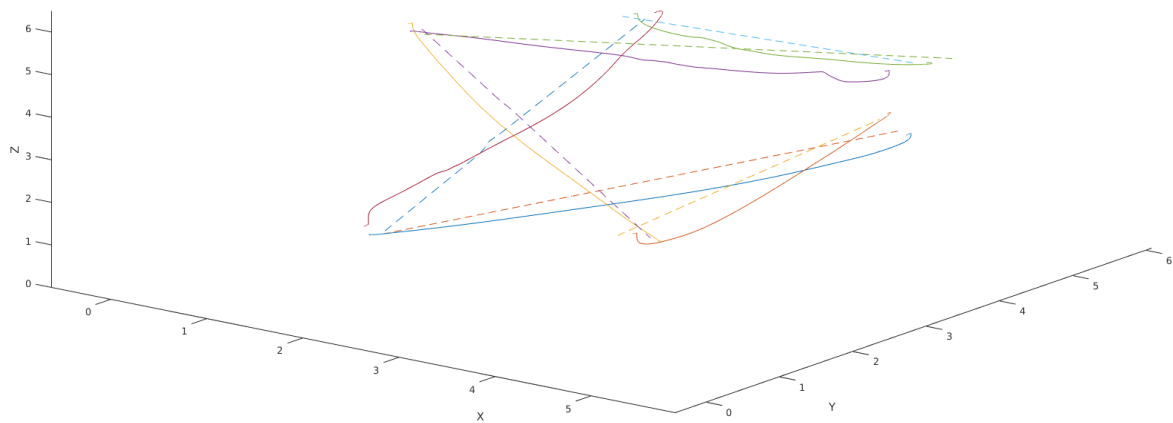


Figure 7-12: Holonomic vs quadcopter simulation, 6-agent sphere avoidance performance.

adjustments to achieve their reference positions. This is shown in Figure 7-12 where the plot characteristics are the same as stated before. Despite only needing a few adjustments the vehicles still exhibit higher MSEs than the single agent trajectory tracking shown in the previous section, however the MSE varies from vehicle to vehicle depending of the amount of

	MSE
Agent 1	0.8713
Agent 2	0.1696
Agent 3	0.3520
Agent 4	0.3409
Agent 5	0.2196
Agent 6	0.0321

Table 7-7: Difference between the holonomic agent and quadcopter simulation sphere avoidance results.

avoidance that was necessary. The main result from this test is to demonstrate how much free space is available to the vehicles in a formation that otherwise looks to be obstacle dense. To further stress the vehicles more quadcopters would need to be initialized in Gazebo, this does not scale well and is computationally expensive, or the vehicles must be constrained further.

7-2-4 Six Agents Collision Avoidance – Planar

As seen above, one way to constrain the vehicles further is to initialize them on the same plane and have them switch places. The vehicles can still avoid in 3-D, but for the most part will remain near the plane they are initialized on resulting in more potential collisions that must be avoided.

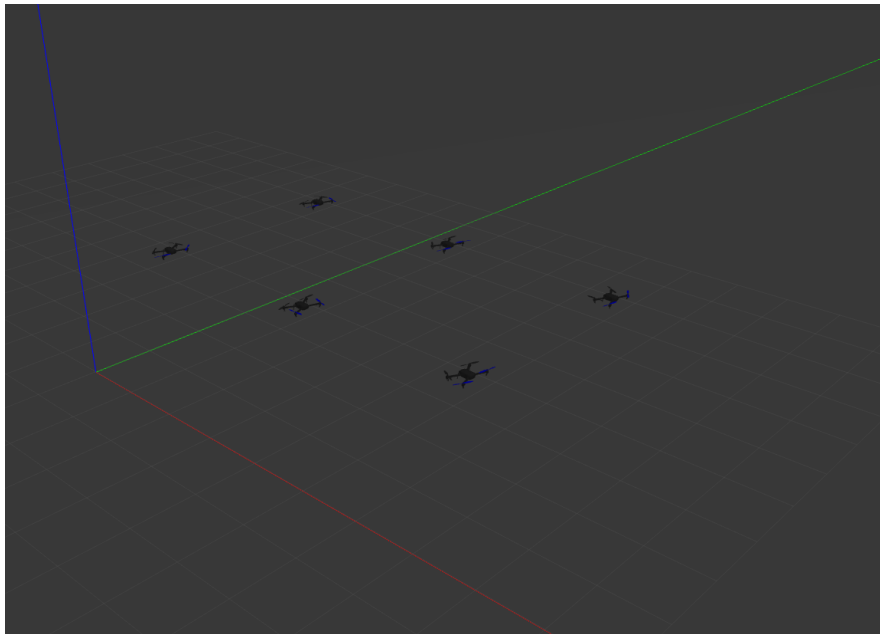


Figure 7-13: Initialization of a 6-agent formation on a single plane.

Figure 7-13 shows six agents initialized at the same altitude. The initialization points of the agents are circularly shifted which becomes the new setpoints. In the holonomic case the agents are capable of consistently successfully avoiding collision. However, in the real-time

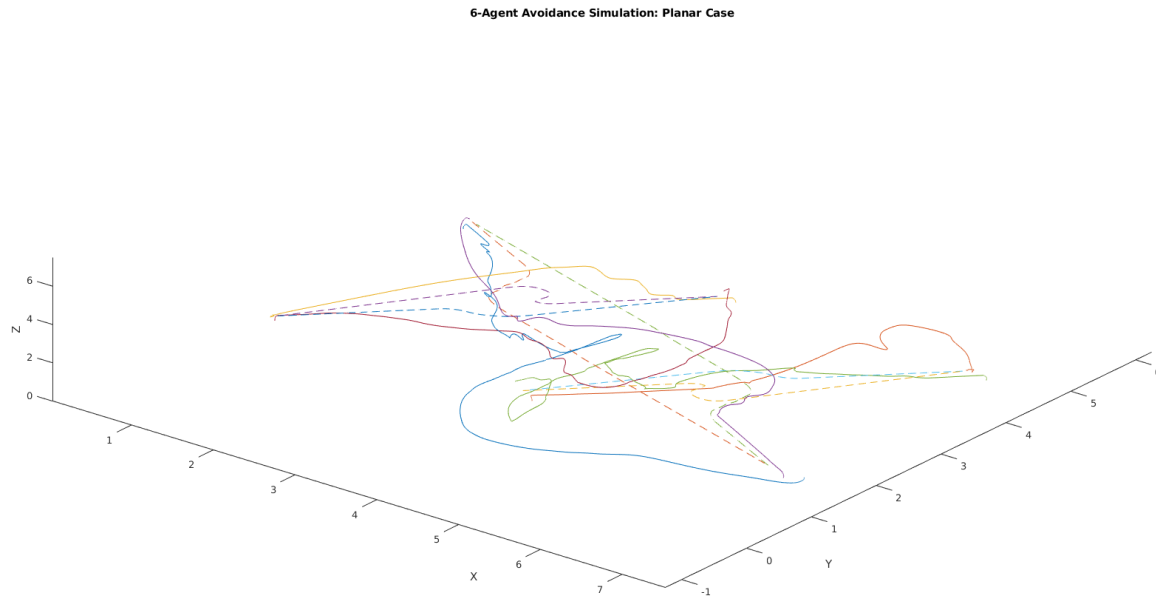


Figure 7-14: Holonomic vs quadcopter simulation, 6-agent planar collision avoidance performance.

	MSE
Agent 1	0.2747
Agent 2	0.4331
Agent 3	0.7296
Agent 4	0.4888
Agent 5	1.2395
Agent 6	0.1453

Table 7-8: Difference between the holonomic agent and quadcopter simulation planar avoidance results.

case multiple collisions occur. This is shown in Figure 7-14 where the dashed lines are the holonomic agents and the solid lines are the quadcopter vehicles. The MSE here is shown in Table 7-8 and is much higher than the previous tests. This is mainly due to the agents inability to process multiple potential collisions quickly enough. One behavior that the holonomic agents sometimes exhibit in high obstacle density environments is circling, where the agent will double back rapidly essentially waiting in place. This behavior is not exhibited by the quadcopter vehicles, instead the vehicles maintain tend to maintain a minimal velocity. This can be attributed to difficulties experienced trying to control a quadcopter using a velocity setpoint.

Conclusions and Future Work

8-1 Conclusions

In this thesis a novel network-decentralized network-estimated collision avoidance and formation producing algorithm was presented. Because of the number of parameters and complexity a closed form analysis presented, it was decided to test the algorithm through simulation. Initially implemented in 2-D using holonomic agents, a key subset of parameters were identified based on the impact they had on the convergence of different multi-agent scenarios. Two separate avoidance regions, the scaling rate of the outer detection region, avoidance angles to maximize distance from an obstacle, inter-agent communication type, communication radii, network-decentralized observer gains, and continuously defined detection regions were all critical for successful convergence. One aspect of this algorithm that is unique is the low-information environment needed for convergence. Each agent only has access to its own estimated position, as well as the estimated positions of other agents within its communication radius. Despite this, convergence guarantees are shown as long as each agent is tuned properly according to its own performance characteristics, eg: agents with higher max velocities must have faster scaling avoidance regions. This shifts some of the burden from externalities, such as successful and reliable communication, to properties internal to each agent. Lastly, the communication method between agents was analyzed. It was found, in some ways counter to intuition, that in multi-agent systems where the agents had more strongly connected graphs (more inter-agent communication) there were slower convergence times. When agents could only communicate locally they tended to interact locally first, avoiding situations where many agents would meet at the center of a formation and have to avoid one another.

The initial implementation and analysis in 2-D yielded a foundation for which further extensions to the algorithm could be built. Most literature concerning decentralized control or estimation is conducted with single and double integrator systems, so a natural question is how well this work extends to real-world real-time vehicles. In this thesis the vehicle of choice was a quadcopter, for which the presented algorithm needed to be extended to 3-D. An extension to 3-D was shown and validated through simulation using holonomic agents, as was done in the 2-D case. The same collision avoidance guarantees presented in the 2-D

case were shown to hold in the extension as well. One of the main aspects of the avoidance portion of the algorithm is that each agent will avoid obstacles by turning to the right. This is a fundamentally 2-D concept and to make this hold in 3-D an arbitrary plane is defined using the obstacle-agent vector of each agent, the XY plane, and a vector orthogonal to the XY plane. This approach yielded avoidance vectors with components in all three dimensions. The detection of potential collisions was also extended to 3-D where the positions of each agent and obstacle were projected onto the XY plane. This solved an additional problem that is currently an open research question in quadcopter research, how to compensate for the downwash effect of other vehicles. By detecting obstacles through a projection onto the XY plane it was ensured that each agent would not pass overtop of one another. Lastly, a free-space metric was developed that allows for tests to be compared even across dimensions, eg: between 2-D and 3-D results. It was shown that the average free-space metric provides valuable insight as to when a certain parameter tuning may experience a loss of performance.

To make the jump from holonomic simulations to real-time quadcopters both a suitable lab environment and vehicle platform were required. The Delft Center for Systems and Control (DCSC) lab was re-designed and virtualized using an ESXi hypervisor running on a Dell r740 server. This allowed for multiple desktop computers and redundant routers to be replaced, and for information pertinent to students to be centralized. Two different options were developed for student access to the virtualized environment, and a communication analysis was performed on both. In addition to the lab, a quadcopter platform was designed with a focus on fast dynamics and student accessibility. The dynamics of a typical quadcopter were presented along with a controller linearized about the vehicle's hover envelope. This look into the dynamics yielded relevant system identification parameters and a first principles identification method was presented which allows for new vehicles to be introduced to the lab. The identification process was later validated in the physical testing portion of the thesis.

Given a newly designed lab environment and quadcopter vehicle, physical testing could be performed. Unfortunately the DCSC department was unable to acquire the designed platform so another was used in its place. A comparison between the performance of a single physical quadcopter, a holonomic agent, and a quadcopter simulation environment with fitted parameters was performed using a variety of trajectories. It was found that a single physical quadcopter closely tracked the behavior of both the holonomic agent and quadcopter simulation. A different simulation environment was utilized for multi-agent testing which focused on real-time characteristics. The testing focus here was on full-information collision avoidance. A two agent collision avoidance scenario was tested for both planar and altitude varying cases and it was found that the real-time vehicles showed larger differences with the holonomic case than in previous testing. When this was extended to the six agent case the differences increased and it was shown that the six agent real-time planar avoidance case resulted in collisions whereas the exact same test was successful with holonomic agents. This was attributed to a number of causes. The first is difficulty with reference velocity tracking for quadcopters. While both the holonomic agents and quadcopters in this thesis utilized backstepping control, the velocity override for avoidance is not as straight forward for quadcopters as in the holonomic agent case. Additionally there was frequent state switching between avoidance maneuvers and reference tracking in the real-time case which can be attributed to asynchronous sampling rates. Since the avoidance tests with six agents exhibited collisions in a full information environment, it was decided to not implement the network-estimation portion of the algorithm in a real-time setting. Network-estimation only adds uncertainty to the information

available to each agent, therefore it is necessary to first better understand why the real-time full information case results in collisions. This more detailed analysis is left as future work.

8-2 Future Work

- **Rigorously analyze the root cause of collisions in the real-time collision avoidance case:** Before the real-time multi-agent case can be further implemented in a network-decentralized way the root cause of the collisions shown in Chapter 7-2 must be better understood. A comparison between velocity controller behavior for both the holonomic agent and the real-time vehicle should be performed. Additionally, some consideration should be given as to why the real-time vehicle switches between avoidance velocity and reference position states more than the holonomic agents, and if this behavior can be reduced or eliminated.
- **Online trajectory generation with applications to real-time avoidance maneuvers:** In this thesis the collision avoidance aspect of the algorithm is implemented as a velocity override of an agent's current behavior. This was shown to work well for the holonomic agents, however it presented some unique difficulties in the real-time quadcopter case. It was shown that the quadcopter vehicle was able to very closely mimic the performance of a holonomic agent when tracking a reference position. Therefore, the online implementation of a position based avoidance trajectory (with the same theme of turning to the right) might yield promising results.
- **Smooth online velocity control of a quadcopter:** For a single quadcopter updating its desired acceleration using a velocity setpoint it is possible to achieve approximately critically damped performance. However, in the multi-agent collision avoidance case the agents are switching between tracking a reference position and an avoidance velocity. This creates issues in the real-time quadcopter simulation. An analysis of these effects and smoothing of the switching behavior might improve real-time performance.
- **Optimal gain selection in the network-decentralized observer:** In this thesis a high gain approach was utilized for the observer, where every agent used the same static gain. However, what constitutes the ideal set of observer gains changes over time based on the multi-agent communication graph. The ability to find time-varying (maybe even optimal!) gains for each communication graph in a decentralized way might lead to interesting insights when compared with the high gain approach used in this thesis.
- **Network decentralized estimation of vehicle velocities:** In this thesis only estimated positions are shared over the network. While this allows for less communication overhead, there are some situations where an agent will perform an avoidance maneuver for an obstacle that, if ignored, would not have posed a threat. By estimating the velocities of obstacles the agents may be able to make more informed decisions as to when avoidance actions are necessary.
- **Dynamically update the parameters defining collision avoidance regions and speed limitations based on the free-space metric:** As it was shown in this thesis the collision avoidance regions were defined by parameters that did not vary with time.

Additionally, the vehicles were uniformly speed-limited and this limit did not consider the capabilities of each agent. This resulted in situations where an agent would travel at slow speeds in environments with a low number of obstacles. One way that would allow the vehicles to converge faster would be to dynamically tie an agents speed limiting and avoidance region parameters to the metric defining free-space.

- **Learning based tuning methods for algorithm parameters:** The algorithm presented in this thesis proved difficult to tune. Once tuned, the results were good and it was shown that these results held in situations that had similar or higher values for the average free-space metric. However, it could be useful to leverage reinforcement learning to discover a set of parameters and compare them to ones known to work.
- **Modify the MAVROS libraries for the Raspberry Pi Zero:** For the physical development of the quadcopter design proposed in Chapter 5 an on-board computer with 2.4 GHz communication capabilities is necessary to share information between the on-board flight controller (FC) and the base station. A library that does this nicely, and is designed for use with Robot Operating System (ROS), is MAVROS. For on-board computing the Pi Zero is ideal because of its size and power consumption (detailed further in [46]), however there is an issue with the instruction set architecture (ISA) used on the Pi Zero, whereas the Pi 3b+ works perfectly well. Unfortunately the Pi 3b+ is not ideal when it comes to size and power consumption. Therefore, porting the MAVROS library over to the Pi Zero is important.
- **Physical swarm implementation:** For this thesis it was only possible to acquire one quadcopter. The next logical step is to acquire more vehicles and test the virtualized lab environment with them. This would also allow for physical testing of real-time collision avoidance, which should then be compared to the real-time results shown from simulations in Gazebo.

Bibliography

- [1] Why do birds flock together? <https://www.howitworksdaily.com/why-do-birds-flock-together/>. Accessed: 2019-22-05.
- [2] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, 28(2):51–64, April 2008.
- [3] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, April 2012.
- [4] Flying machine arena. <https://flyingmachinearena.org/>. Accessed: 2018-09-27.
- [5] Performance of arm virtualization. <https://www.linaro.org/blog/on-the-performance-of-arm-virtualization/>. Accessed: 2019-15-05.
- [6] Ccnny ascending technologies hummingbird. [http://library.isr.ist.utl.pt/docs/roswiki/Robots\(2f\)AscTec.html](http://library.isr.ist.utl.pt/docs/roswiki/Robots(2f)AscTec.html). Accessed: 2019-15-05.
- [7] Draganflyer v ti pro review, tips and tricks. <http://www.andybrain.com/extras/draganflyer.htm>. Accessed: 2019-17-05.
- [8] Qav-r2. <https://www.lumenier.com/consumer/qav-r-2>. Accessed: 2019-17-05.
- [9] Crazyflie 2.1. <https://www.bitcraze.io/crazyflie-2-1/>. Accessed: 2019-20-05.
- [10] S. Bouabdallah. Design and control of quadrotors with application to autonomous flying, 2007.
- [11] H. Huang, G. Hoffmann, S. Waslander, and C. Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. *IEEE International Conference on Robots and Automation*, pages 3277–3282, May 2009.
- [12] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The grasp multiple micro uav testbed. *IEEE Robotics & Automation Magazine Volume 17 Issue 3*, 17(3):56–65, Sept. 2010.

- [13] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafał Jęszefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, 2018.
- [14] Patrick Wensing, Albert Wang, Sangok Seok, David M Otten, Jeffrey H Lang, and Sangbae Kim. Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. *IEEE Transactions on Robotics*, PP:1–14, 01 2017.
- [15] Peter R Wurman, Raffaello D’Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.
- [16] DS Pavlov and AO Kasumyan. Patterns and mechanisms of schooling behavior in fish: a review. *Journal of Ichthyology*, 40(2):S163, 2000.
- [17] Irene Giardina. Collective behavior in animal groups: Theoretical models and empirical studies. *HFSP Journal*, 2(4):205–219, 2008. PMID: 19404431.
- [18] Jason M Graham, Albert B Kao, Dylana A Wilhelm, and Simon Garnier. Optimal construction of army ant living bridges. *Journal of theoretical biology*, 435:184–198, 2017.
- [19] Y. Cao, W. Yu, W. Ren, and G. Chen. An overview of recent progress in the study of distributed multi-agent coordination. *arXiv*, 9(1):427–438, Feb 2013.
- [20] F. Blanchini, E. Franco, and G. Giordano. Structured-lmi conditions for stabilizing network-decentralized control. *IEEE Conference on Decision and Control*, pages 6880–6885, Dec 2013.
- [21] F. Blanchini, E. Franco, and G. Giordano. Network-decentralized control strategies for stabilization. *IEEE Transactions on Automatic Control*, 60(2):491–496, Feb 2015.
- [22] G. Giordano, F. Blanchini, E. Franco, V. Mardanlou, and P. Montessoro. The smallest eigenvalue of the generalized laplacian matrix, with application to network-decentralized estimation for homogeneous systems. *IEEE Transactions on Network Science and Engineering*, 3(4):312–324, Oct 2016.
- [23] G. Giordano. Network decentralized stabilization, control, and estimation, 2016.
- [24] F. Blanchini, D. Casagrande, G. Giordano, and P. Montessoro. A robust decentralized control for channel sharing communication. *IEEE Transactions on Control of Networked Systems*, 4(2):336–346, June 2017.
- [25] A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.
- [26] W. Ren. Collective motion from consensus with cartesian coordinate coupling - part i: Single-integrator kinematics. *IEEE Conference on Decision and Control*, pages 1006–1011, Dec 2008.

-
- [27] M. Aranda, G. Lopez-Nicolas, C. Sagıjıs, and M. Zavlanos. Coordinate-free formation stabilization based on relative position measurements. *Elsevier Systems: Automatica*, 57:11–20, 2015.
 - [28] H. Tanner, A. Jadbabaie, and G. Pappas. Stable flocking of mobile agents, part i: Fixed topology. *IEEE International Conference on Decision and Control*, 2:2010–2015 Vol.2, Dec 2003.
 - [29] H. Tanner, A. Jadbabaie, and G. Pappas. Stable flocking of mobile agents, part ii: Dynamic topology. *IEEE International Conference on Decision and Control*, 2:2016–2021 Vol.2, Dec 2003.
 - [30] N. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. *IEEE Conference on Decision and Control*, 2001.
 - [31] W. Ren. Collective motion from consensus with cartesian coordinate coupling. *IEEE Transactions on Automatic Control*, pages 1012–1017, Dec 2009.
 - [32] P. Lin, K. Qin, Z. Li, and W. Ren. Collective rotating motions of second-order multi-agent systems in three-dimensional space. *Elsevier Systems & Control Letters*, 60(6):1012–1017, 2011.
 - [33] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, March 2006.
 - [34] P. Ogren, E. Fiorelli, and N. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302, Aug 2004.
 - [35] P. Lin and Y. Jia. Distributed rotating formation control of multi-agent systems. *Elsevier Systems & Control Letters*, 59(10):587 – 595, 2010.
 - [36] F. Blanchini, E. Franco, G. Giordano, V. Mardanlou, and P. Montessoro. Compartmental flow control: decentralization, robustness and optimality. *Automatica*, 64:18 – 28, 2015.
 - [37] L. Mier y Teran-Romero, E. Forgoston, and I. Schwartz. Coherent pattern prediction in swarms of delay-coupled agents. *IEEE Transactions on Robotics*, 28(5):1034–1044, Oct 2012.
 - [38] M. Pavone and E. Frazzoli. Decentralized policies for geometric pattern formation and path coverage. *Proceedings of the American Control Conference*, 129(5):633–643, Oct 2007.
 - [39] S. Roumeliotis and G. Bekey. Distributed multi-robot localization. *IEEE Transactions on Robotics*, 18(5):781–785, 2002.
 - [40] D. van Wijk. Network-decentralized control with collision avoidance for multi-agent systems, 2018.
 - [41] R. Saber and R. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, Sept 2004.

- [42] Y. Zhang and Y. Tian. Consentability and protocol design of multi-agent systems with stochastic switching topology. *Automatica*, 45(5):1195 – 1201, 2009.
- [43] J. Goldhirsh and W. Vogel. *Handbook of Propagation Effects for Vehicular and Personal Mobile Satellite Systems*. John Hopkins / The University of Texas at Austin, 1998.
- [44] Slamtec rplidar a3. <http://www.slamtec.com/en/Lidar/A3>. Accessed: 2019-15-04.
- [45] Slamtec rplidar a3 documentation. http://bucket.download.slamtec.com/8a9aa114ad4e372110aa9d9783e441c79ab76d90/LD310_SLAMTEC_rplidar_datasheet_A3M1_v1.3_en.pdf. Accessed: 2019-15-04.
- [46] P. Ledzian. Network-decentralized collision avoidance with a scalable quadcopter lab design: Operating guide. Technical report, Delft University of Technology, July 2019.
- [47] J. van den Berg, M. Ling, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. *IEEE International Conference on Robotics and Automation*, pages 1928–1935, 2008.
- [48] D. Wilke, J. van den Berg, and D. Manocha. Generalized velocity obstacle. *IEEE International Conference on Intelligent Robots and Systems*, pages 5573–5578, 2009.
- [49] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.
- [50] Fibonacci sphere algorithm. <https://www.openprocessing.org/sketch/41142>. Accessed: 2019-03-02.
- [51] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron. Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006.
- [52] M. Valenti, B. Bethke, D. Dale, A. Frank, J. McGrew, S. Ahrens, J. How, and J. Vian. The mit indoor multi-vehicle flight testbed. *2007 IEEE International Conference on Robotics and Automation*, pages 2758–2759, April 2007.
- [53] J. How. Aero-astro magazine highlight – raven: Testbed for autonomous uavs. <http://web.mit.edu/aeroastro/news/magazine/aeroastro-no4/raven-uav.html>, 2006-2007. Accessed: 2018-10-09.
- [54] A. Kushleyev, D. Mellinger, and V. Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, Nov. 2013.
- [55] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control. *IEEE International Conference on Robotics and Automation*, pages 2520–2525, May 2011.
- [56] Asctec hummingbird. <http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-hummingbird/>. Accessed: 2018-10-05.
- [57] S. Lupashin, A. Schödl, M. Hehn, and R. D’Andrea. The flying machine arena as of 2010. *IEEE International Conference on Robotics and Automation*, pages 2970–2971, May 2011.

-
- [58] S. Lupashin, M. Hehn, M. Mueller, A. Schoellig, M. Sherback, and R. D'Andrea. A platform for aerial robotics research and demonstration: The flying machine arena. *Elsevier: Mechatronics* 24, 24(1):41–54, Feb. 2014.
 - [59] Motive: Optical motion capture software. <https://optitrack.com/products/motive/>. Accessed: 2018-11-01.
 - [60] J. Sticha. Validating the real-time capabilities of the ros communication middleware, 2014.
 - [61] Poweredge r740 specifications. https://i.dell.com/sites/csdocuments/Shared-Content_data-Sheets_Documents/en/poweredge-r740-spec-sheet.pdf. Accessed: 2019-15-05.
 - [62] Ubiquiti unifi ac shd ap. <https://unifi-shd.ui.com/>. Accessed: 2019-15-05.
 - [63] Vm workstation 15 pro. <https://www.vmware.com/nl/products/workstation-pro/workstation-pro-evaluation.html>. Accessed: 2019-20-05.
 - [64] John Ector and Ryan Kenny. Addressing swap challenges in military platforms with 65-nm fpgas and structured asics. Technical report, Altera Coporation, 10 2007.
 - [65] The asctec hummingbird for flight dynamics & swarming. http://www.asctec.de/downloads/public/HB_AscTec-Hummingbird_safety-data-sheet.pdf. Accessed: 2019-15-05.
 - [66] 3d fpv camera, the blackbird 2. <https://flyingmodscom.files.wordpress.com/2017/06/3dfpv-manual-blackbird2-07.pdf>. Accessed: 2019-15-05.
 - [67] Efficiency vs. performance - how to build a drone with a long flight time. <https://www.flyingtech.co.uk/blog/efficiency-vs-performance-how-build-drone-long-flight-time>. Accessed: 2019-15-05.
 - [68] Parrot bebop drone. <https://www.drones.nl/drones/parrot-bebop-drone/specs>. Accessed: 2019-17-05.
 - [69] Rc benchmark otus tracking system. <https://docs.rcbenchmark.com/en/motion-capture/otus-tracker.html>. Accessed: 2019-20-05.
 - [70] Sneak peak of what's coming at icra 2019. <https://www.bitcraze.io/2019/05/sneak-peak-of-whats-coming-at-icra-2019/>. Accessed: 2019-10-06.
 - [71] Pixhawk 4 mini. <http://www.holybro.com/product/pixhawk4-mini/>. Accessed: 2019-20-05.
 - [72] Lorenz Meier. Px4 development guide (master). Technical report, The Dronecode Project, 2019.
 - [73] Gazebo community. <https://community.gazebosim.org/>. Accessed: 2019-20-05.
 - [74] Raspberry pi documentation faq. Technical report, Raspberry Pi Foundation, 2019.

- [75] S. Bouabdallah and R. Siegwart. Full control of a quadrotor. *IEEE International Conference on Intelligent Robots and Systems*, pages 153–158, Oct. 2007.
- [76] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, control of quadrotor. *IEEE Robotics & Automation Magazine*, 19(3):20–32, Sept. 2012.
- [77] K. Alexis, G. Nikolakopoulos, and A. Tzes. Model predictive quadrotor control: attitude, altitude and position experimental studies. *IET Control Theory and Applications*, 6(12):1812–1827, Aug. 2011.
- [78] D. Gurdan, J. Stumpf, M. Achtelik, K. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. *IEEE International Conference on Robots and Automation*, pages 361–366, April 2007.
- [79] D. Mellinger. Trajectory generation and control for quadcopters, 2012.
- [80] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar. Design, modeling, estimation and control for aerial grasping and manipulation. *IEEE International Conference on Intelligent Robots and Systems*, pages 2668–2673, Sept. 2011.
- [81] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar. *Influence of Aerodynamics and Proximity Effects in Quadrotor Flight*, pages 289–302. January 2013.
- [82] Fusion 360 cad software. <https://www.autodesk.com/products/fusion-360/students-teachers-educators>. Accessed: 2019-07-05.

Glossary

List of Acronyms

DCSC	Delft Center for Systems and Control
LIDAR	Light Detection and Ranging
RSSI	Received Signal Strength Indicator
CoG	center of gravity
GRASP	general robotics, automation, sensing, and perception
UPenn	University of Pennsylvania
EFF	Earth fixed frame
BFF	body fixed frame
SO(3)	special orthogonal group 3
ENU	East-North-Up
DCMs	direction cosine matrices
DOF	degrees of freedom
IMU	inertial measurement unit
CAD	computer aided design
ESC	electronic speed controller
RPM	rotations per minute
LiPo	Lithium polymer
SWAP	size, weight, and power
MIT	Massachusetts Institute of Technology

RAVEN	Real-time indoor Autonomous Vehicle test ENvironment
FMA	Flying Machine Arena
MoCap	motion capture
RC	radio controlled
CPU	central processing unit
ROS	Robot Operating System
SoC	System on a Chip
UDP	User Datagram Protocol
FOV	field of view
TCP	Transmission Control Protocol
OS	operating system
WNIC	wireless network interface card
SSD	solid-state drive
DDR4	double data-rate 4
RAM	random-access memory
VM	virtual machine
SSH	secure shell
VDI	virtual desktop infrastructure
AP	access point
MU-MIMO	multi-user, multiple input, multiple output
FIFO	first in, first out
PoE	power over ethernet
SWaP	size, weight, and power
MTOW	maximum takeoff weight
COTS	commercial off the shelf
DC	direct current
FC	flight controller
IMU	inertial measurement unit
RTOS	real-time operating system

I2C	inter-integrated circuit
SPI	serial peripheral interface
UART	universal asynchronous reciever-transmitter
ISA	instruction set architecture
PCL	Point Cloud Library
iDRAC	integrated Dell remote-access controller
MSE	mean-squared error

