DISS. ETH NO. 21157

The Distributed Flight Array

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

RAYMOND OUNG

M.Sc. in Mechanical Engineering, ETH Zurich
B.A.Sc. in Electrical Engineering, University of Toronto

born May 10, 1983
citizen of Canada

accepted on the recommendation of

Prof. Dr. Raffaello D'Andrea, ETH Zurich, examiner
Prof. Dr. Roland Siegwart, ETH Zurich, co-examiner
Prof. Dr. Daniela Rus, MIT, co-examiner

2013

The Distributed Flight Array

Raymond Oung

Institute for Dynamic Systems and Control
ETH Zurich

Zurich, April 2013

Cover illustration by Carolina Flores.

Institute for Dynamic Systems and Control
ETH Zurich
Switzerland

Abstract

Information processing, sensing, and communication technologies have been advancing at a phenomenal rate. It is quickly becoming economically viable to embed powerful computers, sensors, and communication technology into almost any physical device. For the field of control systems and robotics, this has made it possible to envision a system-level architecture in which many relatively simple components coordinate and cooperate in order to perform a joint task. This distributed architecture promisses much in terms of performance, reliability, and design simplicity. However, it requires new concepts in system-level design, and a unified picture of control, communication, and computation.

This dissertation aims to develop a state-of-the-art mobile robotics testbed that abstracts many of the challenging, real world issues that are likely to be encountered when developing the next generation of controllable distributed dynamic systems. Doing so will require novel concepts in systems engineering and an understanding of the effects in which communication plays in such systems. It will force us to develop new algorithms and/or extend existing ones, as well as to examine their limits and potential.

Our mobile modular robotics platform, the Distributed Flight Array, consists of multiple autonomous units that are able to drive, dock with its peers, and coordinate with one another in order to drive and fly together. It is a combination of two advancing fields in robotics – modular robotics and micro aerial vehicles – drawing research and design challenges existing in both areas.

The first part of this dissertation primarily details the hardware platform. It presents the feasibility study, which initiated the project, the design revisions that followed, and how it has evolved into what it currently is today. The second part of this dissertation describes algorithms that were motivated in part by the development of this testbed, namely: a distributed tilt estimation strategy that employs a network of distance measurement sensors, which led to the development of a robust distributed average consensus algorithm; and a decentralized control strategy fit for any flight-feasible configuration of a multi-rotor vertical take-off and landing vehicle.

Zusammenfassung

Informationsverarbeitung, Sensorik und Kommunikationsstechnik haben innerhalb von kurzer Zeit grosse Fortschritte gemacht. Aus ökonomischer Sicht wird es immer einfacher Recheneinheiten, Sensoren und Kommunikationseinheiten in nahezu jedes Endgerät einzubetten. Im Bezug auf die Regelungstechnik und Robotik sind daher Systemarchitekturen vorstellbar, in denen sich viele relativ einfache Komponenten koordinieren und miteinander kooperieren um eine gemeinsame Aufgabe zu erfüllen. Eine solche verteilte Systemarchitektur bietet viele Vorteile: erhöhte Rechenleistung und Zuverlässigkeit sowie ein einfacheres Design. Es bedarf allerdings neuer Konzepte im Systemaufbau und eines Gesamtbildes der Regelungs-, Kommunikations- und Computertechnologie.

Der Zweck dieser Arbeit ist es, eine mobile Roboterplattform nach aktuellestem Stand der Technik zu entwickeln. Diese Plattform abstrahiert herausfordernde Probleme, von denen zu erwarten ist, dass sie in ähnlicher Form bei der Entwicklung der nächsten Generation von regelbaren verteilten dynamischen Systemen auftreten werden. Dieses bedarf neuartiger Konzepte im Systementwurf und eines besseren Verständnis der Kommunikation der einzelnen Systemkomponenten untereinander. Dies wird von uns verlangen, neue Algorithmen zu entwickeln und existierende Algorithmen zu erweitern.

Unsere mobile modulare Roboterplattform, das Distributed Flight Array (DFA), besteht aus mehreren autonomen Robotern, die sich am Boden bewegen und in einer Vielzahl von Konfigurationen fahren und fliegen können. Es ist eine Kombination von zwei sich schnell entwickelnden Forschungsgebieten der Robotik – *modular robotics* und *micro aerial vehicles*. Die Herausforderungen beider Forschungsgebiete finden sich im DFA vereint.

Der erste Teil dieser Dissertation beschreibt die Hardware-Plattform. Er enthält die Machbarkeitsstudie, welche zu diesem Projekt geführt hat, die darauf folgenden Design-Änderungen und den aktuellen Stand der Plattform. Der zweite Teil dieser Dissertation beschreibt Algorithmen, die durch die Entwicklung dieser Plattform entstanden sind. Dies ist zum einen eine Verteilte-Drehwinkel-Schätz-Strategie zum Auswerten eines Netzwerkes von Abstandssensoren, die zur Entwicklung eines robusten *distributed average consensus* Algorithmus führte. Zum anderen handelt es sich um eine dezentrale Regelungsstrategie, die in der Lage ist eine beliebige Konfiguration von Modulen zu fliegen.

Acknowledgements

This work would not have been possible without the support and contribution from a number of individuals, and here I extend to them my sincerest gratitude.

First, I would like to acknowledge Professor Raffaello D'Andrea, for the vision and spark that ignited this work, for giving me the opportunity and freedom to explore my own research interests, and for recalibrating my emphasis on details.

I would like to thank the students, technical staff, and colleagues of the 2008 – 2009 *!And Yet it Moves* class for providing me with such a stimulating atmosphere during my first year at ETH Zurich. In particular, I am grateful to Matthew Donovan for showing me how to craft things in ways I could only imagine (and for not taking things too seriously), Frédéric Bourgault for showing me how to take to the skies, and Daniel Burch for the adventures that took place outside of the office.

One of my greatest pleasures at ETH Zurich was working so closely with all the support staff. I would especially like to thank Igor Thommen for his support in mechanical design, introducing me to the latest in modern fabrication techniques, and teaching me how to make things *click*. Hans Ulrich Honegger, for all of the tedious work required to realize our designs. Marc-Andre Corzillius, for his never-failing enthusiasm, his sharp wits, skillful routing and debugging abilities, and for being there to simply bounce ideas off. Carolina Flores (and not to forget her assistant Nena), for their artistic flare in creating some of the most wonderful illustrations and animations for the project, and introducing me to the world of improv. Katharina Munz for taking care of all the administrative duties, finances, being there to talk to, and making life at the office that much easier.

I was fortunate to have supervised many students during my time here, and one of my greatest joys was being able to collaborate with many brilliant minds. We were not always sucessful in achieving the results that we wanted, but it was always an invaluable learning experience for me.

I would like to thank all of my colleagues at the Institute for Dynamic Systems and Control. In particular, I would like to thank Mohanarajah Gajamohan for his humility and calm attitude, methodical and persistent thinking, and his insightful thoughts about any problem that we would discuss on our weekly morning jogs around the park. Sebastian Trimpe, for sharing his academic insights with me, for his professionalism in the office, and for the beers outside of the office. Philipp Reist, for his constant source of enthusiasm, for his encouragement, and for all the great times that we've shared both in and out of the office together. Maximilian Kriegleder, who collaborated with me on numerous aspects concerning this work, from soldering surface mount components to late night whiteboard discussions. He was a constant source of motivation, and it still amazes me how he doesn't tire of it.

I would also like to express my gratitude to ETH Zurich and the Swiss National Science Foundation, for their support and funding of my research. I believe that there are very few institutions in the world that are able to provide the same level of support and resources as was given to me.

And finally, I would like to thank my friends and family. I believe that each of you respresent a pillar of inspiration, strength, and support in my life. I would especially like to thank my parents and brother for their unconditional support, for always questioning the things that mattered, their trust and patience along this long road of education, and for making me the person that I am today.

Contents

1 Introduction

Our society is experiencing a shift from traditional industry to an economy based on the publication, consumption, and manipulation of information – this period in time may be referred to as the Information Age. One of its characteristics is the ability of individuals and machines to freely transfer information anywhere in the world, and to have instant access to information that would have been previously difficult or otherwise impossible to obtain. The Information Age has been precipitated in part by the micro-miniaturization advances in computer technology, starting with the arrival of the personal computer in the late 1970s, the Internet reaching critical mass in the early 1990s, and the mainstream adoption of computer technology by the general public in the two decades thereafter. A direct consequence of this adoption has been the rapid development of computer, sensor and communication technology. This has resulted in an exponential rise in portable and mobile electronic devices such as smart phones, global positioning systems, and laptop computers – to name only a few. It is now quickly becoming economically viable to embed powerful computers, sensors, and communication technology into almost any physical device, leading to what some call cyber-physical systems [1]. What has already led to a dramatic rise of information will only continue to do so for the foreseeable future. This capability alone, however, has little value without the ability to act on such information.

Control theory at its heart is an information science. Traditionally controlled dynamic systems have relied on feedback systems theory as a means of extracting information about the state of the system to ensure reliable behavior in the presence of uncertainty, see Figure 1.1. Using this model, control theory has made spectacular advances since its inception both in developing mathematical tools and applying this model to many different problems. This, in part, has generated technological advances that permeates modern systems of everyday life, ranging from the simple home thermostat to sophisticated flight control systems on airplanes.

The implicit assumption that is made in these systems – that is inherent to this model – is that almost unlimited amounts of information can be communicated and processed by its components. In the majority of traditionally controlled dynamic processes, this assumption has largely held true. These applications typically involved systems with a relatively small number of degrees of freedom, inputs and outputs, as well as on a

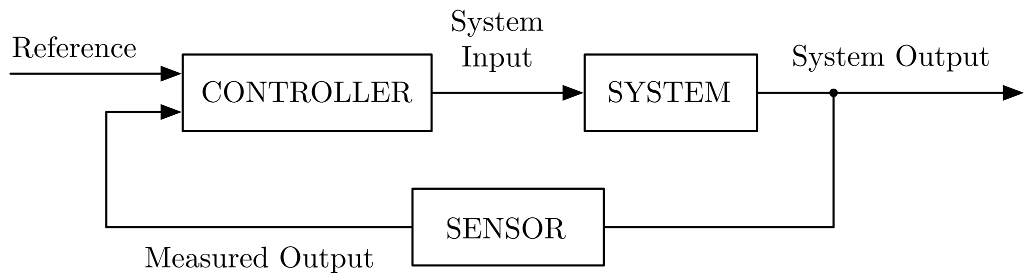


Figure 1.1 A traditional control system block diagram consists of a dynamic process whose state needs to be controlled. Take for example the cruise control on a car: a controller, which uses measurements and reference as input (e.g. speed), manipulates the inputs to a system (e.g. throttle position) to obtain the desired effect on the output of the system.

centralized processing architecture. The resulting strategies have not necessarily been keeping pace with present-day technology.

With present-day technology, it is now possible to envision and realize systems in which multiple simple components cooperate to achieve a joint task. There exists three distinct achitectures for such *multi-agent* systems:

Centralized Each agent shares its information with all other agents in the system, and each agent is capable of processing all of this information. Although there are multiple physical components, the system can be modelled as a single agent system (as in Figure 1.1), and traditional methods can be employed (disregarding computational issues).

Decentralized Each agent makes its decision based on the information that it possesses locally. In this case, information is not shared despite having a joint task. Instead, agents coordinate their decision by exploiting some correlated property of the system. Take for example a group of lifeguards working together at a swimming pool. Their joint task is to keep the patrons at the pool safe. To do this, they don't necessarily need to communicate. Instead they can divide the pool into sections and each lifeguard will guard his/her section of the pool.

Distributed Each agent can share its information with some other agents in the system. This is a cooperative approach, where agents directly coordinate their actions in order to achieve a joint goal. Take for example the combined motion of a flock of birds, a school of fish, or a herd of land animals. There are survival benefits when moving together in large groups, such as protection, mating, aerodynamics, and warmth. In this case each animal navigates according to its local perception of the dynamic environment, while taking into account the heading of its peers in its immediate vicinity [2].

In using such a system architecture, the individual agents or components can be relatively simple in design, while not compromising its overall ability and performance. With

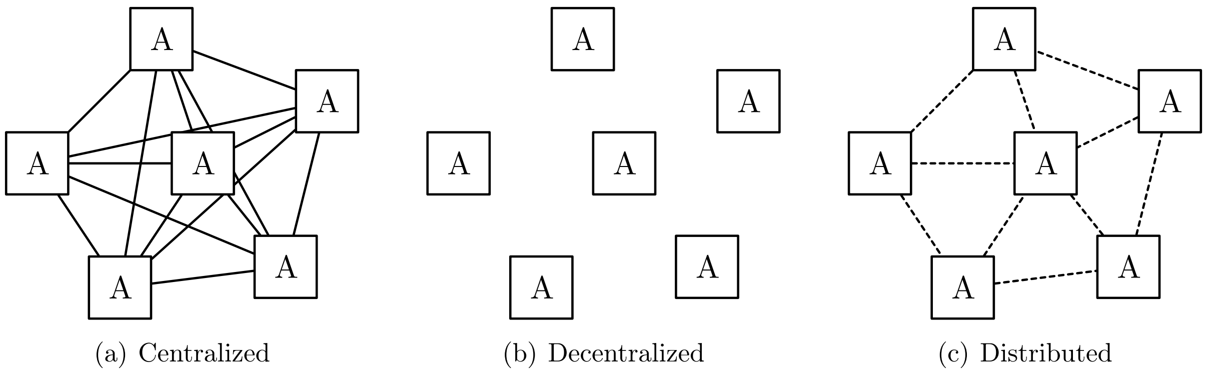


Figure 1.2 Control architectures for multi-agent systems, where A respresents an agent, solid lines represent no-loss communication links, and dotted lines represent lossy communication link.

more agents than needed for a task comes an increasing amount of flexibility and redundancy, leading to an overall increase in reliability. Such systems in general are considered to be rich with information. However, each agent may have access to different information and sharing *all* information may not be feasible, such as the case of a centralized system architecture. Consider the practical implications such as imperfect communication and limited processing capabilities. This design architecture therefore challenges the assumption that the flow of information is unrestricted and reliable.

Information is thus a double-edge sword. Sharing it allows components to cooperate and achieve possibly better performance and/or enhanced capabilities, which would otherwise not be possible alone. Due to physical limitations, however, its sheer quantity cannot neccessarily be communicated and/or processed, leading to decentralized and/or distributed solutions. To fully exploit the benefits promised by such system, there is a need to address the issues involved in its design. Any progress made in understanding the analysis and synthesis of such systems would then necessarily expand our knowledge in communication, computation and control.

Part of this dissertation aims to develop fundamental algorithms for controlling systems with many interconnected components whose dynamics are not precisely known and/or change with time. We want to push the limits of what can be achieved by feedback control with today's cutting edge technology on both a theoretical and practical level. In order to do this, we have designed a state-of-the-art mobile robotics testbed that abstracts many of the challenging, real world issues that are likely to be encountered when developing the next generation of controllable distributed dynamic systems.

The robotic paradigm, in its traditional sense, is appropriate as a control systems testbed because it inherently embodies its behaviour, which has three distinct phases [3]: (1) sense, (2) plan, and (3) act. The following introduces our mobile robotics testbed and outlines some of its challenges.

The Distributed Flight Array

The Distributed Flight Array (DFA) is a mobile modular robotics platform consisting of multiple autonomous units that are able to drive, dock with its peers, and coordinate with one another in order to drive and fly together, see Figure 1.3. Each unit, or what we will refer to as a *module* or an *agent*, is the smallest independent autonomously working component of the system that is able to coordinate with its neighbour(s) in order to perform a joint task – its system architecture is, by design, distributed.

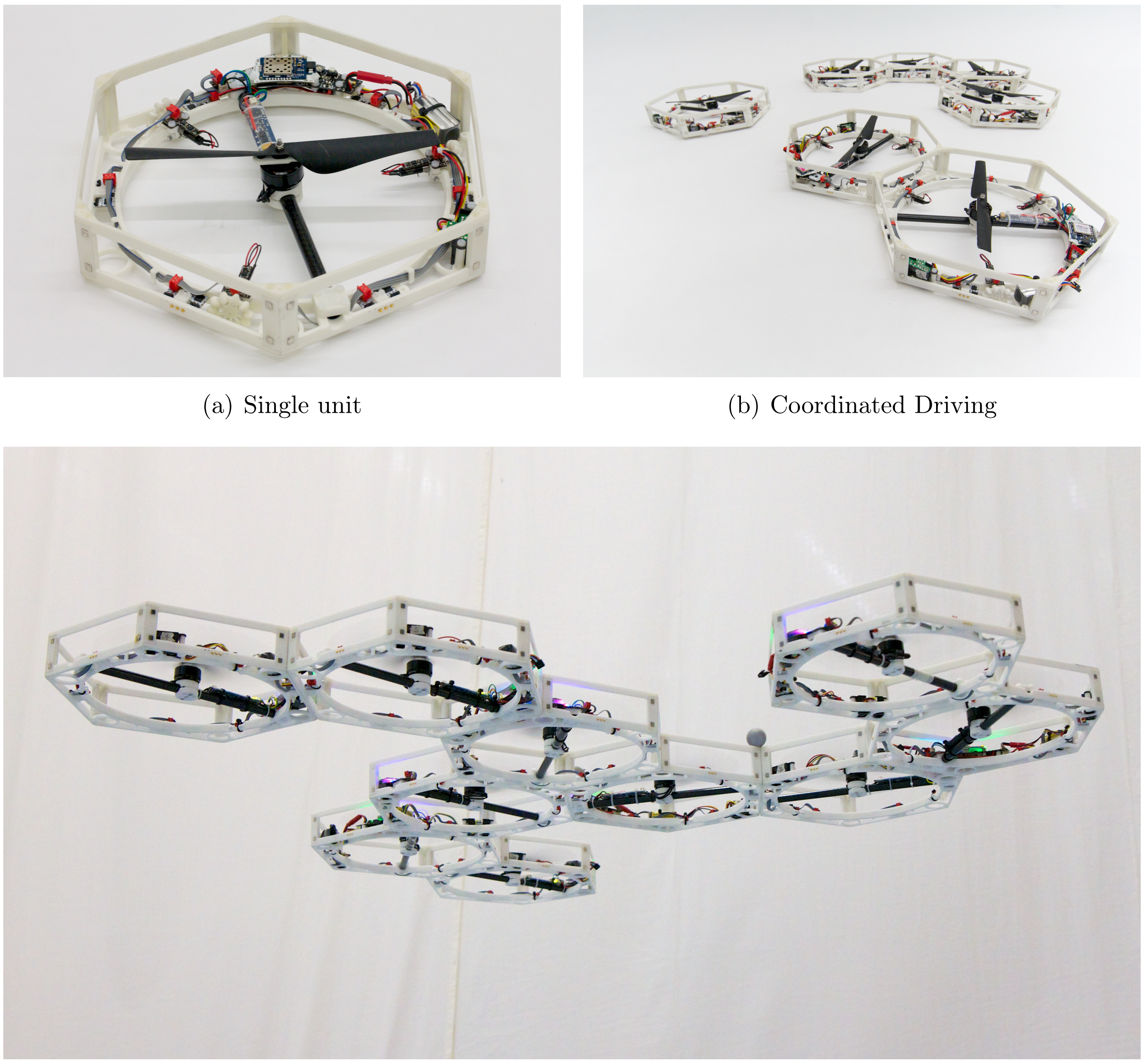

(a) Single unit
(b) Coordinated Driving
(c) Coordinated Flight

Figure 1.3 The Distributed Flight Array is a modular robotics platform consisting of multiple autonomous hexagonal units that are able to drive, dock with its peers, and fly in a limitless number of ad hoc configurations. It is a testbed for distributed estimation and control.

The DFA can be seen as a mixture of both modular robotics (see [4]–[6] and references therein) and micro aerial vehicles (see [7]–[13] and references therein). These areas of robotics research were initiated a little more than a decade ago, thanks in part to the technological advances in computing and sensing available at that time, and their progress has flourished since. Modular robotics, in general, offers its users a versatile platform that can support multiple modalities for locomotion, manipulation, and/or perception; components (or modules) of such systems can be rearranged in order to adapt to new circumstances, perform new tasks, and/or recover from damage. Micro aerial vehicles are a direct consequence of miniaturization advancements in computer and sensor technologies. They offer a new physical dimension to robotics research, which was previously constrained to planar ground-based mobile platforms. Our work draws inspiration from the research challenges existing in both areas. In particular, our intent is to develop a modular robotic system like no other, one that is capable of flight, which will require novel concepts in systems engineering; it will force us to develop, employ, and/or adapt existing strategies in distributed estimation and control, and to have us examine their limits and potential.

More than just a hardware proof of concept, the DFA is a platform that is rich with real-world engineering problems and serves as testbed for developing algorithms in, but not limited to, distributed estimation and control. Its purpose is to motivate the development of algorithms with the expectation that the number of agents (or modules) in our system will grow, and that these algorithms function robustly on the physical hardware. Its secondary function is to illustrate control theory research, which is often plagued by its abstract concepts and mathematical complexity, in a tangible way to a mainstream audience with the hope of inspiring researchers of tomorrow.

References

[1] E. A. Lee, “Cyber physical systems: Design challenges,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-8, Jan 2008. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-8.html

[2] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 25–34.

[3] R. R. Murphy, *An Introduction to AI Robotics*. MIT press, 2000.

[4] S. Murata and H. Kurokawa, “Self-reconfigurable robots,” *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 71–78, 2007.

[5] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, “Modular self-reconfigurable robot systems [Grand challenges of robotics],” *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, March 2007.

[6] K. Gilpin and D. Rus, “Modular robot systems,” *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 38–55, 2010.

[7] S. Bouabdallah, P. Murrieri, and R. Siegwart, “Design and control of an indoor micro quadrotor,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 5. IEEE, 2004, pp. 4393–4398.

[8] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, “The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC),” in *Proceedings of Digital Avionics Systems Conference*, vol. 2. IEEE, 2004, pp. 12.E.4–121–10.

[9] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, “PIXHAWK: A system for autonomous flight using onboard computer vision,” in *IEEE International Conference on Robotics and Automation.* IEEE, 2011, pp. 2992–2997.

[10] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, “The GRASP multiple micro-UAV testbed,” *Robotics & Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56–65, 2010.

[11] R. Mahony, V. Kumar, and P. Corke, “Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sept. 2012.

[12] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, “Real-time indoor autonomous vehicle test environment,” *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, Apr. 2008.

[13] V. Kumar and N. Michael, “Opportunities and challenges with autonomous micro aerial vehicles,” *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1279–1291, 2012.

Contribution and Organization

The bulk of this dissertation is comprised of four main chapters, Chapters 2 – 5, which can be read independently; the notation between any two chapters may differ slightly. The chapters are ordered logically, in the way one would expect to design and implement the system. The chapters draw its information mainly from the author's publications and design notes.

Below lists the specific contributions made in each of the subsequent chapters, as well as the references used.

Chapter 2: Feasibility Study
This chapter details the feasibility study conducted on the design and implementation of the Distributed Flight Array. It introduces the goals and desired capabilities of the testbed; a scalable flight control strategy; and it examines three major aspects of the vehicle: (1) passive docking mechanism, (2) coordinated driving, and (3) coordinated flight. Preliminary experiments were made using a proof-of-concept version of the vehicle, which demonstrated each of the desired capabilities of the testbed.

Major contributions described in this chapter include:

- Proposal and development of a novel proof-of-concept distributed mobile embedded platform for distributed estimation and control
- A first-order flight dynamics model for a rigid body multi-rotor vertical take-off and landing vehicle
- A scalable flight control strategy that uses an intuitive set of tuning parameters
- Demonstrated with experiments the various desired capabilities of the testbed, which includes: passive docking mechanism, coordinated driving, and coordinated flight

The results contained in this chapter were published in

[1] R. Oung and R. D'Andrea, "The Distributed Flight Array," *Mechatronics*, vol. 21, no. 6, pp. 908–917, Sept. 2011.

[2] R. Oung, F. Bourgault, M. Donovan, and R. D'Andrea, "The Distributed Flight Array," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, USA, May 2010, pp. 601–607.

[3] R. Oung, A. Ramezani, and R. D'Andrea, "Feasibility of a Distributed Flight Array," in *Proceedings of the IEEE Conference on Decision and Control*, Shanghai, China, Dec. 2009, pp. 3038–3044.

Chapter 3: Design and Implementation
This chapter describes in detail the design and implementation of the DFA as it has

evolved from its proof-of-concept stage, described in Chapter 2. A significant amount of improvements were made and the most recent generation of the vehicle is capable of both driving and flying in wide variety of configurations. It also describes the infrastructure, such as the base station, software architecture, and simulation environment required for using the DFA as a research platform.

Major contributions described in this chapter include:

- To this author's knowledge, implementation of the first modular reconfigurable vertical take-off and landing vehicle, which is also capable of mobilizing on the ground
- Custom designed hardware, including: chassis, wheels, and electronics
- A passive genderless docking mechanism, with a wired and wireless inter-module communication interface
- A wireless base-station and scalable software infrastructure for sending remote commands, logging telemetry, sending experiment parameters, etc.
- A simulation framework for testing various sub-components of the firmware and for conducting a variety of simulated experiments

Chapter 4: Real-Time Distributed Tilt Estimation
This chapter describes a means for estimating the tilt of a rigid body using a network of distance measurement sensors, which would otherwise be impossible to do using one of these sensors alone. This work is motivated by the DFA, where tilt estimation is of primary importance for flight control. In this case, each agent is equipped with such a distance measurement sensor and by sharing appropriate information with its neighbours, agents can estimate the tilt of the entire vehicle. In a centralized architecture, where all sensor measurements are available to all agents, a linear-optimal estimate can be trivially computed by solving a linear least squares problem. More interesting, however, is to accomplish the same task in a distributed manner. The centralized solution can be reformulated as the solution to two averaging problems. These averaging problems may then be solved by employing one of the many existing distributed average consensus methods available in literature. In order to implement such a method on our system, however, these algorithms need to be made more robust to a variety of network adversities. We extended an existing distributed average consensus algorithm for this purpose and prove its validity. We then demonstrate with experiments how each agent in this network can robustly estimate the tilt of the body without the need of a data centre, simply by exchanging information with its peers. This tilt estimate, however, is delayed and has a relatively slow update rate, which scales with the algebraic connectivity of the network. To compensate for this delay, the distributed tilt estimate is fused with local angular rate measurements in order to obtain a real-time distributed tilt estimate that can be used for controlling a rigid body with relatively fast dynamics.

Major contributions described in this chapter include:

- A method for estimating the tilt of a rigid body using a network of distance measurement sensors, which would otherwise be impossible to achieve with just one of these sensors alone
- A generalized distributed average consensus algorithm with the following key properties: (1) employs a homogenous protocol; (2) robust to network adversities such as asynchronous communication, unreliable communication links, and swiching topologies; (3) results in an unbiased average; (4) capable of what in literature is referred to as dynamic consensus; (5) well defined convergence characteristics and tuning parameters
- Simulation and experimental results demonstrating the key properties of our distributed average consensus algorithm, as well as results for our proposed distributed tilt estimation method

The results contained in this chapter were published and/or submitted to

[1] M. Kriegleder, R. Oung, and R. D'Andrea, "Distributed altitude and attitude estimation using a network of distance measurement sensors," *To be submitted*, 2013.

[2] ——, "Asynchronous implementation of a linear average consensus algorithm on a distributed embedded system," in *Submitted to the IEEE/RSJ International Conference of Intelligent Robots and Systems*, Tokyo, Japan, 2013.

[3] ——, "Distributed altitude and attitude estimation from multiple distance measurements," in *Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems*, Algarve, Portugal, Oct. 2012, pp. 3626–3632.

Chapter 5: Decentralized Flight Control
This chapter presents a detailed analysis of all aspects of the DFA pertaining to flight. In particular, it presents parameterized decentralized control strategy capable of controlling *any* flight-feasible configuration of a modular vertical take-off and landing vehicle. We describe a method for optimizing its tuning parameters in order to achieve the best possible performance subject to the system's physical constraints. We then propose a method for mapping the configuration space of the vehicle to the control parameter space. Experimental results are included, demonstrating flight for a variety of configurations, both indoors and outdoors.

Major contributions described in this chapter include:

- A generic scalable methodology for controlling any flight-feasible configuration of a multi-rotor vertical take-off and landing vehicle
- A method for mapping the configuration space of the vehicle to its control parameter space

- To this author's knowledge, the first set of experimental results demonstrating flight of a multi-rotor vertical take-off and landing vehicle in a variety of configurations, both indoors and outdoors

The results contained in this chapter were published and/or submitted to

[1] R. Oung and R. D'Andrea, "The Distributed Flight Array: Design, implementation, and analysis of a modular vertical take-off and landing vehicle," *International Journal of Robotics Research, Under Review (Since Jan. 2013).*

[2] R. Oung, M. Picallo, and R. D'Andrea, "A parameterized control methodology for a modular flying vehicle," in *Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems*, Algarve, Portugal, Oct. 2012, pp. 532–538.

This dissertation concludes with Chapter 6 by taking a retrospective glance at the work accomplished. It discusses in a broad sense open problems, both practical and theoretical, and future research directions. Conclusions that summarize the specific contributions and/or provide direct extensions to the work are presented at the end of each chapter. For completeness and a framework for future research and development, appendices have been included. In Appendix A, we touch on coordinated driving, an aspect of the DFA that receives little discussion in the main body of work. In here, the kinematic equations for both a single module and a general multiple-module vehicle is provided as well as real-world constraints that should be taken into consideration. Finally, Appendix B outlines a decentralized self-assembly algorithm for the DFA and a method of implementation.

2

Feasibility Study

The development of a Distributed Flight Array (DFA) proof-of-concept started in the fall of 2008 within the framework of an 8-month long project-based design class called *!And Yet It Moves*, which was taught by Raffaello D'Andrea, Matt Donovan, and Frédéric Bourgault at ETH Zurich. The author joined the class in the end of 2008 as a teaching and research assistant. The remaining participants consisted of two technicians, and eight Master-level students from electrical engineering, mechanical engineering, and computer science. The result was the successul completion of a 4-module prototype in July 2009. This prototype demonstrated three key aspects of the DFA: (1) docking, (2) coordinated driving, and (3) coordinated flight.

Outline

This chapter discusses the first steps taken towards developing the DFA, and sets the foundation for future work. It summarizes three key aspects of the vehicle – docking, driving, and coordinated flight – and presents experimental results for each. The chapter begins with a high-level system description of a DFA module in Section 2.1. A drive model and flight model is then presented in Section 2.2. A docking strategy is summarized in Section 2.3 and parameterized control strategy for hover control is described in Section 2.4. Experimental results are then presented in Section 2.5 and concluding remarks are made in Section 2.6.

2.1 Design

The design challenges of the DFA mirror those of modular reconfigurable robots and micro aerial vehicles, which include electromechanical interconnection, inter-module communication, and energy storage [1]–[5]. The design of the system can be divided into four tightly interconnected sub-systems: (A) Chassis and Docking Mechanism; (B) Drive Unit; (C) Flight Unit; and (D) Sensing, Communication & Computation.

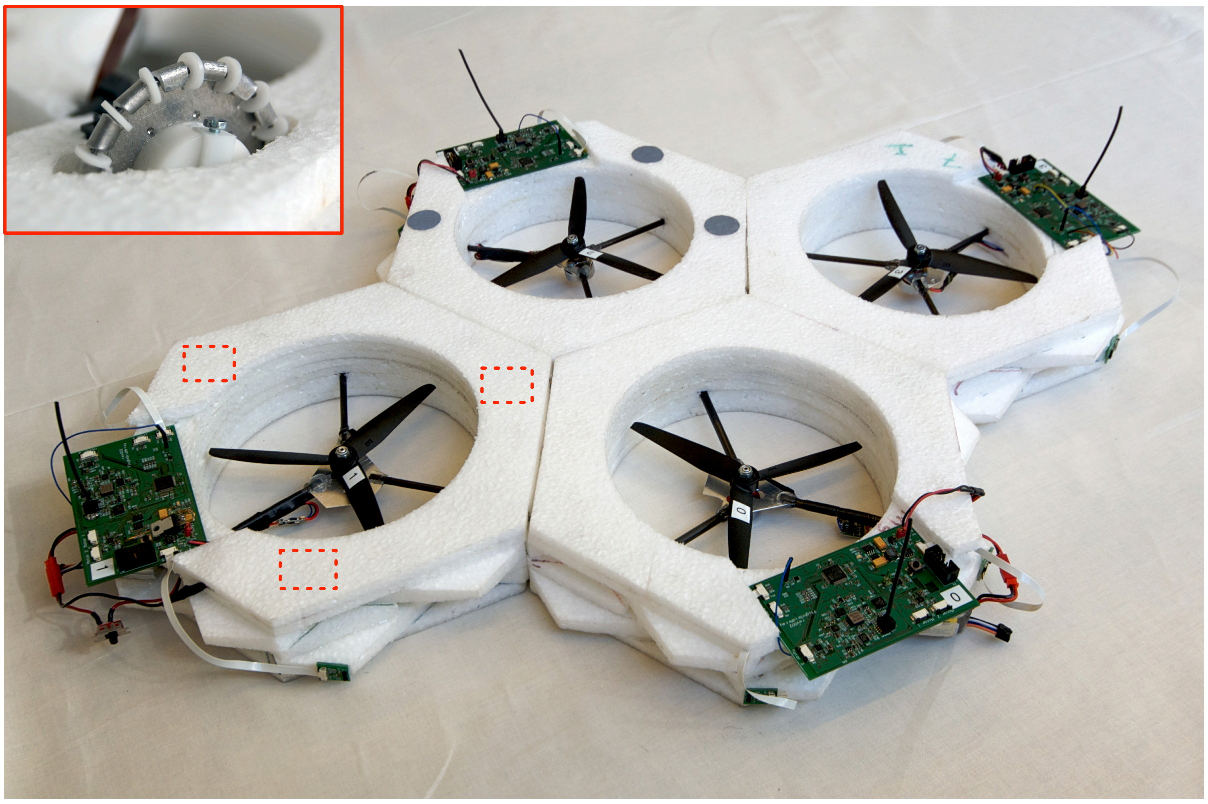


Figure 2.1 Four DFA modules are shown in a docked configuration. The module's hexagonal chassis has protruding features designed for passive alignment and docking; it is assembled from foam sheet cutouts of low-density expanded polypropylene (EPP) foam. Mounted at the center of the chassis is a brushless DC motor and a 3-blade fixed-pitch propeller. The top-left inset shows a custom-made omnidirectional wheel; three of these wheels, indicated by the dotted boxes around the chassis, are used to drive the module on the ground.

A. Chassis and Docking Mechanism

Each DFA module resembles a hexagon with protruding features designed for passive alignment and docking, see Figure 2.1. The chassis must be light enough to facilitate flight, and durable enough to repeatedly withstand a fall from at least two meters. To accomplish this, a low-density expanded polypropylene foam was chosen as the chassis material [6, 7]. Two-dimensional foam cut-outs were layered on top of one another to generate the assemblies shown in Figure 2.1 his relatively simple manufacturing process saves time and cost. However, it removes the possibility of generating a smooth leading edge for the duct which could improve thrust efficiency.

The protruding features assist with alignment and eliminate the unnecessary complexity of an active docking mechanism. A symmetric arrangement of four permanent magnets on each side of the module help to keep the modules attached. The magnets have been chosen to be strong enough to keep the modules together and to withstand the stresses of flight, but weak enough to break apart when a sufficient amount of force and/or torque is applied by a module.

Table 2.1 Physical Attributes of a DFA Module

Symbol	**Description**	**Value**
ℓ	Characteristic length[a]	0.250 m
r_w	Wheel distance[b]	0.100 m
–	Rotor duct diameter	0.180 m
m	Total Mass	0.180 kg

[a] Defined as the distance between opposite sides of a module.

[b] Defined as the lateral distance from the center of the module to the center of the wheel.

Table 2.1 lists some of the important physical attributes that characterize a DFA module.

B. Drive Unit

Custom-made omnidirectional wheels with rollers orthogonal to the axis of the wheel are mounted to three sides of the chassis, see top-left insets of Figure 2.1 0.5 Watt brushed DC motor with integrated encoder for velocity feedback drives each wheel. Omni wheels were chosen because they offer a high degree of in-plane maneuverability, and they eliminate any steering linkage(s) that would otherwise be necessary for coordinated driving, as demonstrated in Section 2.5.

C. Flight Unit

Mounted at the center of the chassis is a 50 Watt brushless DC motor with an off-the-shelf electronic speed controller and a 3-blade fixed-pitch propeller capable of producing more than 3 Newtons of thrust. Embedded in the chassis is a Lithium-Ion Polymer battery that is capable of powering both the motors and the electronics for up to 5 minutes of flight. All modules are identical except for the spin direction (or handedness) of their propeller, where there are two possibilities: clockwise (CW) and counterclockwise (CCW). This is necessary to cancel the aerodynamic torques in trimmed flight.

D. Sensing, Communication & Computation

The electronics were custom designed to meet all the on-board sensing, communication, and computation requirements. Each module comes equipped with a 3-axis rate gyro for measuring angular rates and a pressure sensor for measuring altitude. Bi-directional inter-module communication is accomplished using infrared transceivers mounted to each side of the module. An ARM7 core microcontroller handles all the required computation needed for estimation and control.

2.2 Modeling

The DFA has two distinct modes of operation: (1) driving and (2) flying. This requires two different models which are discussed in Section 2.2 and Section 2.2, respectively.

Kinematic Drive Model

Since the modules drive on the ground with relatively low velocities, dynamic effects that act on the system may be neglected. For this reason, the drive model considers only the kinematics of a rigid body with three independently-driven omni wheels.

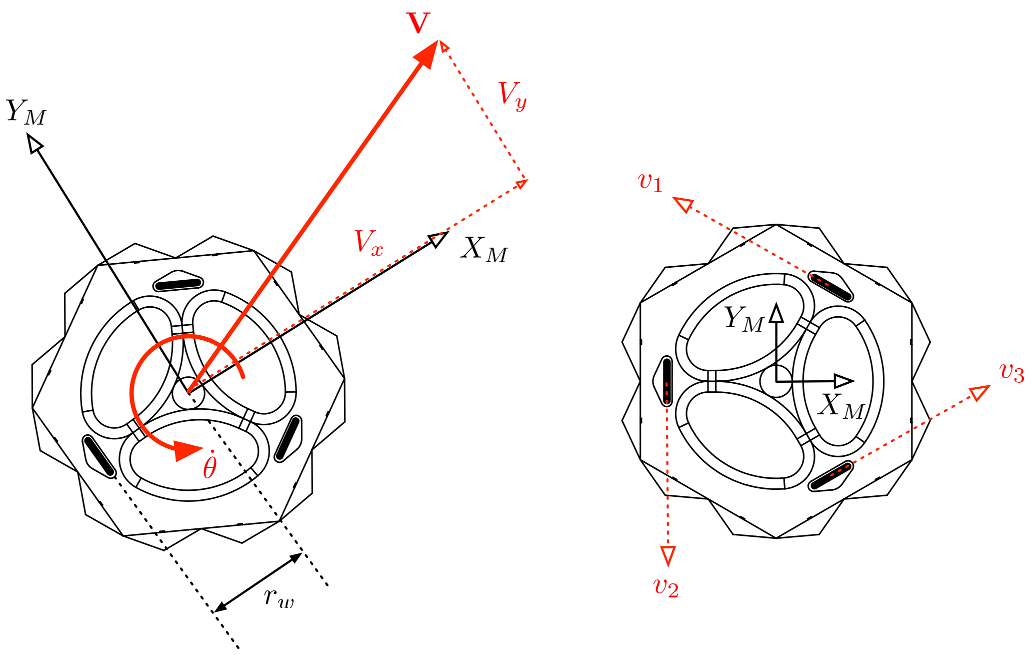


Figure 2.2 The body coordinate frame M of a DFA module coincides with the geometric center of the module and follows a right-handed coordinate system, where the x-axis is orthogonal to a side of the module and the z-axis points upward. The module is driven by three independently driven omni wheels with wheel velocities $\mathbf{v} = (v_1, v_2, v_3)$, which can be transformed to the module's velocity $\dot{\zeta} = (V_x, V_y, \dot{\theta})$.

The module's body coordinate frame M coincides with the geometric center of the module and follows a right-handed coordinate system, where the x-axis is orthogonal to a side of the module and the z-axis points upward, see Figure 2.2. The transformation matrix $\mathbf{J}$, which maps the module's velocity $\dot{\zeta} = (V_x, V_y, \dot{\theta})$ in the module's body coordinate frame to wheel velocities $\mathbf{v} = (v_1, v_2, v_3)$ for an omni wheel vehicle, has been developed in previous work and is only summarized here [8]:

$$\mathbf{v} = \mathbf{J}\dot{\zeta}, \tag{2.1}$$

where

$$\mathbf{J} = \begin{bmatrix} -\frac{\sqrt{3}}{2} & \frac{1}{2} & r_w \\ 0 & -1 & r_w \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & r_w \end{bmatrix}$$

and r_w is the lateral distance from the center of the wheel to the center of the module, see Figure 2.2.

Dynamic Flight Model

The full flight dynamics of the DFA can be quite complex if effects like the flexibility of the propellers, aerodynamic effects of the rotor duct, and the forces that keep the modules together are considered [9]–[16]. As a first step, the system is modeled as a rigid body without any compliant inter-module connections, incorporating a force and torque generation process at each module around the hovering equilibrium. This will be shown to be adequate for the purpose of hover control in Section 2.4.

Flight Dynamics The DFA's body coordinate frame B coincides with the array's center of mass and is oriented along the principal axes of rotation. Module i is located at coordinates $(\mathrm{x}_i, \mathrm{y}_i)$ with respect to this body coordinate frame. A sequence of three rotations described by Euler angles α, β, γ acting along the z-, y-, x-axis in this order describe the orientation of the DFA's body coordinate frame with respect to the inertial coordinate frame O.

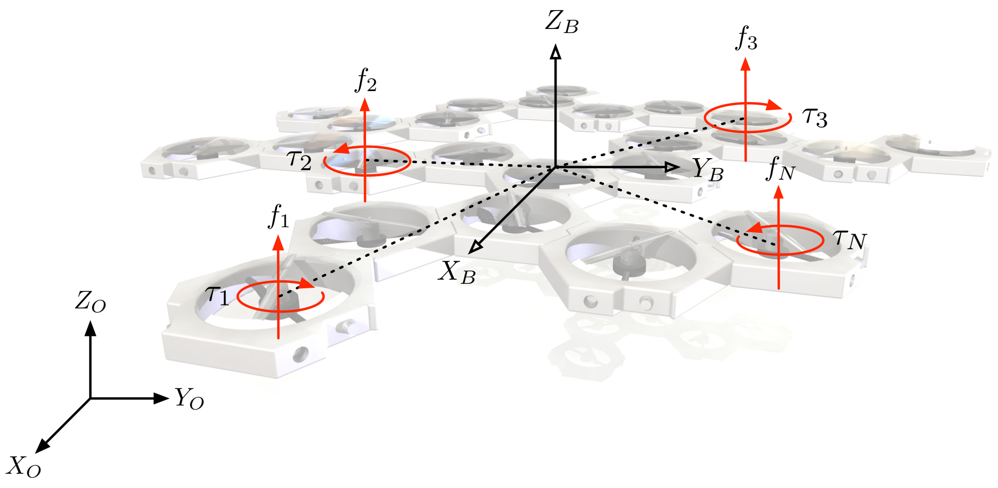


Figure 2.3 The body coordinate frame B of the array is chosen to coincide with the array's center of mass and is aligned with the principal axes of rotation. The altitude and attitude of the DFA can be controlled by varying the force f_i and torque τ_i produced by each module.

The altitude and attitude of the DFA can be controlled by varying the force (or thrust) f_i and torque τ_i produced by each module, see Figure 2.3; how these control inputs are generated will be described later. The total thrust generated by N modules is the sum of all thrusts produced by each module, $F = \sum_{i=1}^{N} f_i$. The rolling torque is the sum of all thrusts acting along the moment arm y_i, $T_\gamma = \sum_{i=1}^{N} \mathrm{y}_i f_i$. Similarly, the pitching torque is the sum of all thrusts acting along the moment arm x_i, $T_\beta = -\sum_{i=1}^{N} \mathrm{x}_i f_i$. The yawing torque is the sum of all reaction torques produced by each module; it will be shown that the torque can be accurately modeled as a linear function of thrust. Hence, the yawing torque can be expressed as $T_\alpha = \sum_{i=1}^{N} c_i f_i$, where the sign of c_i depends on the propeller's handedness.

Transforming the total thrust vector F to the inertial coordinate frame results in a translational component of force along each axis. The translational accelerations $\ddot{x}$ and $\ddot{y}$ in the inertial coordinate frame is a consequence of a pitch β and roll γ rotation, respectively. Since the system is being modeled around the equilibrium, small angles are used to approximate the rotation in roll γ and pitch β. However, the rotation in yaw α is not assumed to be small. The following set of equations summarize the dynamics of the array to first-order, except for the yaw angle α:

$$Nm\ddot{x} = (\beta \cos\alpha + \gamma \sin\alpha) \sum_{i=1}^{N} f_i, \tag{2.2}$$

$$Nm\ddot{y} = (\beta \sin\alpha - \gamma \cos\alpha) \sum_{i=1}^{N} f_i, \tag{2.3}$$

$$Nm\ddot{z} = \sum_{i=1}^{N} f_i - Nmg, \tag{2.4}$$

$$I_x\ddot{\gamma} = \sum_{i=1}^{N} \mathrm{y}_i f_i, \tag{2.5}$$

$$I_y\ddot{\beta} = -\sum_{i=1}^{N} \mathrm{x}_i f_i, \tag{2.6}$$

$$I_z\ddot{\alpha} = \sum_{i=1}^{N} c_i f_i, \tag{2.7}$$

where Nm is the total mass of the array, (I_x, I_y, I_z) are the principal mass moments of inertia, and g is the acceleration constant due to gravity.

Assuming that the array configuration is relatively circular and letting ℓ denote the characteristic length of a module, then $\ell\sqrt{N}/2$ is comparable to the radius of the array. It follows that the principal mass moments of inertia can be written as the following:

$$I_x = \epsilon_x \frac{Nm}{4}\left(\frac{\ell\sqrt{N}}{2}\right)^2, \tag{2.8}$$

$$I_y = \epsilon_y \frac{Nm}{4}\left(\frac{\ell\sqrt{N}}{2}\right)^2, \tag{2.9}$$

$$I_z = \epsilon_z \frac{Nm}{2}\left(\frac{\ell\sqrt{N}}{2}\right)^2, \tag{2.10}$$

where $(\epsilon_x, \epsilon_y, \epsilon_z)$ capture the mass distribution of the array. If the configuration of the array is full, like a disk, $(\epsilon_x, \epsilon_y, \epsilon_z)$ are expected to be close to 1. In reality, the configuration of the array may be sparse due to unpredictable docking constraints. In such a case, the values of $(\epsilon_x, \epsilon_y, \epsilon_z)$ are expected to be greater than 1.

The equations of motion are normalized using the following variables in order to gain some intuition on how the size of the array affects flight dynamics:

$$\hat{\mathrm{x}}_i = \frac{\mathrm{x}_i}{\frac{\ell\sqrt{N}}{2}}, \quad \hat{\mathrm{y}}_i = \frac{\mathrm{y}_i}{\frac{\ell\sqrt{N}}{2}}, \quad \hat{c}_i = \frac{c_i}{\ell}, \quad A_i = \frac{f_i}{m}, \tag{2.11}$$

where $\hat{\mathrm{x}}_i$ and $\hat{\mathrm{y}}_i$ are normalized position coordinates and are at most on the order of 1 for a circular array, $\hat{c}_i$ is the normalized force to torque conversion constant and is expected to be much less than 1, and A_i is the normalized thrust in units of acceleration.

The normalized thrust A_i can be broken down into its components, $A_i = \bar{A}_i + a_i$, where $\bar{A}_i$ is the normalized thrust required to establish the hovering equilibrium and a_i is the normalized control input. In the special case where there are an equal number of CW and CCW modules in the array, one can set $\bar{A}_i = g$. In general, however, the system may be overactuated with an unequal number of CW and CCW modules. In this case, one can choose the values of $\bar{A}_i$ via least squares, or any other suitable method.

The following set of equations summarize the normalized and linearized equations of motion about hovering equilibrium:

$$\ddot{x} = \frac{1}{N}(\beta\cos\alpha + \gamma\sin\alpha)\sum_{i=1}^{N} a_i, \tag{2.12}$$

$$\ddot{y} = \frac{1}{N}(\beta\sin\alpha - \gamma\cos\alpha)\sum_{i=1}^{N} a_i, \tag{2.13}$$

$$\ddot{z} = \frac{1}{N}\sum_{i=1}^{N} a_i, \tag{2.14}$$

$$\hat{I}_x\ddot{\gamma} = \frac{1}{N}\sum_{i=1}^{N} \hat{\mathrm{y}}_i a_i, \tag{2.15}$$

$$\hat{I}_y\ddot{\beta} = -\frac{1}{N}\sum_{i=1}^{N} \hat{\mathrm{x}}_i a_i, \tag{2.16}$$

$$\hat{I}_z\ddot{\alpha} = \frac{1}{N}\sum_{i=1}^{N} \hat{c}_i a_i, \tag{2.17}$$

where

$$\hat{I}_x = \frac{\epsilon_x \ell \sqrt{N}}{8}, \quad \hat{I}_y = \frac{\epsilon_y \ell \sqrt{N}}{8}, \quad \hat{I}_z = \frac{\epsilon_z \ell N}{8}. \tag{2.18}$$

It can be seen from (2.14) that the maximum vertical acceleration $\ddot{z}$ is independent of N. On the other hand, by substituting (2.18) into (2.15) – (2.17) and rearranging, it can be shown that the maximum accelerations in roll $\ddot{\gamma}$ and in pitch $\ddot{\beta}$ decrease by a factor of $\sqrt{N}$, while the maximum acceleration in yaw $\ddot{\alpha}$ decreases by a factor of N.

Disregarding the x and y translational components of motion, the DFA requires at least four modules to hover; four is the minimum number of control inputs a_i needed to control the four remaining degrees of freedom (2.14) – (2.17). This, however, is only a necessary condition and not a sufficient condition to keep the array hovering. For example, four modules lined together in a row would not be linearly stable in attitude; the system would either be controllable in roll and not in pitch, or vice versa.

Force and Torque Generation The module's force-torque generation unit (i.e. motor controller, brushless DC motor, battery, propeller, and rotor duct) is treated as a black box model. The input to this system is a pulse-width modulation (PWM) duty cycle D that effectively controls the angular velocity of the rotor. Battery voltage can also be considered as an input to the system, however the effects of voltage on the dynamics of the system are ignored since the nominal voltage of Lithium-Ion Polymer batteries is relatively constant over the battery cycle [17, 18]. The force-torque characteristics that are described here assume the nominal voltage case around the equilibrium thrust. The output from this system is both the generated force and torque of the module. Torque is a result of the imparted rotational flow and change in angular momentum of the rotor.

Force and torque measurements were made at various duty cycles around the equilibrium thrust. The force experienced by a module is dominated by the thrust of the rotor. Experimental results show that thrust and duty cycle can be approximated by an affine relationship, while the torque resulting from the rotor's drag can be approximated as a linear function of thrust, see Figure 2.4. Recall that this thrust-torque relationship was used in (2.7) and is expressed in the following:

$$\tau_i = cf_i, \tag{2.19}$$

where $c = \pm 1.13 \times 10^{-2}\,\text{m}$; recall that the sign of c depends on the the propeller's handedness. Other experiments demonstrated that the effect of torque produced by the rate of change in angular momentum of the rotor to be negligible.

Motivated by the results in Figure 2.4, the transfer function G_f which relates the input desired thrust $f(D)$ to the output thrust was modeled as a linear time-invariant system. The transfer function that was obtained from the Bode plot of the thrust response, shown in Figure 2.5, was found to approximate a first-order system:

$$G_f(s) = \frac{\omega}{s+\omega}, \tag{2.20}$$

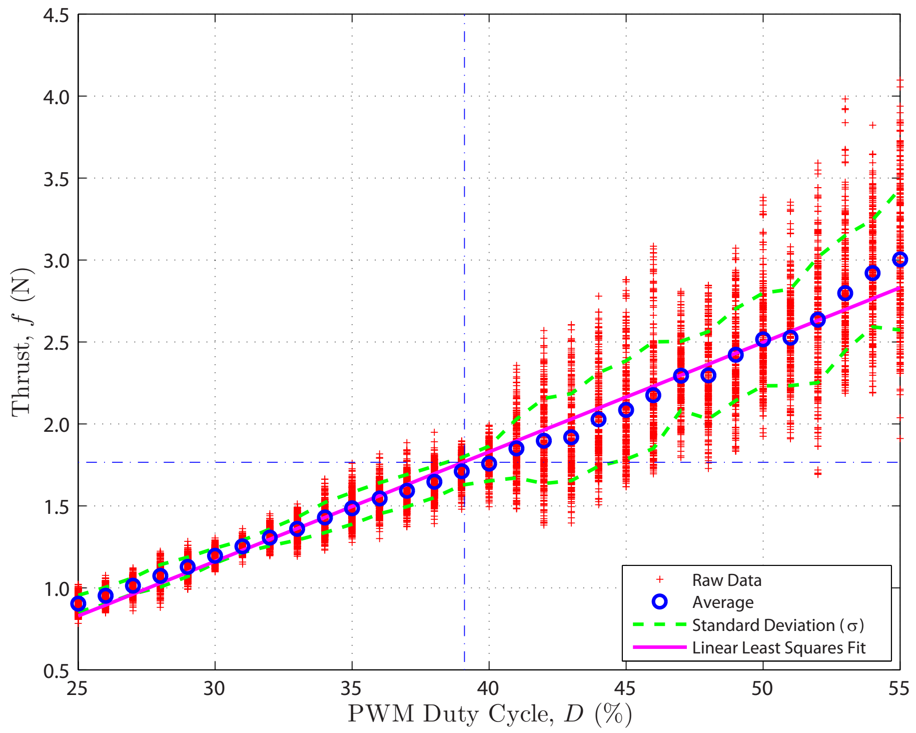
where $\omega = 14.3\,\text{rad/s}$.

2.3 Driving Strategy

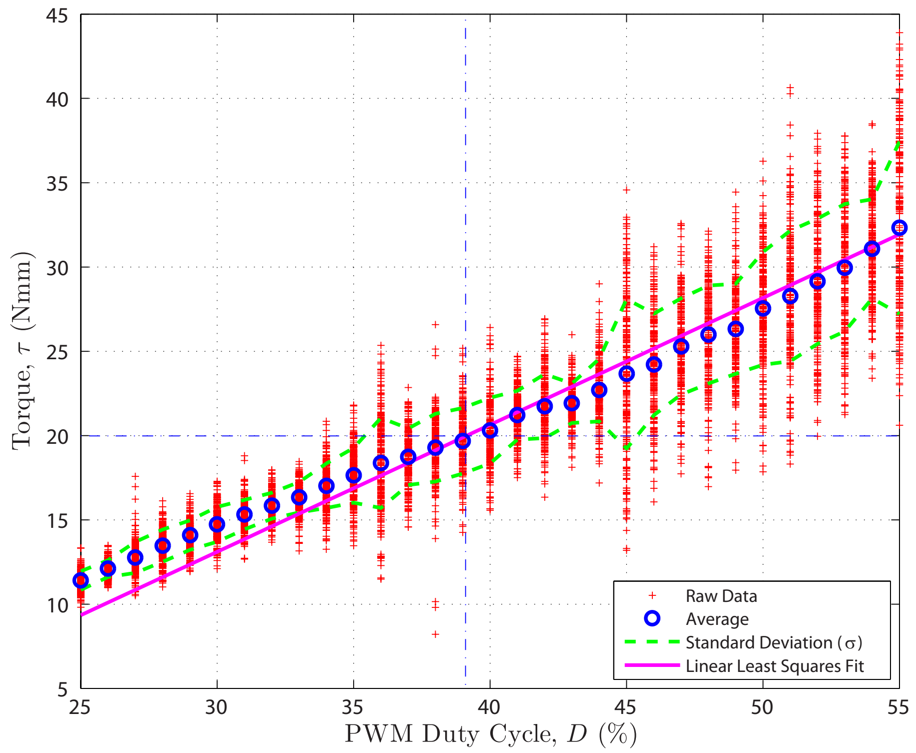
At the start of each cycle, a number of DFA modules will be scattered randomly across the floor. Without any prior knowledge about the environment, the module's objective will be to dock with its peers in preparation for coordinated flight. Although the assembly process could be controlled to achieve a particular array configuration, a decision was made to let the self-assembly process occur at random with the intent that a new configuration is flown each time. The docking strategy summarized here is one of many possibilities that increases the likelihood of modules finding one another.

Consider an overhead light source as the only environmental feature detectable by each module via a single on-board photodiode. This photodiode along with encoder odometry will be the only source of information used for pose estimation. Modules will be able to move around a prescribed circular region as defined by the cone of illumination from the overhead light source, see Figure 2.6. A user-defined light intensity threshold will force a module into one of two states: 1) within the circular region and 2) outside the circular region.

The goal of each module will be to drive towards the center of the circular region and arbitrarily dock with its peers. Modules will initially drive in random directions. Once a module crosses the boundary of the circular region at three different locations, it will be able to use these three data points to estimate the center of the circle and to drive towards it. Modules may randomly connect during the data point collection



(a) A linear least squares fit of the data relating thrust f to PWM duty cycle D results in the function $f(D) = 6.68 \times 10^{-2}D - 8.41 \times 10^{-1}$.



(b) A linear least squares fit to the function $cf(D)$ of the data relating torque τ to PWM duty cycle D results in the function $\tau(D) = 1.13 \times 10^{-2} f(D)$.

Figure 2.4 Force and torque values were measured with a 6-axis force-torque strain gauge load cell at various PWM duty cycles near the hovering equilibrium (approximately D = 39%), and is denoted by the pair of dashed-dotted horizontal and vertical lines. At each duty cycle, 500 measurements were made over a period of 5 seconds.

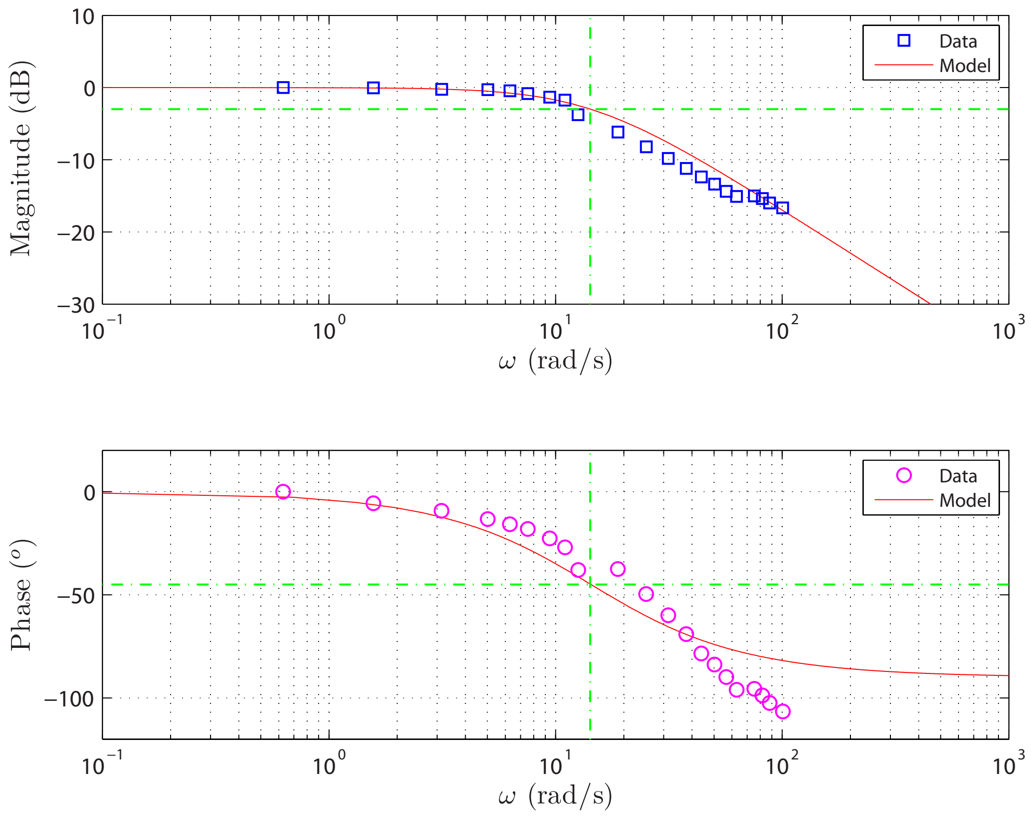


Figure 2.5 The Bode plot of the flight motor thrust response is shown here to approximate a first-order system. A sinusoidal varying PWM duty cycle, with an offset equivalent to the hovering thrust and an amplitude equivalent to 0.5 N, was commanded to the system. Force and torque measurements were made using a 6-axis force-torque strain gauge load cell. The dash-dotted lines in the plots indicate the magnitude and phase at -3 dB frequency, which is 14.3 rad/s. Note that at high frequencies the measured phase diverges from the model; this is due to unmodeled dynamics, such as delays.

process, during which they may share information to improve their common estimate of the circular region [19, 20]. Depending on their estimate, they will coordinate to either continue exploration, or to move towards the center.

A 2D simulator using the Box2D physics engine [21] is currently being developed to experiment with this and other docking strategies, see Figure 2.7. It will be used to obtain array configuration statistics, which will be needed to determine the typical mass distribution parameter values (ϵ_x, ϵ_y, ϵ_z) required for testing various control strategies and analyzing their performance.

2.4 Flight Control

This section presents an easily tunable strategy for hover control based on physical parameters of the DFA and compares its performance and tradeoffs to that of an $\mathcal{H}_2$ controller. The derivation of this control strategy is generalized, and assumes full state feedback

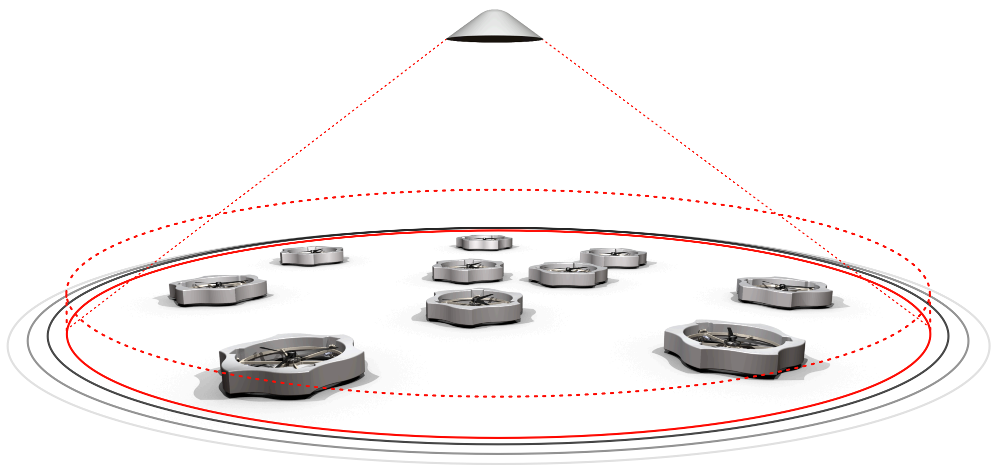


Figure 2.6 The DFA modules may be randomly scattered across a circular area defined by the cone of illumination from an overhead light source. The perimeter of this circular region can be adjusted by varying the light intensity, the height of the lamp, the aperture of the light source, and/or the intensity threshold of the module's photodiode. The modules can use this light source as a means of localization, which can in turn assist with self-assembly.

of the system. It is assumed that an estimator is used to obtain the state of the system [22, 23].

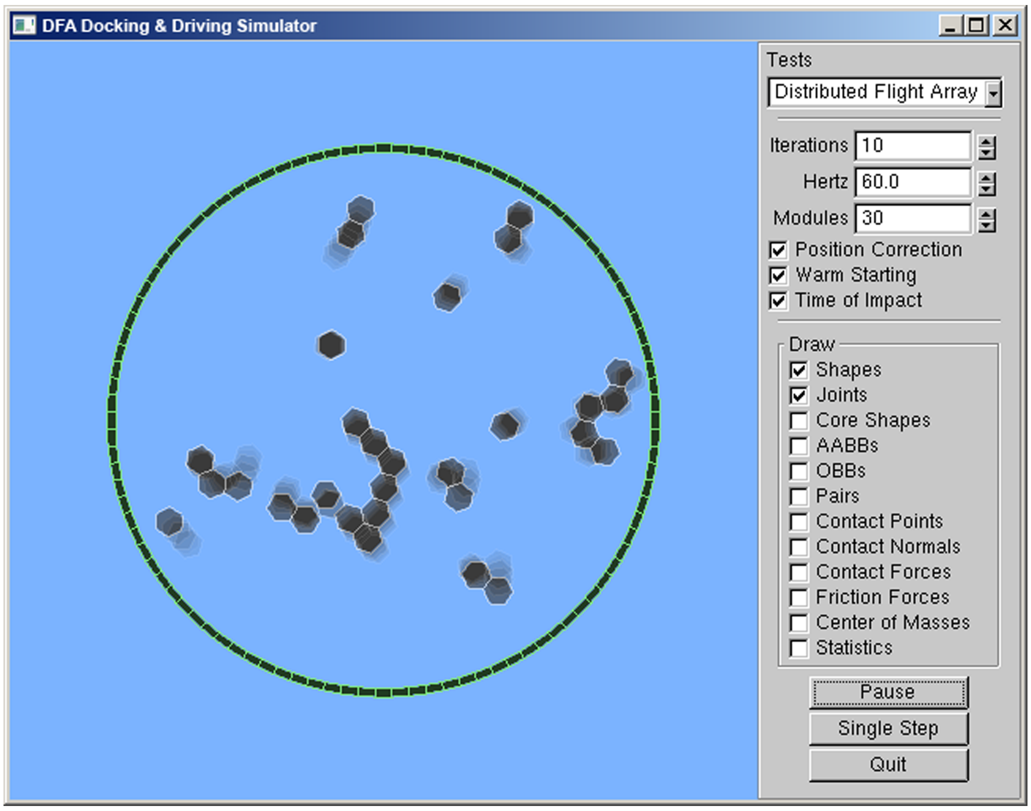
Control Strategy

Starting with the dynamic model of the DFA that was developed in the previous section, and ignoring the degrees of freedom along the x and y axes, the normalized and linearized equations of motion about the hovering equilibrium (2.14) –(2.17) can be more compactly expressed as

$$\mathbf{M}\ddot{\mathbf{s}} = \mathbf{P}^T\mathbf{a}, \tag{2.21}$$

where

$$\begin{aligned}\mathbf{M} &= \mathrm{diag}(1, \hat{I}_x, \hat{I}_y, \hat{I}_z),\\ \mathbf{s} &= [z, \gamma, \beta, \alpha]^T,\\ \mathbf{a} &= [a_1, \ldots, a_N]^T.\end{aligned}$$



Tests: Distributed Flight Array

Iterations: 10

Hertz: 60.0

Modules: 30

- ☑ Position Correction
- ☑ Warm Starting
- ☑ Time of Impact

Draw

- ☑ Shapes
- ☑ Joints
- ☐ Core Shapes
- ☐ AABBs
- ☐ OBBs
- ☐ Pairs
- ☐ Contact Points
- ☐ Contact Normals
- ☐ Contact Forces
- ☐ Friction Forces
- ☐ Center of Masses
- ☐ Statistics

Figure 2.7 A docking and driving simulator is being developed using the Box2D physics engine. The simulator will be used to obtain array configuration statistics, which will be needed for testing various control strategies and analyzing their performance.

The matrix $\mathbf{P} \in \mathbb{R}^{N\times 4}$ contains information pertaining to the configuration of the array and can be written as

$$\mathbf{P} = [\mathbf{p}_z, \mathbf{p}_\gamma, \mathbf{p}_\beta, \mathbf{p}_\alpha], \tag{2.22}$$

where

$$\mathbf{p}_z = \frac{1}{N}[1, \ldots, 1]^T, \qquad \mathbf{p}_\gamma = \frac{1}{N}[\hat{\mathrm{y}}_1, \ldots, \hat{\mathrm{y}}_N]^T,$$
$$\mathbf{p}_\beta = -\frac{1}{N}[\hat{\mathrm{x}}_1, \ldots, \hat{\mathrm{x}}_N]^T, \qquad \mathbf{p}_\alpha = \frac{1}{N}[\hat{c}_1, \ldots, \hat{c}_N]^T.$$

Consider the control strategy of the following form:

$$\mathbf{a} = \mathbf{Q} f(z, \dot{z}, \gamma, \dot{\gamma}, \beta, \dot{\beta}, \alpha, \dot{\alpha}), \tag{2.23}$$

where

$$\begin{aligned}\mathbf{Q} &= [\mathbf{q}_z, \mathbf{q}_\gamma, \mathbf{q}_\beta, \mathbf{q}_\alpha],\\ \mathbf{f} &= [f_z(z,\dot{z}), f_\gamma(\gamma,\dot{\gamma}), f_\beta(\beta,\dot{\beta}), f_\alpha(\alpha,\dot{\alpha})]^T,\end{aligned}$$

and where the $f_\cdot(\,\cdot\,)$ are arbitrary functions to be determined.

The objective is to decouple the degrees of freedom. This can be accomplished by designing $\mathbf{Q} \in \mathbb{R}^{N\times 4}$ such that $\mathbf{P}^T\mathbf{Q} = \mathbf{I}_4$, resulting in the following diagonal system:

$$\ddot{z} = f_z(z,\dot{z}), \tag{2.24}$$
$$\hat{I}_x\ddot{\gamma} = f_\gamma(\gamma,\dot{\gamma}), \tag{2.25}$$
$$\hat{I}_y\ddot{\beta} = f_\beta(\beta,\dot{\beta}), \tag{2.26}$$
$$\hat{I}_z\ddot{\alpha} = f_\alpha(\alpha,\dot{\alpha}). \tag{2.27}$$

Notice that the system is over-determined for an array with more than four modules.

Due to the coordinate system described in Section 2.2, $\mathbf{p}_z$, $\mathbf{p}_\gamma$, and $\mathbf{p}_\beta$ are orthogonal. If there are an equal number of CW and CCW modules, then $\mathbf{p}_z$ and $\mathbf{p}_\alpha$ are also orthogonal. In what follows, consider an equal number of CW and CCW modules; the results can readily be generalized.

Let $\mathbf{Q} = \mathbf{P}\mathbf{D}\overline{\mathbf{Q}}$, where

$$\mathbf{D} = \mathrm{diag}(\frac{1}{||\mathbf{p}_z||^2}, \frac{1}{||\mathbf{p}_\gamma||^2}, \frac{1}{||\mathbf{p}_\beta||^2}, \frac{1}{||\mathbf{p}_\alpha||^2})$$

and $||\cdot||$ is the Euclidean norm.

Multiplying $\mathbf{P}^T$ on both sides yields:

$$\begin{aligned}\mathbf{I}_4 &= \mathbf{P}^T\mathbf{Q} = \mathbf{P}^T\mathbf{P}\mathbf{D}\overline{\mathbf{Q}}\\ &= \begin{bmatrix} ||\mathbf{p}_z||^2 & 0 & 0 & 0\\ 0 & ||\mathbf{p}_\gamma||^2 & 0 & \mathbf{p}_\gamma^T\mathbf{p}_\alpha\\ 0 & 0 & ||\mathbf{p}_\beta||^2 & \mathbf{p}_\beta^T\mathbf{p}_\alpha\\ 0 & \mathbf{p}_\alpha^T\mathbf{p}_\gamma & \mathbf{p}_\alpha^T\mathbf{p}_\beta & ||\mathbf{p}_\alpha||^2 \end{bmatrix}\mathbf{D}\overline{\mathbf{Q}}\\ &= \begin{bmatrix} 1 & 0 & 0 & 0\\ 0 & 1 & 0 & e_1\\ 0 & 0 & 1 & e_2\\ 0 & e_3 & e_4 & 1 \end{bmatrix}\overline{\mathbf{Q}},\end{aligned} \tag{2.28}$$

where

$$e_1 = \frac{\mathbf{p}_\gamma^T \mathbf{p}_\alpha}{\|\mathbf{p}_\alpha\|^2}, \qquad e_2 = \frac{\mathbf{p}_\beta^T \mathbf{p}_\alpha}{\|\mathbf{p}_\alpha\|^2}, \qquad e_3 = \frac{\mathbf{p}_\alpha^T \mathbf{p}_\gamma}{\|\mathbf{p}_\gamma\|^2}, \qquad e_4 = \frac{\mathbf{p}_\alpha^T \mathbf{p}_\beta}{\|\mathbf{p}_\beta\|^2}.$$

It can readily be shown that

$$\overline{\mathbf{Q}} = \begin{bmatrix} 1 & 0 \\ 0 & \overline{\overline{\mathbf{Q}}} \end{bmatrix}, \tag{2.29}$$

where

$$\overline{\overline{\mathbf{Q}}} = \frac{1}{1-(e_2e_4+e_3e_1)} \begin{bmatrix} 1-e_2e_4 & e_1e_4 & -e_1 \\ e_2e_3 & 1-e_3e_1 & -e_2 \\ -e_3 & -e_4 & 1 \end{bmatrix}.$$

In the special case where there are a large number of CW and CCW modules that are uniformly distributed in the array, $\mathbf{p}_\gamma$ and $\mathbf{p}_\beta$ are approximately orthogonal to $\mathbf{p}_\alpha$. This can be made mathematically precise, but the intuition is straightforward: a roll or pitch action employs roughly the same number of CW and CCW modules, and thus the net yawing torque is zero. The result of having a large number of modules N in the array is that $e_i \to 0$, which results in $\overline{\overline{\mathbf{Q}}} \to \mathbf{I}_3$ and $\mathbf{Q} \to \mathbf{PD}$. One could then use the following decoupling strategy:

$$\mathbf{q}_z = \frac{\mathbf{p}_z}{\|\mathbf{p}_z\|^2}, \qquad \mathbf{q}_\gamma = \frac{\mathbf{p}_\gamma}{\|\mathbf{p}_\gamma\|^2}, \qquad \mathbf{q}_\beta = \frac{\mathbf{p}_\beta}{\|\mathbf{p}_\beta\|^2} \qquad \mathbf{q}_\alpha = \frac{\mathbf{p}_\alpha}{\|\mathbf{p}_\alpha\|^2}.$$

It can be shown that the above elements of $\mathbf{Q}$ do not scale with N, and thus this decoupling strategy is independent of N. Moreover, this strategy has the desirable property of minimizing the inter-module shear stresses in the array resulting from pitch and roll errors. This can readily be seen by substituting $\mathbf{Q}$ into the control strategy (2.23) and analyzing the resulting expression; the control input a_i increases linearly the further away a module is from the center of mass. As a result, the shear stress acting between two modules is minimized. This is an important feature because too much shear stress would cause the module(s) to break away from the array.

This control strategy only works if both $\mathbf{P}$ and $(\hat{I}_x,\ \hat{I}_y,\ \hat{I}_z)$ are known. Assuming that all modules are identical and that the mass and mass moments of inertia are given, then both of these parameters can be computed if the position and the propeller's handedness for each module is known. It follows that this is the only information that needs to be communicated across the array before taking flight.

Now that the control strategy has been decoupled, one can consider each degree of freedom separately. For example, the following functions can be chosen:

$$f_z(z, \dot{z}) = -2\omega_z\zeta_z\dot{z} - \omega_z^2(z - z_d), \tag{2.30}$$

$$f_\gamma(\gamma, \dot{\gamma}) = -\hat{I}_x(2\omega_\gamma\zeta_\gamma\dot{\gamma} + \omega_\gamma^2\gamma), \tag{2.31}$$

$$f_\beta(\beta, \dot{\beta}) = -\hat{I}_y(2\omega_\beta\zeta_\beta\dot{\beta} + \omega_\beta^2\beta), \tag{2.32}$$

$$f_\alpha(\alpha, \dot{\alpha}) = -\hat{I}_z(2\omega_\alpha\zeta_\alpha\dot{\alpha} + \omega_\alpha^2\alpha), \tag{2.33}$$

where z_d represents the desired hovering altitude and each degree of freedom is a second-order system with two sets of tuning parameters: 1) the natural frequencies (ω_z, ω_γ, ω_β, ω_α), and 2) the damping ratios (ζ_z, ζ_γ, ζ_β, ζ_α). The tuning of these two sets of parameters will depend on the DFA's mass moment of inertia, which is affected by the size and configuration of the array. Recall that the effect due to the size of the array N is more pronounced in the yaw degree of freedom than in roll and/or pitch by a factor of $\sqrt{N}$, see Section 2.2.

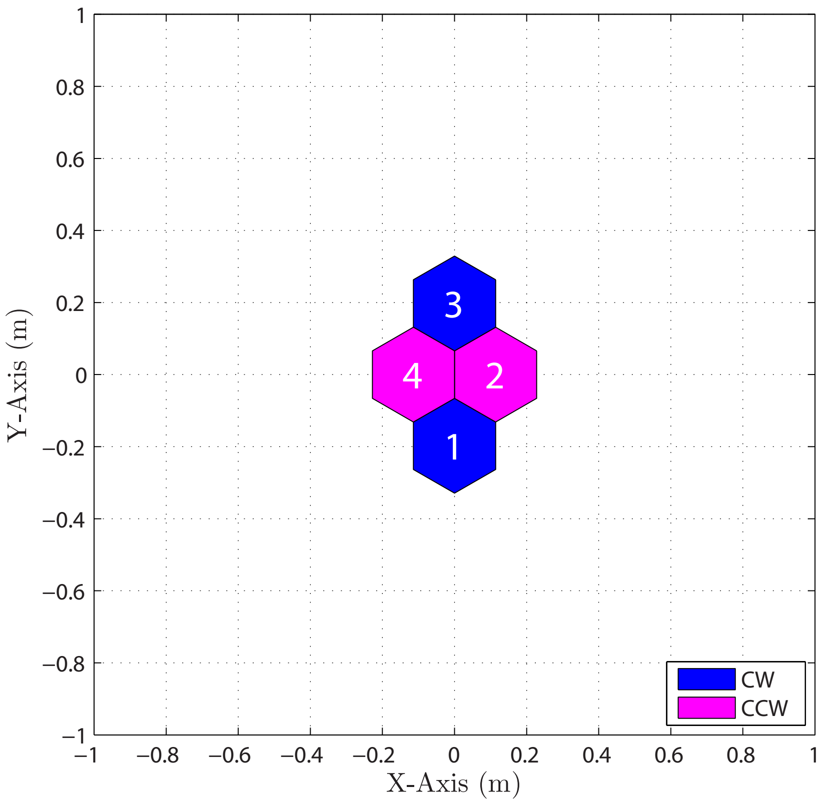
Be aware that this control strategy uses normalized thrust as the control input, which is in fact indirectly generated by the DFA module as described in Section 2.2. Thrust dynamics and saturation of the control inputs should be considered. Time-scale separation is needed between the desired dynamics of the system and the rotor dynamics. A way to achieve this is to invert the transfer function G_f over a desired frequency range, enough to achieve time-scale separation.

Simulation

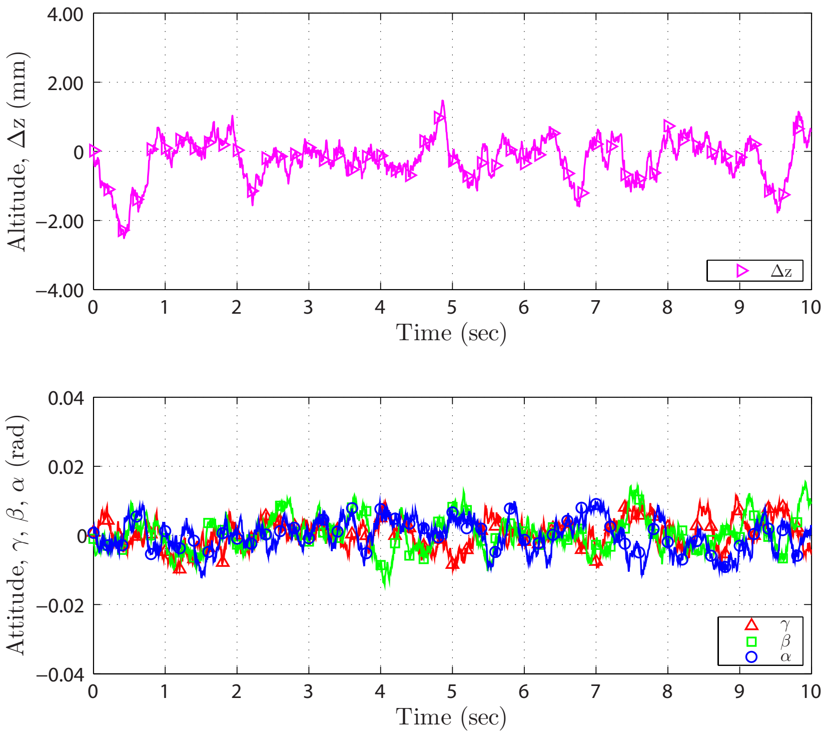
The control strategy described here has been simulated in MATLAB for random array configurations consisting of 4 to 20 modules, see Figure 2.8 – 2.9. The state of the array is obtained locally from the onboard sensors of each module. That is to say that no information is passed between modules, resulting in a completely decentralized control scheme; altitude is obtained from the pressure sensor, angular rates are measured using the rate-gyros, and these are integrated to obtain the Euler angles of the array in flight. Simulations take into account sensor and process noise derived from physical experiments and the motor model described in Section 2.2.

Performance

A quantitative comparison was made between the easily tunable control strategy described previously and an $\mathcal{H}_2$ controller. A generalized plant $G(h)$, which includes the plant model based on the configuration of the array seen in Figure 2.9(a), sensor and process noise derived from physical experiments, and weights h on the states $(z, \gamma, \beta, \alpha)$

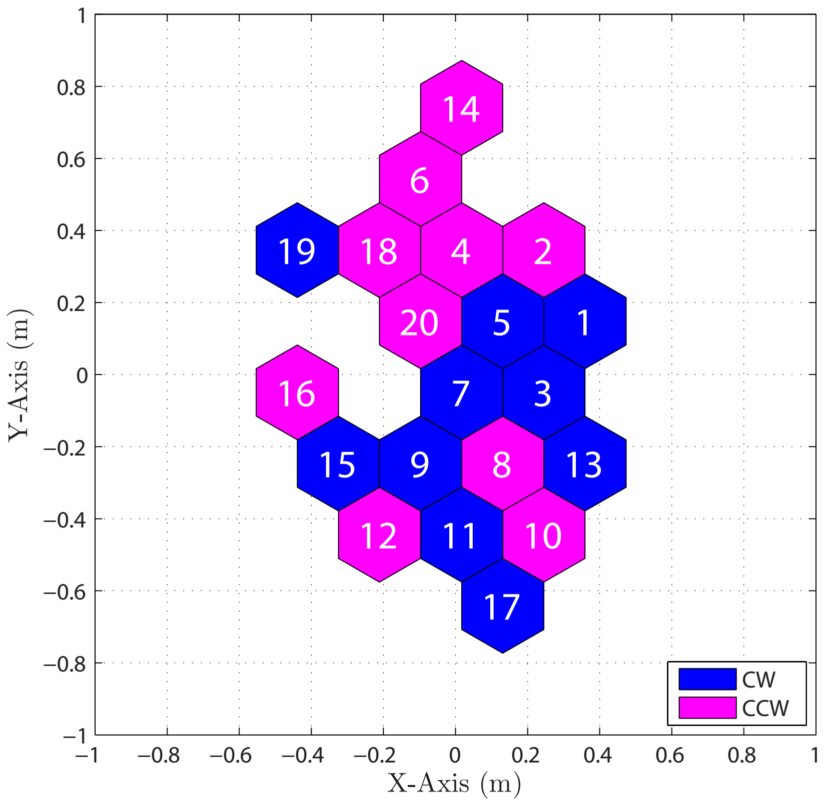


(a) 4-Module Array Configuration

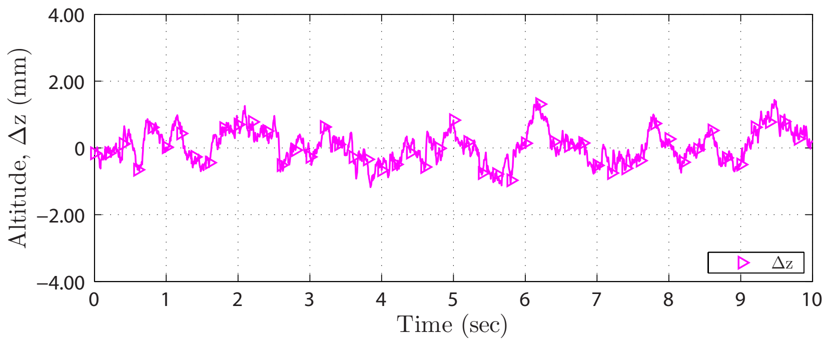


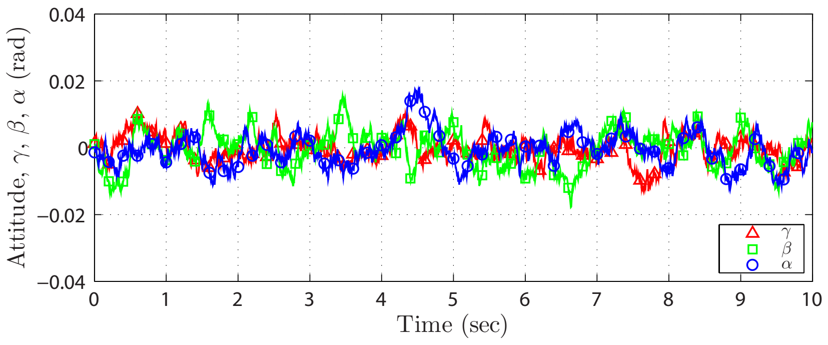
(b) Simulation Results around the Hovering Equilibrium using a Decentralized Controller.

Figure 2.8 Simulation results around the hovering equilibrium (b) using a decentralized controller on a 4-module array configuration (a), where the natural frequencies were set to $(\omega_z, \omega_\gamma, \omega_\beta, \omega_\alpha) = (9.016, 9.021, 9.021, 9.016)$ rad/s and the damping ratios were set to $(\zeta_z, \zeta_\gamma, \zeta_\beta, \zeta_\alpha) = (1, 1, 1, 1)$. An $\mathcal{H}_2$ norm of 5.1 was obtained for this particular scenario.



(a) 20-Module Array Configuration

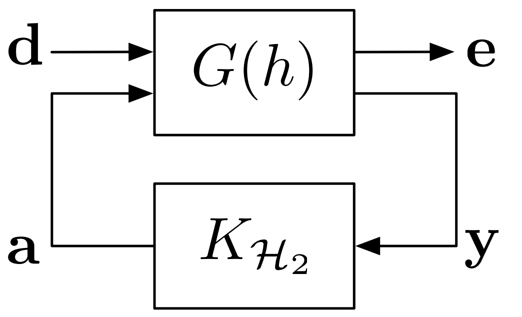




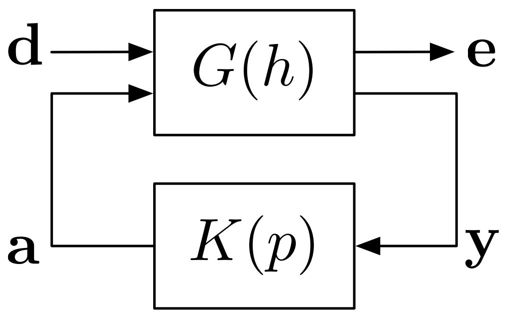
(b) Simulation Results around the Hovering Equilibrium using a Decentralized Controller.

Figure 2.9 Simulation results around the hovering equilibrium (b) using a decentralized controller on a 20-module array configuration (a), where the natural frequencies were set to $(\omega_z, \omega_\gamma, \omega_\beta, \omega_\alpha) = (9.012, 9.020, 9.020, 9.013)$ rad/s and the damping ratios were set to $(\zeta_z, \zeta_\gamma, \zeta_\beta, \zeta_\alpha) = (1, 1, 1, 1)$. An $\mathcal{H}_2$ norm of 5.529 was obtained for this particular scenario.

and on the normalized control input $\mathbf{a}$ were used for the purpose of this comparison, see Figure 11. The $\mathcal{H}_2$ optimal control problem is to find a stabilizing controller $K_{\mathcal{H}_2}$ that minimizes the $\mathcal{H}_2$ norm (i.e. the expected root mean square value of the output $\mathbf{e}$ in response to white noise excitation [24]) of the closed loop system. The optimal $\mathcal{H}_2$ controller was synthesized in MATLAB by setting the state weights $(h_z, h_\gamma, h_\beta, h_\alpha)$ to a reasonable set of constant values and adjusting the normalized control input weight h_a subject to not saturating the thrusts produced by the rotors. By doing this, an $\mathcal{H}_2$ norm of 5.385 was computed. See Figure 12 for a simulation of the output states of the $\mathcal{H}_2$ controller around the hovering equilibrium.



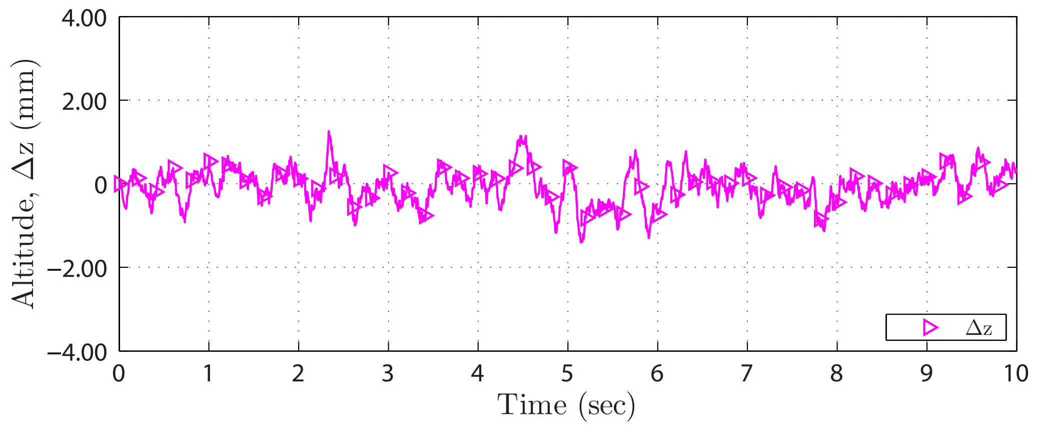
(a) Block Diagram of the System with $K_{\mathcal{H}_2}$ Feedback.



(b) Block Diagram of the System with $K(p)$ Feedback.

Figure 2.10 Block diagrams representing the generalized plant $G(h)$ with feedback from (a) the $\mathcal{H}_2$ controller, $K_{\mathcal{H}_2}$, and (b) the controller described in Section 5.1, $K(p)$. The signals for these systems are: the normalized control input $\mathbf{a} = [a_1, \ldots, a_N]^T$ in units of acceleration; the disturbances $\mathbf{d} = [d_1 \ldots, d_N]^T$, where d_i is a vector of process and measurement noises affecting module i; the error signals $\mathbf{e} = [z, h_\gamma \gamma, h_\beta \beta, h_\alpha \alpha, h_a \mathbf{a}]^T$, which are the weighted states and control input that are to be minimized in the $\mathcal{H}_2$ sense; and the states of the array $(z, \gamma, \beta, \alpha, \dot{z}, \dot{\gamma}, \dot{\beta}, \dot{\alpha})$ corrupted by noise $\mathbf{y}$.

The same generalized plant $G(h)$ and weights h were used with the easy to tune controller $K(p)$, see Figure 2.10(b). Recall from (2.30) – (2.33) that the control tuning parameters p are the natural frequencies $(\omega_z, \omega_\gamma, \omega_\beta, \omega_\alpha)$ and the damping ratios $(\zeta_z, \zeta_\gamma, \zeta_\beta, \zeta_\alpha)$. For this comparison, the damping ratios were fixed to a constant value of 1 and the natural frequencies were adjusted to minimize the $\mathcal{H}_2$ norm via a gradient descent method; this resulted in a norm of 5.529. Even with just four tuning parameters $(\omega_z, \omega_\gamma, \omega_\beta, \omega_\alpha)$, this controller comes to within 3% of the optimal controller for this particular scenario. In addition to its good performance, this control strategy is well structured and transparent; the degrees of freedom are decoupled, which results in a straightforward design and method of tuning this controller in comparison with the non-intuitive $\mathcal{H}_2$ controller.



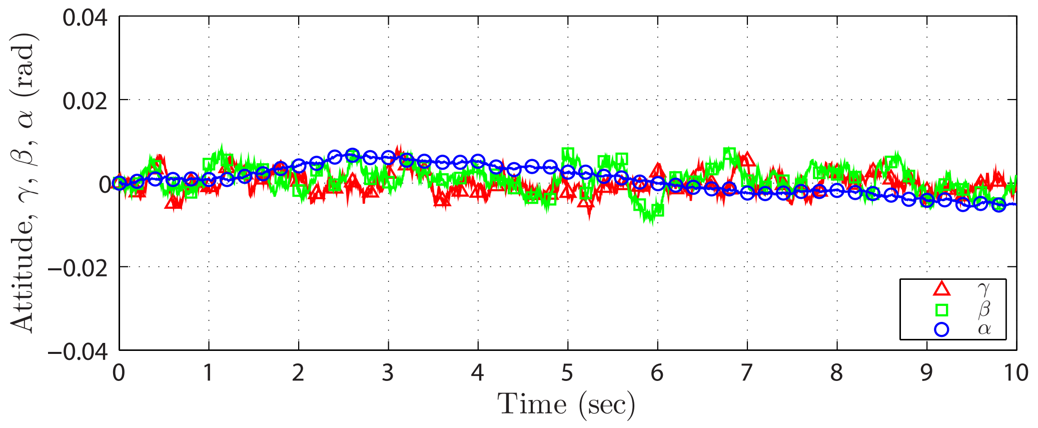


Figure 2.11 Simulation results around the hovering equilibrium using an $\mathcal{H}_2$ controller on the 20-module array configuration as seen in Figure 10(a). An $\mathcal{H}_2$ norm of 5.385 was obtained for this particular scenario.

2.5 Experiments

Three important aspects of the DFA were demonstrated in order to verify its feasibility: 1) Docking, 2) Driving, and 3) Flying. Each one in sequence is a prerequisite to the next, and therefore each is needed to achieve the end goal of coordinated flight.

Docking

A variety of experiments were performed to test the module's ability to dock. Based on experimental results, two modules have the highest probability of successfully docking when one is rotating and another is translating along converging trajectories, see Fig 2.12. Although docking will be sufficiently random, this is an important observation to consider when developing a driving strategy to ensure that modules readily connect with one another [25, 26].

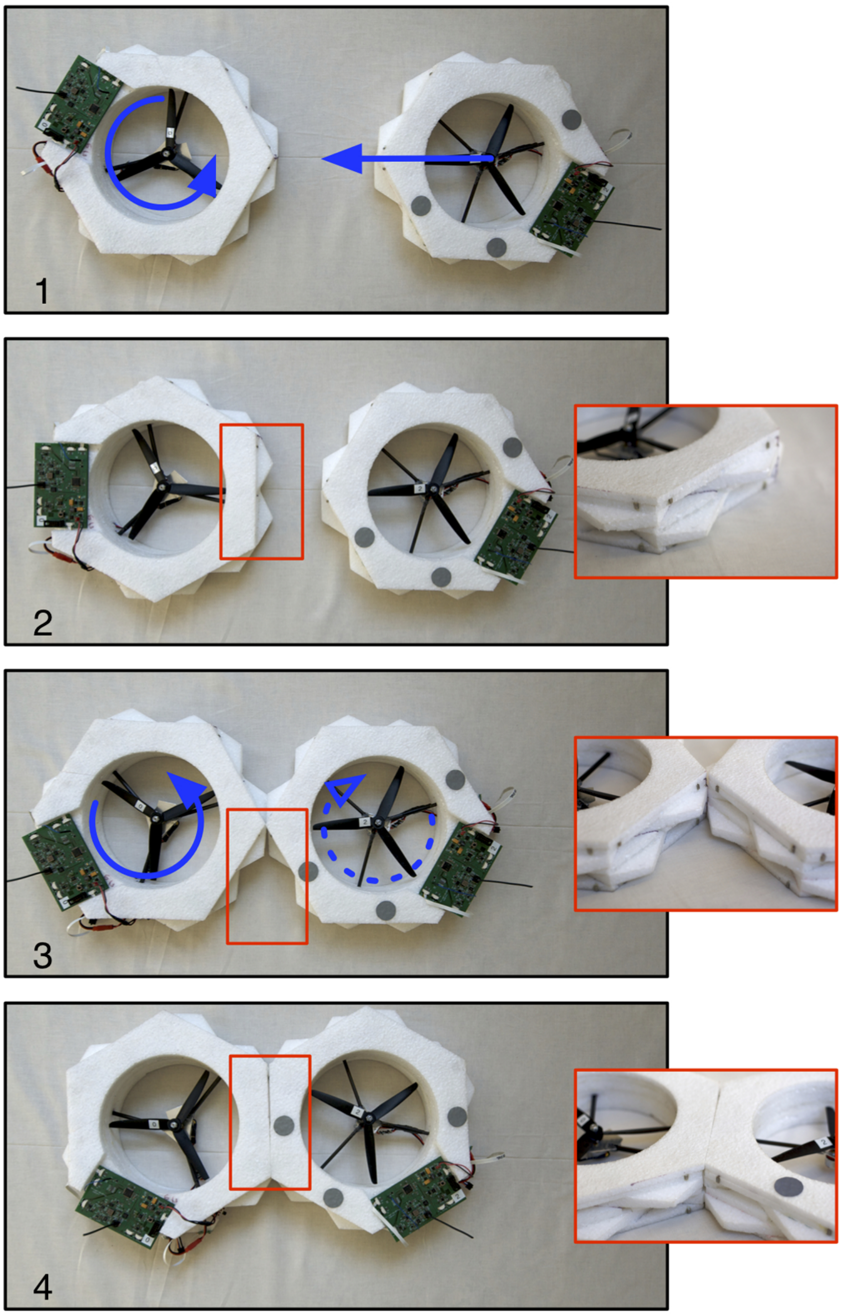


Figure 2.12 In frame 1 – 2 of this docking image sequence, the left module rotates counterclockwise while the right module drives forward, towards the rotating module. In frame 3, the modules collide. The left module continues to rotate counterclockwise and due to its protruding features it induces a clockwise rotation in the right module, which continues to drive forward. The modules finally dock together, making a solid connection in frame 4.

Driving

Two modules were driven together and used to demonstrate maneuvers which would otherwise be impossible without the use of omni wheels or a steering mechanism; omni wheels increase the degree of maneuverability and in turn eases docking constraints. The pair of modules performed pirouettes along a circular trajectory, i.e. the pair rotated around their combined center while both modules followed a 1.3 meter diameter circular trajectory in the direction opposite to their rotation.

A camera-based motion capture system was used to measure the performance of this pirouette maneuver, which performed very well considering that the maneuver was carried out open loop, see Figure 2.13 [27]. The pair was able to meet the desired trajectory with relatively good accuracy and precision while performing a 1.3 meter diameter orbit over six times. This repeatability adds weight to the method of module fabrication and the chosen driving surface.

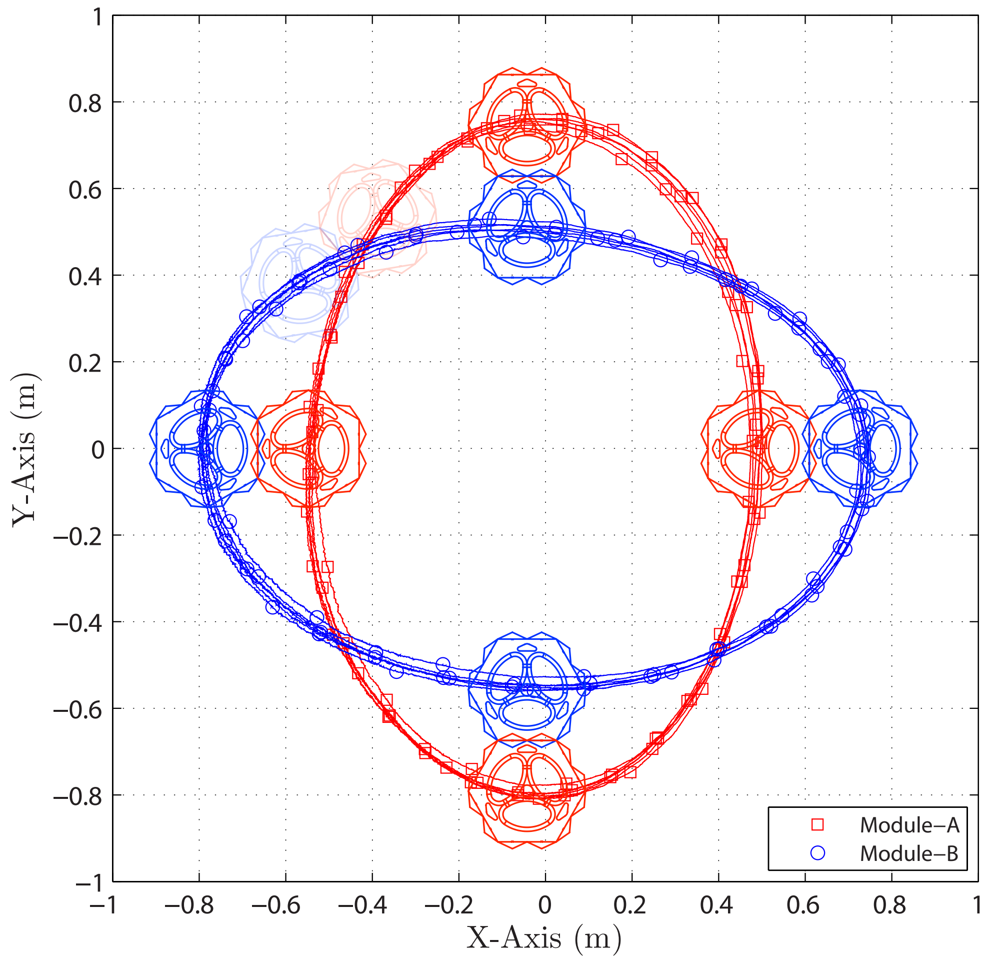


Figure 2.13 The two sets of trajectories in this plot represent the centers of two docked modules (A and B) during an open-loop pirouette driving maneuver, and demonstrates repeatability for over six orbits. The position of the modules were measured using a camera-based motion capture system.

Flying

The linear model presented in Section 2.2 and the results gathered from simulating the decentralized controller described in Section 2.4 were verified by testing this control strategy on the DFA in the array configuration shown in Figure 2.1. Preliminary results established that the pressure sensor performed poorly and provided imprecise altitude measurements; sudden changes to the environment, like the opening and closing of a room door, resulted in a dramatic change of pressure. Future revisions of the DFA will address this issue with sensors that are not as sensitive to changes in the environment, such as optical time-of-flight sensors. For this reason altitude control was excluded from the experiments. Moreover, yaw control was intentionally left out to simplify the analysis. As a result, experiments were made using only roll and pitch as feedback to the controller, see (2.31) – (2.32).

Before taking flight, the modules synchronized via infrared transceivers and removed all sensor bias over a 5 second initialization sequence. Rate-gyro measurements were made at 200 Hz and the controller was operated at 60 Hz. The function f_z (2.30), which is the deviation around the hovering thrust in units of acceleration, was set to a very small value. A camera-based motion capture system was used to measure both the altitude and the attitude of the DFA during flight, see Figure 2.15.

The DFA was shown to fly successfully with roll and pitch control. The experimental results shown in Figure 2.14(b) are comparable to the simulation results shown in Figure 2.14(a), thus verifying the utility of the linear model and the simulator.

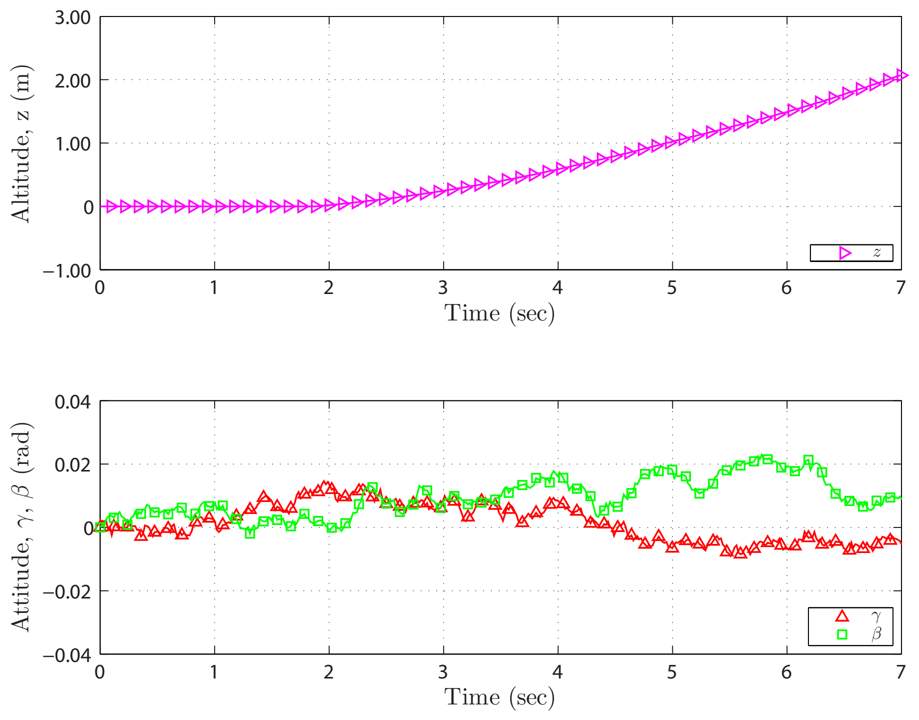
One interesting and unexpected result was observed due to yaw being left uncontrolled: the DFA ascended in a spiraling motion, keeping its x and y position of climb within the perimeter of the spiral, and thus mitigating drift. This behaviour is in fact expected due to the sine and cosine terms seen in (2.2) – (2.3), assuming that roll γ and pitch β are non-zero, and that the magnitude of yaw α is increasing over time. This motivates the following constant yaw-rate control law:

$$f_\alpha(\dot{\alpha}) = -\hat{I}_z \omega_\alpha (\dot{\alpha} - \dot{\alpha}_d), \tag{2.34}$$

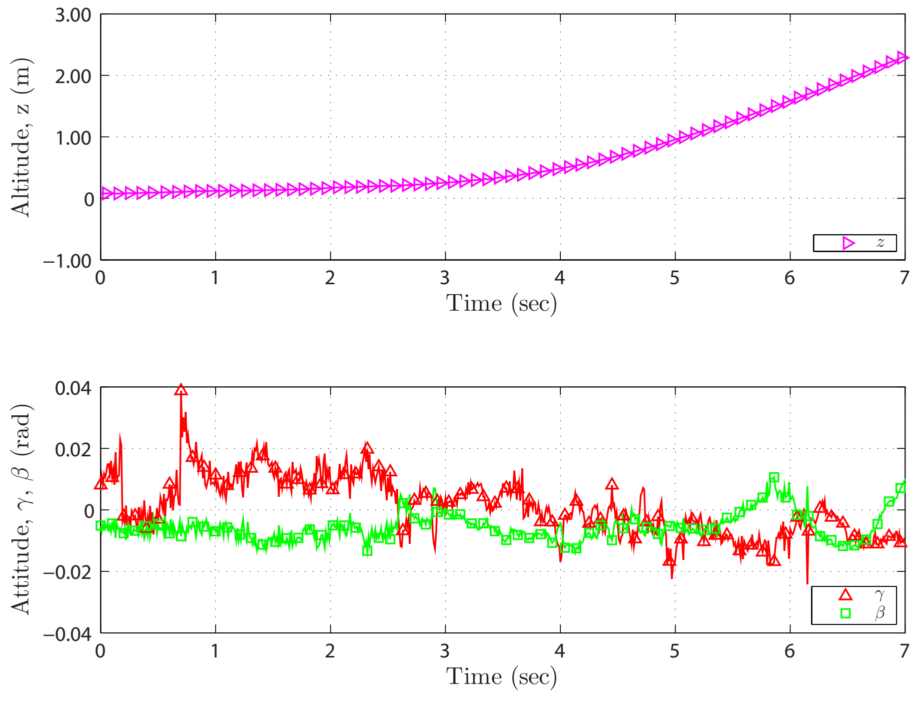
where $\dot{\alpha}_d$ is the desired yaw-rate.

2.6 Conclusions

This chapter presented the Distributed Flight Array (DFA), a modular multi-rotor vehicle capable of coordinated driving and coordinated flight. The initial prototype of the DFA established its feasibility by demonstrating various important aspects of the system, including: docking, driving, and flying. Nevertheless, it is only a proof-of-concept and has numerous limitations, resulting in its inability to fly in a variety of sizes and



(a) Simulation



(b) Experiment

Figure 2.14 Comparison of the results obtained from simulation versus experiment for a 4-module array configuration without feedback on altitude and yaw, where the natural frequencies and damping ratios for decentralized controller were set to $(\omega_\gamma, \omega_\beta) = (13, 13)$ rad/s and $(\zeta_\gamma, \zeta_\beta) = (1, 1)$, respectively. Measurements in the experiment were made using a camera-based motion capture system.

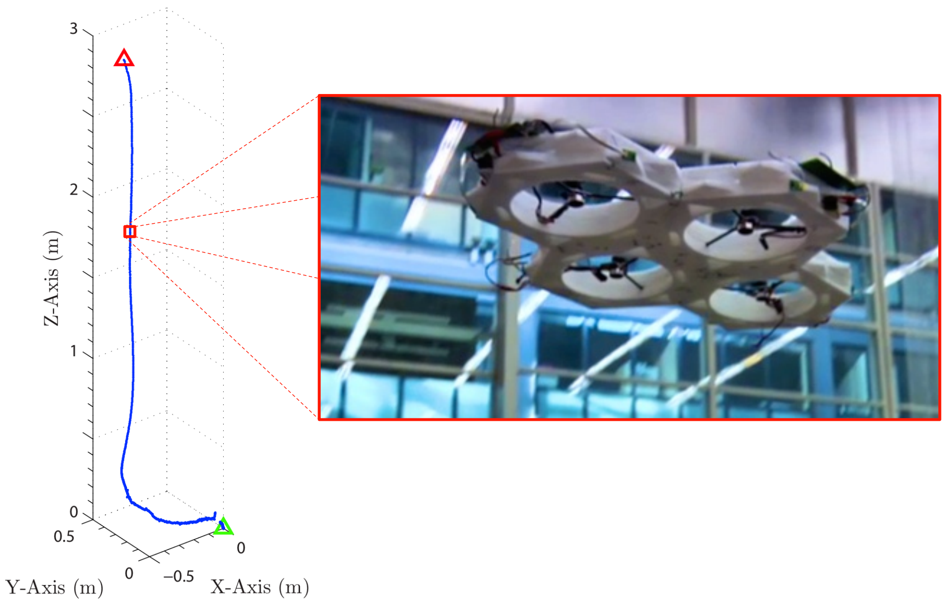


Figure 2.15 A plot of the DFA's 3D position relative to its take-off origin, obtained from experimental results, which were measured using a camera-based motion capture system.

configurations.

In order to handle all of the requirements set out in the introduction of this dissertation and to be able to use it as research platform, the system will need to be made more robust to mechanical impact. In particular, the printed circuit board will need to shrink in size for it to fit and be protected by the module's chassis. An alternative sensor will be needed for measuring distance to ground for its use in altitude control. Communication across each of its six interfaces and a method for computing the vehicle's topology online will need to be implemented. Both driving and flying functionality should be integrated, as they are currently separated in the proof-of-concept design in order to ease design contraints. Finally, a wireless base-station with a supporting software infrastructure will be needed to remotely command the vehicle, log telemetry data and send vehicle parameters (e.g. control tuning parameters) for experiments.

References

[1] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [Grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, March 2007.

[2] A. Klaptocz and J. D. Nicoud, *Technology and Fabrication of Ultralight Micro-Aerial Vehicles.* New York, USA: Springer-Verlag, 2009.

[3] D. Schafroth, S. Bouabdallah, C. Bermes, and R. Siegwart, “From the test benches to the first prototype of the mufly micro helicopter,” *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1, pp. 245–260, 2009.

[4] F. Van Breugel, W. Regan, and H. O. D. Lipson, “Demonstration of a passively stable, untethered flapping-hovering micro-air vehicle,” *IEEE Robotics & Automation Magazine*, vol. 15, pp. 68–74, 2008.

[5] S. Bouabdallah, A. Noth, and R. Siegwart, “PID vs LQ control techniques applied to an indoor micro quadrotor,” in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, vol. 3, Sept. 2004, pp. 2451–2456.

[6] R. Bouix, P. Viot, and J. L. Lataillade, “Polypropylene foam behaviour under dynamic loadings: Strain rate, density and microstructure effects,” *International Journal of Impact Engineering*, vol. 36, no. 2, pp. 329–342, 2009.

[7] Y. S. Lee, N. H. Park, and H. S. Yoon, “Dynamic mechanical characteristics of expanded polypropylene foams,” *Journal of Cellular Plastics*, vol. 00, pp. 1–13, 2009.

[8] J. Agullo, S. Cardona, and J. Vivancos, “Kinematics of vehicles with directional sliding wheels,” *Mechanism and Machine Theory*, vol. 22, no. 4, pp. 295–301, 1987.

[9] J. Lee, K. Yee, and S. Oh, “Aerodynamic characteristic analysis of multi-rotors using a modified free-wake method,” *Transactions of the Japan Society for Aeronautical and Space Sciences*, vol. 52, no. 177, pp. 168–179, 2009.

[10] W. Reed III, “Propeller-rotor whirl flutter: A state-of-the-art review,” *Journal of Sound and Vibration*, vol. 4, no. 3, pp. 526–530, 1966.

[11] A. H. Sacks and J. A. Burnell, “Ducted propellers – a critical review of the state of the art,” *Progress in Aerospace Sciences*, vol. 3, pp. 85–135, 1962.

[12] L. L. Howell, “Compliant mechanisms,” *21st Century Kinematics*, pp. 189–216, 2001.

[13] J. G. Leishman, *Principles of Helicopter Aerodynamics*, 2nd ed. Cambridge University Press, 2006.

[14] B. W. McCormick, *Aerodynamics of V/STOL Flight.* Dover Publications, 1999.

[15] R. J. Weir, “Aerodynamic design considerations for a free-flying ducted propeller,” Tech. Rep., Jan. 1988.

[16] P. J. Bristeau, P. Martin, E. Salaün, and N. Petit, “The role of propeller aerodynamics in the model of a quadrotor UAV,” in *Proceedings of the European Control Conference*, Aug. 2009, pp. 683–688.

[17] Y. Nishi, “Lithium ion secondary batteries: Past 10 years and the future,” *Journal of Power Sources*, vol. 100, no. 1-2, pp. 101–106, 2001.

[18] M. Wakihara, "Recent developments in lithium ion batteries," *Materials Science & Engineering*, vol. 33, no. 4, pp. 109–134, 2001.

[19] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[20] F. W. Shaw and E. Klavins, "Distributed estimation and control for stochastically interacting robots," in *Proceedings of the IEEE Conference on Decision and Control*, Dec. 2008, pp. 1895–1901.

[21] "Box2D physics engine homepage," Jan. 2010, [Accessed: Jan. 14, 2010]. [Online]. Available: http://www.box2d.org/

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

[23] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proceedings of the IEEE Conference on Decision and Control.* IEEE, 2007, pp. 5492–5498.

[24] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design.* Wiley, 2007, vol. 2.

[25] E. Klavins, "Programmable self-assembly," *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 43–56, Aug. 2007.

[26] G. M. Whitesides and B. Grzybowski, "Self-assembly at all scales," *Science*, vol. 295, no. 5564, pp. 2418–2421, 2002.

[27] "Vicon MX homepage," Aug. 2009, [Accessed: Dec. 1, 2012]. [Online]. Available: http://www.vicon.com/products/viconmx.html

3 System Description

Following the completion of a Distributed Flight Array (DFA) proof-of-concept, described in the previous chapter, the next step was to develop a fully functional research platform. In comparison to its predecessor, this revision would be capable of flying in a variety of configurations, which would require a solid solution for inter-module communication. The electronics would now include a sensor to measure distance to the ground for altitude control, and the size of the printed circuit board (PCB) would need shrink significantly in order to fit into the protective cavity of a module's chassis. The module itself would be more robust to mechanical impact. This revision would also combine both driving and flying functionality into a single unit, which made the design even more challenging due to weight constraints. Finally, a wireless infrastructure and software framework was needed to command the modules remotely, log telemetry data, send experiment parameters to the vehicle, and perform any other remote function necessary for experiment purposes.

All these design requirements were met in June 2012, where six modules successfully flew in a variety of configurations. Prior to this most recent revision (Revision 2) was one other revision (Revision 1), which took on a slightly different shape in an attempt to improve docking behaviour. In this revision, we experimented with a variety of materials and manufacturing processes, in particular Acetal plastics and laser cutting, which could be accomplished in-house. Although the chassis was mechanically robust, we later concluded that this manufacturing process severely constrained the mechanical design and led to undesired elastic behaviour during flight. We attempted to employ the latest manufacturing abilities, including flexible PCB design. This, however, was a dead end due to (at the time) long lead time and a time consuming design process. The most detrimental issue with this revision was the choice of propeller, a commerical off-the-shelf product. Not until the modules were assembled and being flight-tested, did we realize their difficiencies, which included: a non-reliable mounting mechanism for mounting the propeller to the motor; an unequal magnitude of torque generated by clockwise and counterclockwise propellers; and the mechanical vibrations generated by some of the propellers (not all) when spinning, which would cause the measurements from the onboard rate-gyroscope to saturate. We learned from these mistakes and developed what is now Revision 2, which resolves all of the limitations of the previous versions and has been shown to drive and fly in a coordinated fashion for up to twelve modules. Figure 3.1 shows the various hardware revisions of a DFA module.

Outline

The first part of this chapter begins in Section 3.1 with a detailed hardware description of a DFA module (Revision 2). A high-level architecture for the onboard firmware, needed to drive all of the electronics (e.g. sensors, actuators, communication) as well as the control and estimation algorithms is described in Section 3.2. The second part of this chapter focuses on the hardware and software infrastructure supporting this testbed. The base station and its hardware solution is described in Section 3.3; the software that supports the base station is described in Section 3.4. A simulation environment, which enables one to test various functionalities of the firmware and perform a variety of simulated experiments is described in Section 3.5.

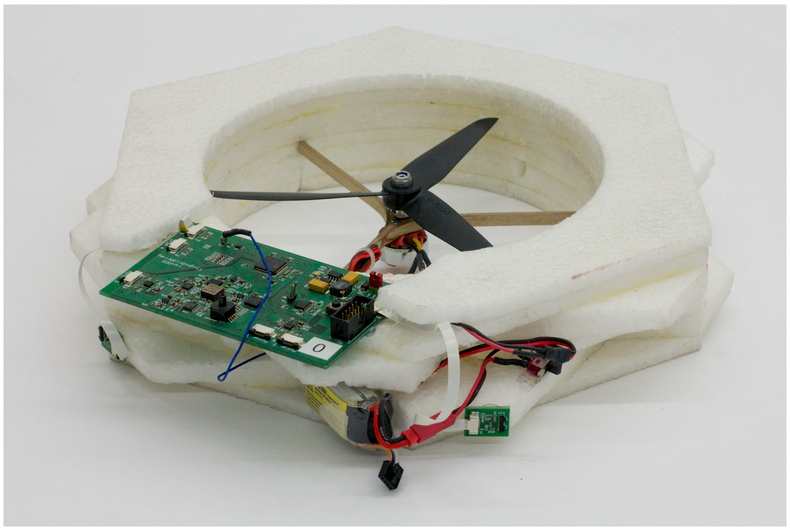
3.1 Module: Hardware Description

The DFA is composed of identical, hexagonal-shaped *modules*, see Figure 3.2. Much like a biological cell, a module is the smallest independent autonomously working unit of the system that is able to coordinate with its neighbours in order to perform a joint task. Each module has a characteristic length (side to side) of 250 mm and has a mass of approximately 255 g. Much of the mass resides on the outer perimeter of a module, giving it a relatively high mass moment of inertia around its vertical axis in comparison to the horizontal axes. Its center of mass is located away from the geometrical center, in the direction of the battery. This 250 mAh, 3-cell Lithium-ion polymer battery (Thunder Power RC G6 Pro Lite) is capable of delivering up to 70 W of continuous power and can power the module for approximately 3 min of flight, which is sufficient for performing experiments and live demonstrations.

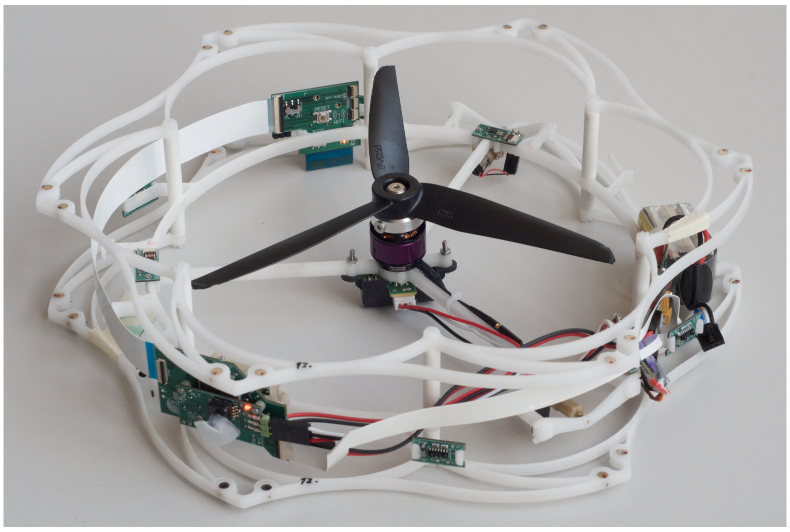
In the following, we divide the description of a module into six main categories: (A) Chassis, (B) Actuation, (C) Sensing, (D) Inter-Module Communication, (E) Base Station Communication, and (F) Computational Unit.

A. Chassis

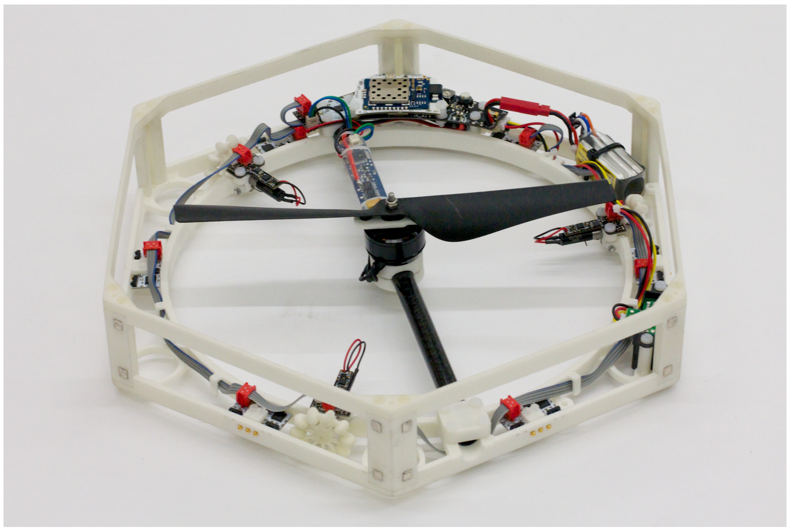
The chassis of each module is composed of three major components, see Figure 3.2: (1) a base-frame, which holds the electronics, motors, battery, etc.; (2) a top-frame used for structural rigidity and protection of the internal components from impact; and (3) standoffs located on each of the six vertices of the chassis, which connect the top frame with the base frame. Each of these components, which are held together by nylon screws, is constructed from Polyamide-12 (PA-12), an engineering plastic with very good mechanical properties. Such properties enable our modules to withstand mechanical impacts from a fall of up to 2 m on hard surfaces, for example. Selective laser sintering is used as the



(a) Proof-of-concept, completed July 2009



(b) Revision 1, completed June 2011



(c) Revision 2, completed June 2012

Figure 3.1 The various hardware revisions of a DFA module.

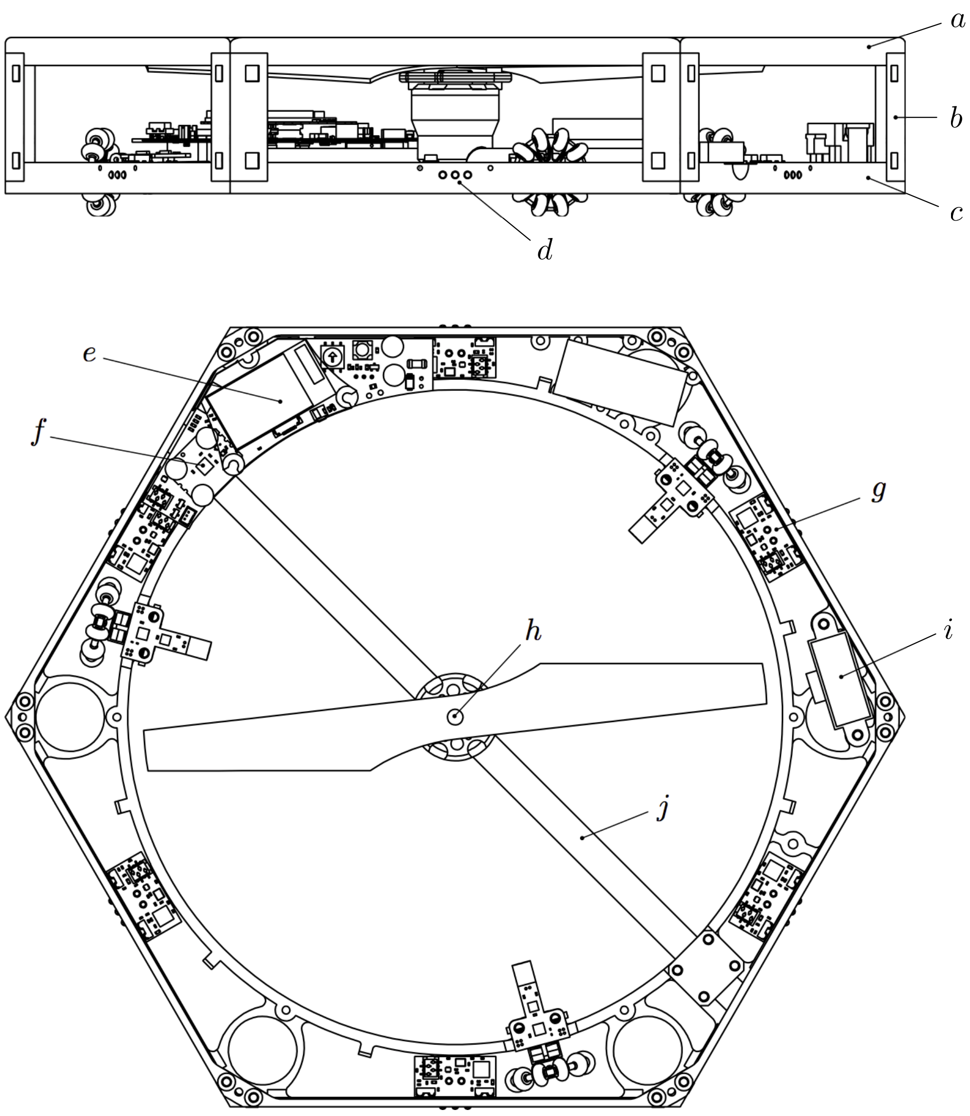


Figure 3.2 The primary components of a DFA module needed for flight include: (a) top-frame, (b) standoffs, (c) base-frame, (d) communication pins, (e) wireless communication and microcontroller electronics, (f) inertial measurement unit, (g) inter-module communication electronics, (h) brushless DC motor and 2-blade fixed-pitch propeller, (i) infrared distance measurement sensor, and (j) carbon fiber tube.

fabrication process, which enables one to design with very little fabrication constraints. Lying across the center of the chassis is a stiff carbon fiber plain weave tube, which adds rigidity to the structure. The chassis, however, is not perfectly rigid.

A module has six genderless electro-mechanical interfaces located along each of its six sides. This enables it to dock with up to six modules at a time. Four square shaped Neodymium Iron Boron magnets are used along each interface to keep adjacent modules connected. Magnets have a connection strength of 0.2 N each and are embedded in the standoffs. The connection strength along each interface has been designed to prevent a single module from disconnecting itself from the vehicle in flight due to its rotor's reaction torque (which is generated by the induced aerodynamic drag and change in angular momentum of the rotor during flight), but weak enough such that the force of gravity is sufficient to disconnect it from the vehicle when its rotor is turned off. This weakness in inter-module connection is of little concern during flight as the control strategy to be described in Chapter 5 minimizes the inter-module shear forces and moments.

B. Actuation

Flight Motor Positioned at the geometric center of the module and sitting on top of the carbon fiber tube is a commercial off-the-shelf 14-pole outrunner brushless DC motor (Hacker X-BL 52S), which is controlled by a brushless DC motor speed controller (X-3D X-BLDC) that is capable of a 100 Hz update rate over an inter-intergrated circuit (I^2C) bus interface. The motor is directly connected to a 2-blade fixed-pitch propeller (Parrot AR Drone Propellers C/A Type), which in combination can produce at least 3 N of thrust at the maximum rotational velocity of the motor. Due to the inability to vary the pitch of a propeller, a combination of clockwise (CW) and counterclockwise (CCW) rotating propellers are needed in a vehicle in order to cancel out the rotor's reaction torques during flight. This is the only feature of our system that defeats its homogeneity.

Drive Motors Each module is driven by 3 omni-directional wheels (in-house design) that are placed 120° apart, see Figure 3.3. These wheels are constructued from Polyamide-12 (PA-12), which is the same material used for the chassis and has very good mechanical properties. Selective laser sintering is used as the fabrication process, which enables one to design with very little fabrication constraints. In fact, the wheel itself is a single part with no need for assembly.

Each wheel, which is 30 mm in diameter, is driven by a DC motor (Faulhaber 0615N003S) with a 256:1 planetary gear (Faulhaber 06/1K). This results in a no-load speed of 78.9 RPM (or 8.26 rad/s) and stall torque of 56.32 mNm. The DC motors are controlled by a custom H-bridge controller that interfaces directly with the onboard microcontroller over an inter-intergrated circuits (I^2C) bus see Figure 3.4.

The motor is mounted on a spring-loaded suspension system. This is needed to absorb the shock that acts on the motor's axle upon impact when falling, which generally occurs when landing from flight. On the ground, assuming a smooth surface, the suspension system keeps the wheels fixed in position.

C. Sensing

Rate-Gyroscope Each module has a 3-axis digital-output microelectromechanical systems rate-gyroscope and accelerometer (Invensense MPU-6050) rigidly mounted to its chassis on top of the carbon fiber tube. This sensor communicates its information to the microcontroller over an I^2C bus, and is capable of sampling angular

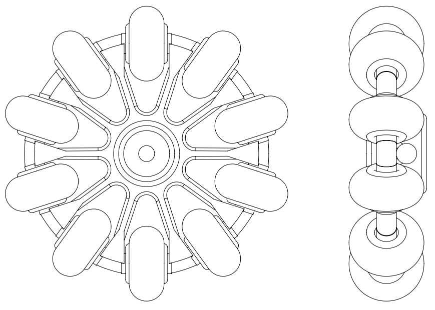


Figure 3.3 Each omni-directional wheel is constructed from Polyamide-12 through a selective laser sintering process, eliminating the need for assembly. The wheel consists of 10 rollers and has a diameter of 30 mm.

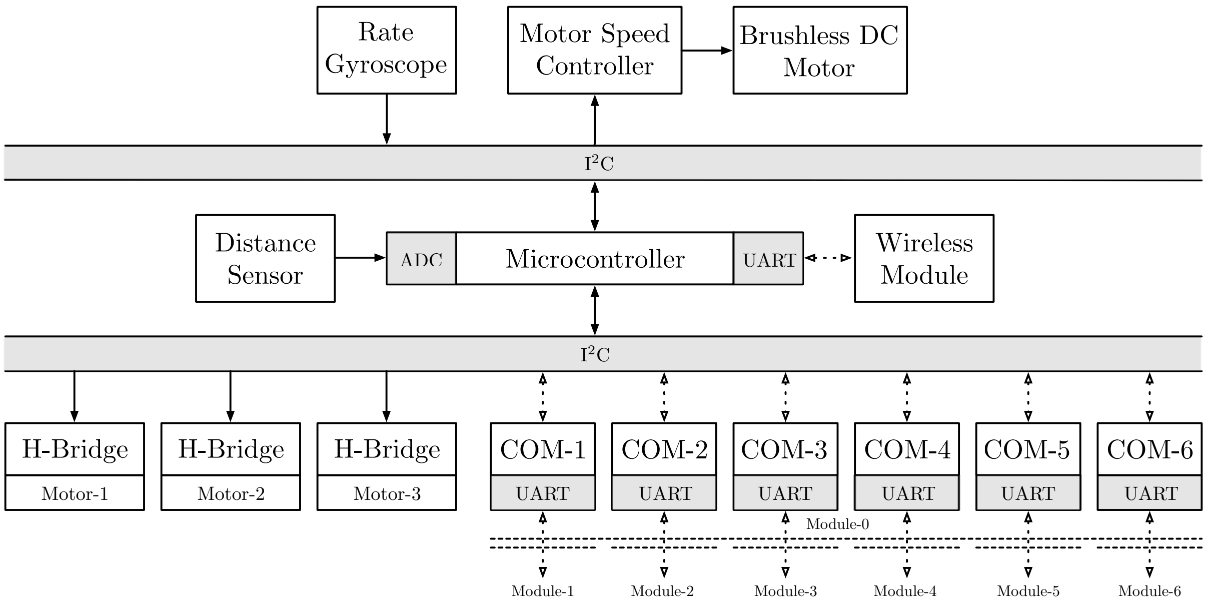


Figure 3.4 The primary electrical components of a DFA module needed for flight consist of a 32-bit microcontroller connected to a 3-axis rate-gyroscope, motor speed controller, and six inter-module communication interfaces over two independent I^{2}C buses. An IR distance measurement sensor is connected to the microcontroller's 12-bit ADC. A wireless module, which is interfaced through the microcontroller's UART peripheral, can be used to transmit telemetry data and/or receive commands to/from a base station.

rates and translational accelerations at 1 kHz with a resolution of 7.6×10^{-3} deg /s and 0.6×10^{-3} m/s^2 in the body-fixed frame of a module.

Distance Measuarement Sensor A module is also equipped with a general purpose infrared (IR) distance measurement sensor (Sharp GP2Y0A02YK), which is pointed downwards and perpendicular to the plane of the vehicle. This commonly used light-weight, low-cost sensor produces an output voltage that is a function of the distance being measured in the specified range of the device. This output voltage is updated by the device at approximately 24 Hz. A 12-bit analog to digital converter (ADC), which is available on the microcontroller, is used to measure the output voltage of this sensor resulting in a resolution of approximately 2.4 mm at 1 m distance. These sensors operate based on the principle of triangulation, which is almost immune to interference from ambient light and is relatively indifferent to varying surface reflectances of the object being detected [1]. Depending on the arrangement of sensors and the measuring distance, sensors tend to become cross-coupled when their beams overlap. For the work described here, this effect is mitigated by choosing configurations and flying height where sensors do not affect each other. In the future, this can solved by replacing the sensor with one that has a tighter beam cone and/or modulating the sensors on and off so as to not affect neighbouring sensors.

D. Inter-Module Communication

There are two different modes of communication available along each of the six electromechanical interfaces of a module: (1) wired interface, and (2) wireless interface.

Wired Interface Along each interface is a set of three spring-loaded pins, which are used to make electrical contact with the adjacent pins of a neighbouring module. The three electrical contacts (i.e. transmit, receive, and electrical ground) allow for full-duplex communication at 115.2 kbps between modules over a standard universal asynchronous receiver/transmitter (UART) interface. Custom designed electronic transceivers, each with their own UART interface (SC16IS75x) and independent pair of 64-byte first-in, first-out (FIFO) transmit and receive buffers are connected together over an inter-intergrated circuits (I^2C) bus, which interfaces directly with the microcontroller, see Figure 3.4; the transceivers act as *slaves* and the microcontroller acts as a *master* on the communication bus.

Wireless Interface A line-of-sight infrared (IR) transceiver allows for half-duplex communication at 115.2 kbps between modules at a distance. The range of these transceivers have been experimentally determined to be 1 m and has a transmission cone of roughly 5° (end-to-end).

E. Base Station Communication

We use one of two interchangeable wireless systems operating on a 2.4 GHz carrier band frequency to communicate with the modules from a base station, this will be discussed in more detail in Section 3.3. On the module-side, both systems interface with the microcontroller over its UART peripheral at a rate of 115.2 kbps. What separates the two systems is their reliability and directionality of communication. Depending on the environment and its intended use, one system is preferred over another.

F. Computational Unit

Handling all of the computation and peripheral drivers is a single onboard 72 MHz ARM-based 32-bit microcontroller (ST Microelectronics ARM M3-Cortex STM32F103x) with firmware that was developed in-house. This microcontroller resides on a custom designed printed circuit board that interfaces with all of the previously described peripherals and can be extended to include more through a number of general purpose input and output pins, see Figure 3.4.

3.2 Module: Firmware Architecture

Inter-Task Interactions

The firmware primarily employs a time-triggered architecture [2]. For modularity, it is divided into three major concurrent tasks: (1) communication, (2) estimation, and (3) control, see Figure 3.5. These tasks are representative of their time-scales, but more importantly relies on an independent set of peripherals. As such, these tasks could be physically separated on different processors in future revisions of the hardware. However, in our implementation, each is independently handled by its own timer-based interrupt and service routine. Within each task are processes that are coordinated via a cooperative-scheduler [3]; this minimizes the overhead needed to guarantee mutual exclusion of processes and results in very deterministic outcomes. The description of the three major concurrent tasks are listed below:

Communication This task handles all inter-module communication, as well as communication with the base station. It is also responsible for commanding the drive motors, which employs the same peripheral hardware (I^2C bus) as the inter-module transceiver electronics. This is needed to ensure that both processes (inter-module communication and drive motor commands) do not use the hardware peripheral simultaneously, which is handled via a cooperative-scheduler within this task.

Estimation This task operates at the update rate of the rate-gyroscope (1 kHz), polls the rate-gyroscopes and ADC, which captures IR distance measurement sensor measurements, and pipes this information through the state estimator.

Control This task handles the flight controller and interfaces with the flight motor electronic speed controller (ESC). It operates at the maximum command rate of the flight motor ESC (100 Hz).

In a time-triggered architecture information is pulled at predefined periodic instances; data remains in memory until updated, and updates are done in-place. This paradigm relegates temporal control to the highest-level functions and supports independence between processes, i.e. functions that share information can operate at different rates and there are no means by which a function that receives information can influence the sending of information from another function [4]. To manage concurrency, in particular race

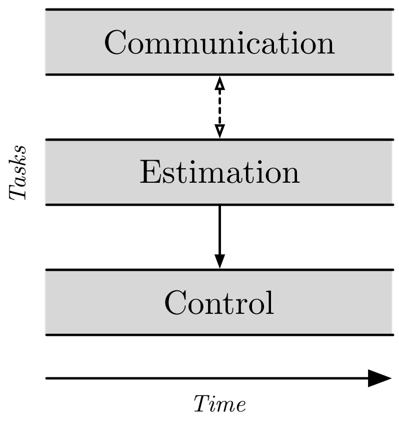


Figure 3.5 There exists three major concurrent tasks: (1) communication, (2) estimation, and (3) control. These tasks are representative of their time-scales, but also relies on an independent set of peripherals. Within each task are processes (not shown) that are coordinated via a cooperative-scheduler. By employing a time-triggered architecture, tasks (which may run at different frequencies) may interact with one another without compromising deterministic timing behaviour, which is needed for real-time systems.

conditions, we employ something similar to the non-blocking write protocol described in [5] for shared data. This requires an additional control variable (with atomic hardware access) for each piece of shared data. This method is more deterministic and achieves better performance compared to real-time systems that lock data, for example mutexes and semaphores, which places a waiting process in a queue (or stack) for an indeterminate period of time.

Inter-Component Interactions

The firmware is divided into three major hierarchical layers, starting from the bottom and working our way up: (1) hardware abstraction, (2) generic middleware, and (3) application layer, see Figure 3.6. The hardware abstraction layer is specific to the hardware and it accounts for low-level communication between peripheral devices. The generic middleware tracks and coordinates data traveling to and/or from the various peripherals; it generally handles all of the timing requirements. The application layer contains high-level algorithms and various libraries, for example linear algebra functions and methods that compute the physical topology of the vehicle, which are needed for state estimation and control of the vehicle. The application layer contains code that can be wrapped in a MATLAB MEX file and simulated offline.

Communication Protocol

All communication (i.e. inter-module and base station) is performed in an *unreliable, but frequent* manner using the user datagram protocol (UDP) along with an addition-based modular checksum for error detection. This has the advantage of low complexity in the communication's transport layer, and for lossy communication channels the most current data always has priority. This is important in a real-time embedded system with limited

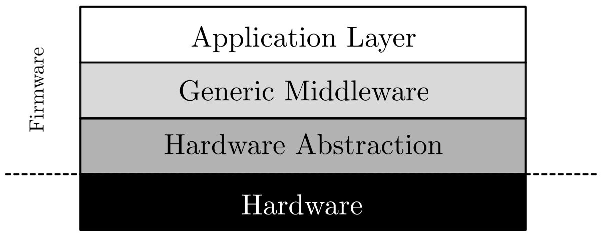


Figure 3.6 The firmware is divided into three layers: (1) hardware abstraction, which is specific to the hardware and accounts for low-level communication between peripheral devices; (2) generic middleware, which tracks and coordinates data traveling to and/or from various peripherals; and (3) application layer, which contains high-level algorithms and libraries.

memory and processing capabilities.

3.3 Infrastructure: Base Station

The base station is a means for interacting with the modules remotely. This could be used, for example, to control the movement of the modules and/or vehicle, send external sensor information to the modules for feedback control, or logging telemetry data that is sent from the modules to the base station. In terms of physical hardware it consists of 3 major components: (A) computer, (B) wireless interface, and (C) user-interface.

A. Computer

Any modern day commercial off-the-shelf computer with a Windows operating system, with at least two universal serial bus ports, and one ethernet port is sufficient to operate as a base station computer.

B. Wireless Interface

As previously mentioned, a module can use one of two interchangeable wireless systems operating on a 2.4 GHz carrier band frequency to communicate with the base station. What separates the two systems is their reliability and directionality of communication. Depending on the environment and its intended use, one system is preferred over another.

Variable-Latency, Bidirectional Communication A variable-latency, bidirectional multicast system (Roving Networks RN-131) can be used to communicate data (e.g. telemetry) and user-commands between modules and the base station. This system uses orthogonal frequency-division multiplexing and complies with IEEE 802.11g specifications. The advantage here is the ability to use an existing wireless infrastructure, provided that there is one. We have observed that this system does not perform reliably in WiFi saturated environments, which is a problem if the information that is being sent over this system is needed for real-time control.

Low-Latency, Unidirectional Communication A low-latency, unidirectional broadcast system (Laird Technologies LT2510[1]) is used for transmitting user-commands (e.g. take-off, land, waypoints, hold, etc.) to the modules from a base station computer. This system uses the method of frequency-hopping spread spectrum for wireless transmission [6], which we have observed as operating reliably in WiFi saturated environments.

C. User Interface

The modules can be remotely commanded by a keyboard and/or a joystick interface. Telemetry data from the modules is provided in a graphical user interface for visual feedback to the user.

3.4 Infrastructure: Software

The backbone architecture for the software that runs on the base station computer derives from the same code base used in ETH Zurich's Flying Machine Arena [7]. This software infrastructure consists of independently running processes that are arbitrarily distributed among one or more physical computing platforms linked together by an Ethernet bus. Inter-process communication is accomplished using multicast UDP streams. As such, a process sends out a packet once and processes that require this information simply subscribe to this multicast stream. Figure 3.7 illustrates the software infrastructure for the DFA's base station. The following lists and describes each process in some detail.

Wireless Bridge This process acts as a portal for transferring data wirelessly to the DFA modules and for handling the data from the modules to the Ethernet bus. There exists, in fact, two separate processes for this due to the fact that there are two wireless systems in place, see previous section. One process handles the bi-directional case of sending and handling data using multi-cast UDP packet streams over an WiFi access point; and the other handles the uni-directional case of just sending data out over a serial port, which is piped to a low-latency broadcast transmitter.

Logger As a convenient side effect to this particular software architecture is that data logging is trivial; a process simply subscribes to all the desired multicast streams and records them with a timestamp to a data file. This information includes, but is not limited to: telemetry data from the modules, remote control commands, and external sensor information.

[1]Although the LT2510 can also be used bidirectionally, implementation is not straightforward with multiple transceivers and is therefore not used.

Graphical User Interface This acts as the software's front-end. Processes can be started and stopped, telemetry data can be displayed in real-time on a graphical time plot, and parameters of the modules can be remotely adjusted.

Send Parameters Experimental parameters can be defined in this process and have it sent over the *Wireless Bridge* to the modules. This saves time in an experiment if, for example, tuning parameters need to be quickly adjusted in an experiment, as oppose to repeatedly re-programming each module.

Joystick This process handles any generic joystick device, which can be used to control the motion of the modules and/or vehicle. This is generally used when operating in a workspace that does not have external sensors for position control.

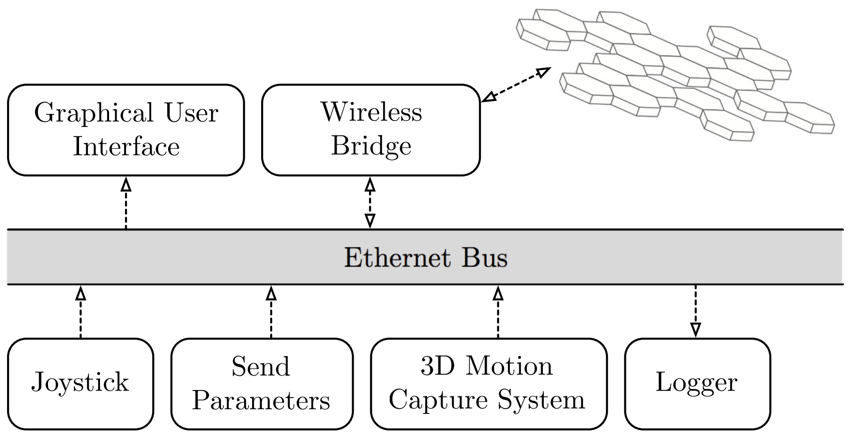


Figure 3.7 The software infrastructure relies on an Ethernet bus as a common resource among all independently running processes. Processes communicate by sending and subscribing to multicast packet streams. These processes include: *3D Motion Capture System*, which tracks the global pose of the vehicle in its workspace; *Send Parameters*, which sends variable parameters to the testbed for quickly adjusting experiments; *Joystick*, which interfaces with a hand-held joystick and is used to manually control the vehicle remotely; *Graphical User Interface*, which is used for software configuration and displaying telemetry data; *Wireless Bridge*, which transmits and receives data to/from the testbed over a wireless interface; and *Logger*, which subscribes to all relevant packet streams for later playback and analysis of experiments.

3.5 Infrastructure: Simulation Environment

Due to its ability to quickly prototype, rapidly visualize information, and its familiarity in the engineering community (which is important if the system is intended to be used as a research platform), we use MATLAB as our simulation environment. Much of the firmware's application layer, which is primarily written in the C programming language, can be wrapped in a MATLAB MEX file and simulated in order to verify its behaviour prior to its use on the hardware.

3.6 Conclusions

The DFA has evolved significantly since its initial stages as a proof-of-concept and we will demonstrate in the remaining chapters of this dissertation some of its functionality. It is worth mentioning, however, some of the limitations imposed by the current hardware platform. Still, one of the limiting factors in this revision is the altitude sensor, i.e. an IR distance measurement sensor. This sensor has a very limited range, where the precision decreases exponentially the further it is away from the measured surface. Apart from this is the fact that these sensors interfere with one another provided that they are placed within each others proximity. Another major concern is weight. The mass of a module in comparison to the maximum available thrust from its rotor is quite high. This puts a hard limit on a module's maximum control effort, and in fact reduces its ability to control certain configurations of the vehicle. A more obvious consequence of the weight is the flight time, which is currently not very substantial. There are certainly improvements that can be made to both the mechanical features and electronics in order to reduce weight. However, this will be a tradeoff between weight and chassis stiffness.

References

[1] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots.* MIT press, 2004.

[2] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[3] M. Pont, *Patterns for Time-Triggered Embedded Systems: Building Reliable Applications with the 8501 Family of Microcontrollers.* Addison Wesley Longman, 2001.

[4] H. Kopetz, *Real-time Systems: Design Principles for Distributed Embedded Applications.* Springer, 2011, vol. 25.

[5] H. Kopetz and J. Reisinger, "The non-blocking write protocol NBW: A solution to a real-time synchronization problem," in *Real-Time Systems Symposium.* IEEE, 1993, pp. 131–137.

[6] J. Proakis, *Digital Communications*, ser. McGraw-Hill Series in Electrical and Computer Engineering. McGraw-Hill, 2000.

[7] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: the flying machine arena," *Mechatronics, Under Review.*

4 Distributed Tilt Estimation

Tilt estimation has been addressed in a myriad of papers using a variety and combination of sensors, ranging from inertial measurement units to computer vision [1]–[7]. Although the majority of these methods certainly apply to the Distributed Flight Array (DFA), we present in this chapter an approach that exploits the distributed nature of our system. In particular, we present a generalized approach for estimating the tilt and altitude of a body with respect to an inertial coordinate frame using a network of distance measurement sensors. In the context presented here, this network of distance measurement sensors come from the modules of the DFA itself. In a centralized system architecture, the solution to this problem is given by solving a linear least squares problem. It will be shown, however, that this can be reformulated as two independent distributed averaging problems. This can then be solved on a distributed system architecture, where agents do not have direct access to all of the information in the network.

One of the main contributions of our work is a method for solving the previously described distributed averaging problem. Our method falls into a class of algorithms known as linear distributed average consensus. We provide proof of its correctness, and demonstrate how it can be implemented on a resource-limited embedded system, such as the DFA.

We then provide experimental results demonstrating the effectiveness of our distributed tilt estimate. This method can then be combined with other forms of attitude sensing and estimation such as integrated angular rates from rate-gyroscopes, which have good short-term characteristics but suffer from long-term drift. Fusing estimates in this case can be accomplished using, for example, a complimentary filter or a Kalman filter.

Outline

We will begin in Section 4.1 by describing how our distributed tilt estimation problem can be formulated more generally as an distributed averaging problem. Section 4.2 will proceed to describe a distributed average consensus algorithm, and suggest in Section 4.3 how one might implement this algorithm on a resource-limited embedded system. This algorithm is generalized, and can be used in a number of applications where one may want to compute the average of some quantity over a network of agents. In the same section we demonstrate its effectiveness with simulations and experiments. In Section 4.4 we show

how this algorithm can be used for distributed tilt estimation. The tilt estimate we obtain from our method performs relatively well in comparison to ground truth. However, the estimate is delayed and has a relatively slow update rate, which scales with the algebraic connectivity of the network. In Section 4.5 we compensate for the delay and fuse this estimate with local angular rate measurements in order to obtain a real-time distributed tilt estimate that can be used for control purposes. Concluding remarks are made in Section 4.6.

4.1 Distributed Altitude and Tilt Estimation using Multiple Distance Measurements

In this section we demonstrate how a network of distance measurement sensors, placed at known positions on a body, can be used to estimate the altitude and tilt of this body. Although this method is generalized, the DFA will be used as an example.

Preliminaries

Our system may be thought of as a dynamic mesh sensor network, where sensor measurements change simultaneously with time. More precisely, however, is that our sensors do not take measurements simultaneously, but their update rate is sufficiently fast with respect to the dynamics of the vehicle, rendering asynchronous measurement effects negligible.

Each distance measurement sensor $i \in \{1, \ldots, \mathrm{N}\}$ is positioned at $(\mathrm{x}_i, \mathrm{y}_i)$, with respect to the body-fixed coordinate frame B of the vehicle and measures the distance $\hat{m}_i$ from the agent to the ground, see Figure 4.1. The body coordinate frame B is positioned at the vehicle's centre of mass and is aligned in a right-hand coordinate system with the principal axes of rotation, where the z-axis points upward. The orientation of B with respect to the inertial coordinate frame I is described by ZYX-Euler angles, which is commonly used in the aeronautics field to represent yaw α, pitch β, and roll γ, respectively acting first around the z-axis, followed by the y-axis, and lastly around the x-axis [8].

Relating Altitude and Tilt to Distance Measurements

The Distributed Flight Array is equipped with infrared distance measurement sensors, which can be used to estimate altitude z and tilt (i.e. roll γ and pitch β) in flight assuming that the surface over which it flies is flat and horizontal. The following provides a model relating these states of the vehicle to sensor measurements.

The body-fixed coordinate frame B is positioned at the vehicle's center of mass and is aligned in a right-hand coordinate system with the principal axes of rotation, where the z-axis is pointing upwards, see Figure 4.2.

The orientation of the body-fixed coordinate frame B with respect to an inertial frame I is described by ZYX-Euler angles, which is commonly used in the aeronautics field to

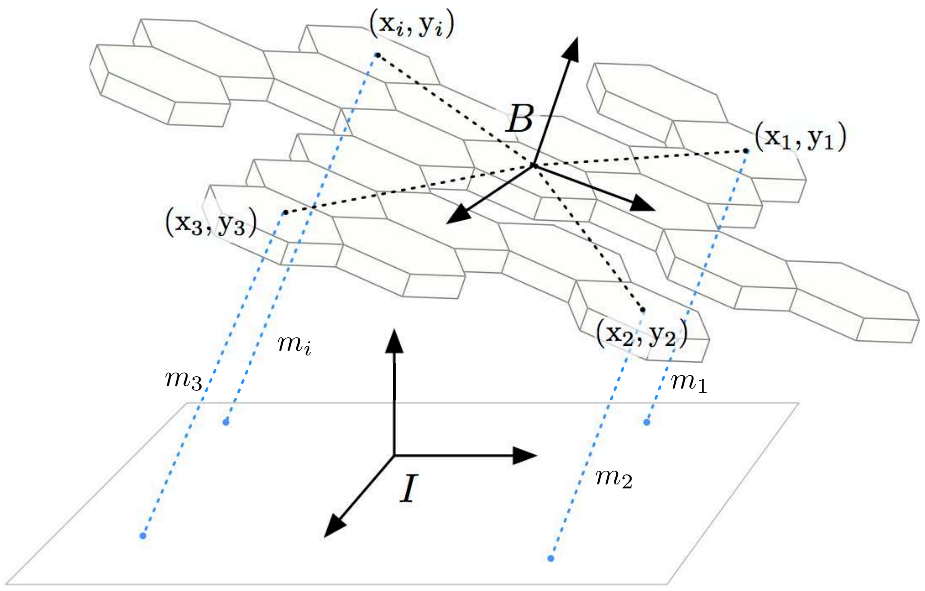


Figure 4.1 The DFA's body-fixed coordinate frame B is located at its centre of mass and aligned with the principal axes of rotation. The orientation of the vehicle with respect to the inertial coordinate frame I can be described by a set of ZYX-Euler angles. Over a flat surface, it is possible to estimate the tilt of the vehicle with respect to this surface using the distance measurements $\hat{m}_i$ of sensor i located at (x_i, y_i) with respect to the vehicle's body coordinate frame.

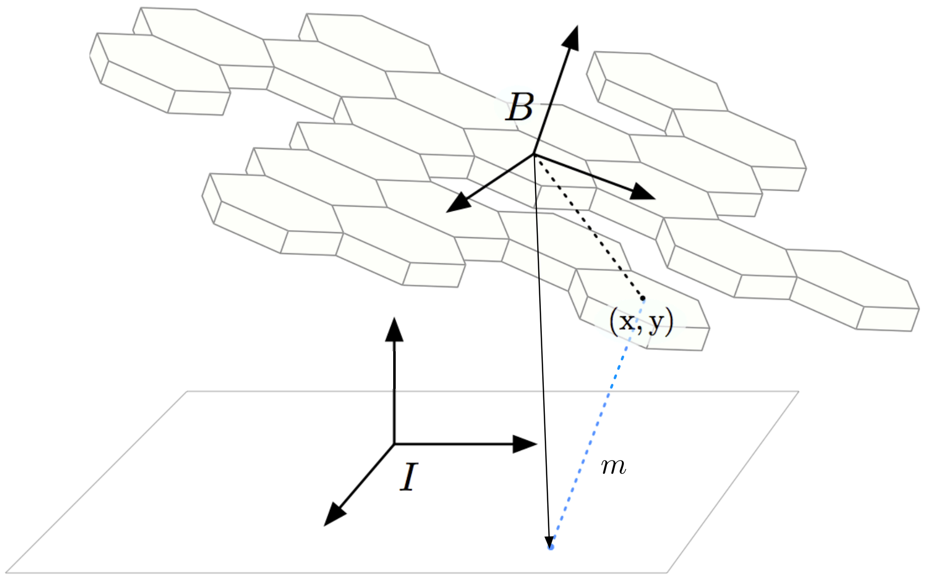


Figure 4.2 The orientation of the vehicle (i.e. the body-fixed coordinate frame B) with respect to an inertial frame I can be described by a set of ZYX-Euler angles.

represent yaw α, pitch β, and roll γ and can be expressed by the rotation matrix

$$ {}^{I}_{B}\mathbf{R}_{\mathrm{zyx}} = \mathbf{R}_{\mathrm{z}}(\alpha)\mathbf{R}_{\mathrm{y}}(\beta)\mathbf{R}_{\mathrm{x}}(\gamma), \tag{4.1}$$

$$= \begin{bmatrix} \mathrm{c}_\alpha \mathrm{c}_\beta & \mathrm{c}_\alpha \mathrm{s}_\beta \mathrm{s}_\gamma - \mathrm{s}_\alpha \mathrm{c}_\gamma & \mathrm{c}_\alpha \mathrm{s}_\beta \mathrm{c}_\gamma + \mathrm{s}_\alpha \mathrm{s}_\gamma \\ \mathrm{s}_\alpha \mathrm{c}_\beta & \mathrm{s}_\alpha \mathrm{s}_\beta \mathrm{s}_\gamma + \mathrm{c}_\alpha \mathrm{c}_\gamma & \mathrm{s}_\alpha \mathrm{s}_\beta \mathrm{c}_\gamma - \mathrm{c}_\alpha \mathrm{s}_\gamma \\ -\mathrm{s}_\beta & \mathrm{c}_\beta \mathrm{s}_\gamma & \mathrm{c}_\beta \mathrm{c}_\gamma \end{bmatrix},$$

where c and s are shorthand forms of cosine and sine, respectively, and

$$\mathbf{R}_\mathrm{z}(\alpha) := \begin{bmatrix} \mathrm{c}_\alpha & -\mathrm{s}_\alpha & 0 \\ \mathrm{s}_\alpha & \mathrm{c}_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.2}$$

$$\mathbf{R}_\mathrm{y}(\beta) := \begin{bmatrix} \mathrm{c}_\beta & 0 & \mathrm{s}_\beta \\ 0 & 1 & 0 \\ -\mathrm{s}_\beta & 0 & \mathrm{c}_\beta \end{bmatrix}, \tag{4.3}$$

$$\mathbf{R}_\mathrm{x}(\gamma) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_\gamma & -\mathrm{s}_\gamma \\ 0 & \mathrm{s}_\gamma & \mathrm{c}_\gamma \end{bmatrix}. \tag{4.4}$$

The distance measurement sensors are pointed downwards and perpendicular to the plane of the vehicle, measuring the distance from the sensor to a point on the ground, see Figure 4.2. This distance m is a function of the sensor's position (x, y) with respect to the vehicle's body coordinate frame B, and the tilt of the vehicle (roll γ and pitch β) with respect to inertial frame I. More precisely, the altitude z of the vehicle is the scalar projection of a vector ${}_B\mathbf{k} = (\mathrm{x}, \mathrm{y}, -m)$ in the vehicle's body coordinate frame B, pointing from the center of rotation of the vehicle to the point of measurement, onto the unit normal ${}_B\hat{\mathbf{z}}$ of the vehicle's plane (known commonly as the Hesse Normal form),

$$-{}_B\hat{\mathbf{z}} \cdot {}_B\mathbf{k} = z. \tag{4.5}$$

The unit normal ${}_B\hat{\mathbf{z}}$ is obtained by rotating the z-axis unit vector in the inertial frame ${}_I\hat{\mathbf{z}} = (0, 0, 1)$ to the body coordinate frame B via the rotation matrix (4.1),

$${}_B\hat{\mathbf{z}} = {}^I_B\mathbf{R}_{zyx} \cdot {}_I\hat{\mathbf{z}} \tag{4.6}$$

$$= \begin{bmatrix} \sin\beta \\ -\cos\beta \sin\gamma \\ -\cos\beta \cos\gamma \end{bmatrix}. \tag{4.7}$$

Making the appropriate substitutions into (4.5) and solving for distance m yields

$$m = \frac{z + \mathrm{y}\cos\beta\sin\gamma - \mathrm{x}\sin\beta}{\cos\beta\cos\gamma}. \tag{4.8}$$

It is clear from (4.8) that distance has a nonlinear dependency on the roll γ and pitch β of the vehicle. When the vehicle is hovering, however, small angles can be assumed and the expression can be linearized $L\{\,\cdot\,\}$ as the following:

$$L\{m\} = z + \mathrm{y}\cdot\gamma - \mathrm{x}\cdot\beta. \tag{4.9}$$

Assuming zero-mean sensor noise v, the measured value of this distance sensor at hover can be modeled as

$$\hat{m} = z + \mathrm{y}\cdot\gamma - \mathrm{x}\cdot\beta + v. \tag{4.10}$$

A Linear Least Squares Problem

The distance measurement model for all sensors in the vehicle can be compactly expressed as

$$\mathbf{m} = \mathbf{C}\mathbf{s} + \mathbf{v}, \tag{4.11}$$

where the elements of vector $\mathbf{m} = (\hat{m}_1, \ldots, \hat{m}_\mathrm{N})$ are the distance measurements from each sensor, $\mathbf{v} = (v_1, \ldots, v_\mathrm{N})$ is a vector containing the combined sensor noises, $\mathbf{s} = (z, \gamma, \beta)$ is a vector containing the states to be estimated, and the matrix $\mathbf{C}$ contains information pertaining to the positions of all the sensors,

$$\mathbf{C} = \begin{bmatrix} 1, & \mathrm{y}_1, & -\mathrm{x}_1 \\ \vdots & \vdots & \vdots \\ 1, & \mathrm{y}_\mathrm{N}, & -\mathrm{x}_\mathrm{N} \end{bmatrix}. \tag{4.12}$$

This information can be obtained prior to flight; agents share information in order to agree upon a unique coordinate frame and computes for itself the position its distance measurement sensor is from this coordinate frame [9].

Given at least three sensors and full column rank of $\mathbf{C}$ (i.e. sensors that are not collinearly aligned), it is trivial to solve for the state $\mathbf{s}$. In general, the set of equations is overdetermined; in this case the maximum-likelihood state estimate $\hat{\mathbf{s}} = (\hat{z}, \hat{\gamma}, \hat{\beta})$ can be obtained via least squares,

$$\hat{\mathbf{s}} = (\mathbf{C}^{\mathrm{T}}\mathbf{C})^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{m}. \tag{4.13}$$

This is in fact the linear optimal solution for this type of problem. The expected value of the state estimate is unbiased and is therefore given by $E\{\hat{\mathbf{s}}\} = \mathbf{s}$. The covariance matrix $\mathbf{\Sigma}$ of the temporal state estimate error $(\hat{\mathbf{s}} - \mathbf{s})$ is given by

$$\mathbf{\Sigma} = (\mathbf{C}^{\mathrm{T}}\mathbf{C})^{-1}\sigma_m^2. \tag{4.14}$$

This implies that the temporal variance of the state estimate is a function of the variance in a sensor's distance measurement. This value will in general decrease with the number of sensors. Moreover, the variance in the tilt estimate will decrease the further the sensors are away from the centre of rotation.

Distributed Linear Least Squares

The solution to solving the linear least squares problem via a Moore-Pensore (or pseudoinverse) (4.13) inherently relies on a centralized system architecture. It assumes that all the measurements and sensor positions is available for computation. Apart from the implementation difficulties of sharing such information reliably over a mesh network, such as that of the DFA, this approach consumes an increasing amount of communication bandwidth as the size of the system N gets larger. Moreover, computation on its own (i.e. $\mathcal{O}(\mathrm{N}^3)$ floating point operations) is by no means scalable.

We seek an approach that is scalable and that can be implemented on a resource-limited embedded system; we would like to solve the set of equations (4.13) in a distributed way, where each agent only communicates with its immediate neighbour(s) and performs a fixed number of floating point operations. To do this, we borrow results from [10] and re-write the least squares solution (4.13) element-wise as

$$\hat{\mathbf{s}} = \mathbf{A}^{-1}\mathbf{b}, \tag{4.15}$$

where

$$\mathbf{A} = \frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}} \mathbf{c}_i^{\mathrm{T}}\mathbf{c}_i, \quad \mathbf{b} = \frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}} \mathbf{c}_i^{\mathrm{T}}\hat{m}_i, \tag{4.16}$$

and $\mathbf{c}_i$ denotes row i of the matrix $\mathbf{C}$. What we now have are two averaging problems (4.16). If, however, the configuration of the vehicle is known ahead of time, then $\mathbf{A}$ can be pre-computed leaving only $\mathbf{b}$ to be computed at each measurement update. In general, the configuration of the vehicle may change during flight (e.g. an agent is added, an agent fails, or an agent is disconnected from the network). In any case, these averaging problems can be solved using a distributed average consensus method. One such method will be described in the following sections.

4.2 Distributed Average Consensus

The distributed averaging problem can be stated roughly as follows. Suppose we have a set of agents $\mathcal{N} = \{1, \ldots, \mathrm{N}\}$ connected together over a communication network, and each agent i has a value (or state) z_i. We would like for each agent in the set to reach a consensus on their average value,

$$\bar{x} = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} z_i, \tag{4.17}$$

simply by exchanging information with only a subset of its peers.

There exists a vast amount of literature devoted to a family of algorithms known as distributed average consensus, which solves this problem (see [11]–[13] and references therein). Consensus algorithms have had a long history, with contributions from a variety of research communities, including distributed computing [14], control systems [11], and social networks [15]. These algorithms have gained a lot of interest in the last decade for its potential use in the distributed coordination of multi-agent systems, which involves, among other things, formation control [16,17], distributed sensor fusion [18], flocking [19], clock synchronization [20], and multi-agent rendezvous [21].

In the following we describe a distributed average consensus algorithm, provide proof of its correctness, and demonstrate how it can be implemented on a resource-limited embedded system such as the DFA. Our modifications and implementation result in the following key properties:

- *Homogeneous Implementation.* Agents exchange information using a common data structure and employ a common set of rules.
- *Robust to Network Adversities.* Functions reliably in the case of asynchronous communication, loss of communication, and switching network topologies.
- *Unbiased Average.* The consensus value that can be reached is the precise average.
- *Dynamic Consensus.* Capable of tracking a time-varying average.
- *Well-Defined Tuning Parameters.* The convergence properties are well-defined and can be adjusted using a single tuning parameter.

Our multi-agent network can be modelled as an undirected and connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ consisting of a set of agents $\mathcal{N}$ and a set of edges $\mathcal{E} \subset \{\{i,j\} | i,j \in \mathcal{N} \text{ and } i \neq j\}$. Each edge $\{i,j\} \in \mathcal{E}$ represents a bidirectional communication link between two agents. The set of neighbours belonging to agent i is denoted by $\mathcal{N}_i = \{j | \{i,j\} \in \mathcal{E}\}$. For simplification, i and j may be used to respectively denote agent i and agent j in the remainder of this work.

Without loss of generality, each agent i stores a state variable $x_i(k) \in \mathbb{R}$ at an iteration step $k \geq 0$, and is initialized to $x_i(0) = z_i$. The states of all agents in the network can be compactly represented by a vector $\mathbf{x}(k) \in \mathbb{R}^{\mathrm{N}}$. An iterative method, which enables the state of all agents at $k = 0$ to converge on the average $\bar{x} = (1/\mathrm{N}) \sum_{i=1}^{\mathrm{N}} x_i(0)$, takes the following form:

$$\mathbf{x}(k+1) = (\mathbf{I} - \alpha \mathbf{L}) \mathbf{x}(k), \tag{4.18}$$

where α is a scalar tuning parameter and $\mathbf{L}$ is the Laplacian matrix of graph $\mathcal{G}$. This matrix captures the graph topology [22] and is defined as

$$L_{ij} = \begin{cases} d_i & i = j, \\ -1 & \{i,j\} \in \mathcal{E}, \\ 0 & \text{otherwise}, \end{cases} \tag{4.19}$$

where d_i denotes the degree of i (i.e. the number of neighbours). To guarantee convergence, the tuning parameter α must satisfy the following constraint [23]:

$$0 < \alpha < \frac{1}{d_{\max}}, \tag{4.20}$$

where $d_{\max}$ denotes the maximum degree of all agents in the network. The assumption made here is that agents require some prior knowledge about the network. This is reasonable for our application because an agent can only communicate with a fixed number of peers. In the case where the graph topology is known, one can compute the optimal tuning parameter α^*, which maximizes the rate of convergence [23]:

$$\alpha^* = \frac{2}{\lambda_1(\mathbf{L}) + \lambda_{\mathrm{N}-1}(\mathbf{L})}, \tag{4.21}$$

where $\lambda_1(\mathbf{L})$ and $\lambda_{\mathrm{N}-1}(\mathbf{L})$ represent the largest and second-smallest eigenvalue of $\mathbf{L}$, respectively.

This iterative method (4.18) can be written element-wise as

$$x_i(k+1) = x_i(k) + \alpha \sum_{j \in \mathcal{N}_i} (x_j(k) - x_i(k)), \tag{4.22}$$

which suggests that each agent i computes its new state $x_i(k+1)$ using its previous state $x_i(k)$ and the sum of current disagreements $x_j(k) - x_i(k)$ scaled by a factor α. A key thing to note is that the sum of the states $\sum_i^{\mathrm{N}} x_i(k)$ at any iteration k is always equal to the sum of the values $\sum_i^{\mathrm{N}} z_i$ stored at the agents, because any change made to $x_i(k)$ on behalf of the state of neighbour j is compensated by a reciprocal change to $x_j(k)$ on behalf of the state of i. This is a necessary condition for all states to converge to the desired average $\bar{x}$ and will become useful in the following.

In a manner similar to [24], one can introduce an additional variable $\delta_{ij}(k)$ for each neighbour j of agent i and re-write (4.22) as

$$x_i(k+1) = z_i + \alpha \sum_{j \in \mathcal{N}_i} \delta_{ij}(k+1), \tag{4.23}$$

$$\delta_{ij}(k+1) = \delta_{ij}(k) + (x_j(k) - x_i(k)), \tag{4.24}$$

where $\delta_{ij}(0)$ is zero for all i and j. Writing the algorithm in this manner extends its capabilities in two immediate ways:

1. *Dynamic Topology.* Agents may gracefully join or leave the network at any point in time. The mechanism that enables this is δ_{ij}, which stores the cumulative disagreement between agent i and j. If an agent leaves the network, all of its neighbours set the appropriate δ_{ij} to zero, such that its contribution is neutralized and the states converge to the appropriate average of values z_i in the new network. It can be shown that with this mechanism, the sum of the states x_i over all connected agents in the network at any given point in time is conserved, which is necessary for computing a precise average. This, however, assumes that an agent can detect when another agent leaves the network.

2. *Dynamic Average.* Although not explicitely written, the average quantity of interest may change with time. Take for example a sensor network where the target being sensed is time-varying. This is handled by the variable z_i, which denotes the measured quantity to be averaged over the network. This is referred to in literature as *dynamic consensus* [24].

One other subtle capability that is enabled by re-writing the algorithm in this way is that state updates (4.23) can now be performed at any time (which is enabled by the cummulative disagreement variable δ_{ij}) without compromising the end result. The inability to do this previously in (4.22) is in fact a major drawback concerning its implementation. In order to reach an average consensus, the algorithm defined in (4.22) implicitely requires the state information from each agent in the network to be successfully transmitted to its corresponding neighbour(s) prior to updating its state information. This imposes a form of synchronization of information among all agents, which requires a non-trivial implementation of communication acknowledgements between agents. A consequence of such an implementation is that this method is impeded by the slowest communication link. In comparison, the cumulative disagreement variable δ_{ij} of (4.23) – (4.24) acts as an information buffer, and loosens this synchronization constraint; it is now possible to perform a state update as soon as an agent finishes communicating with a neighbouring agent. We will demonstrate in Section 4.3 how this can be implemented.

Convergence

Of practical importance is the number of iterations necessary for our distributed average consensus method to come within a particular precision of the average. We define precision $\sigma(k)$ as the standard deviation of all state values $\mathbf{x}(k)$ in the network at iteration counter value k with respect to the average $\bar{\mathbf{x}}$,

$$\sigma(k) := \sqrt{\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}(x_i(k) - \bar{\mathbf{x}})^2}. \tag{4.25}$$

It can be shown from the result for *convergence time* in [23] that the number of iterations k_d needed to reach a desired precision σ_d from the network's initial precision $\sigma(0)$ is given by

$$k_d = \frac{\ln(\sigma_d/\sigma(0))}{\ln(\mathrm{r}_{\mathrm{asym}})}, \tag{4.26}$$

where the *asymptotic convergence factor* $\mathrm{r}_{\mathrm{asym}}$ is defined as

$$\mathrm{r}_{\mathrm{asym}} := \max\{1 - \alpha\lambda_{\mathrm{N}-1}(\mathbf{L}), \alpha\lambda_1(\mathbf{L}) - 1\}. \tag{4.27}$$

This factor increases with the algebraic connectivity of a network, which is described by the second smallest eigenvalue $\lambda_{\mathrm{N}-1}$ of the Laplacian matrix [25]. It may be worth mentioning that when comparing networks of the same size, the algebraic connectivity is lowest for path graphs [26], where for $\mathrm{N} \geq 2$, exactly two agents have a degree of one, and all other nodes have a degree of two. Thus, the value of $\mathrm{r}_{\mathrm{asym}}$ is bounded by those obtained from path graphs, see Figure 4.3. In the case where the optimal tuning parameter α^* is used, the asymptotic convergence factor can be expressed analytically as (see Appendix 4.A)

$$\mathrm{r}_{\mathrm{asym}} = \cos\left(\frac{\pi}{\mathrm{N}}\right). \tag{4.28}$$

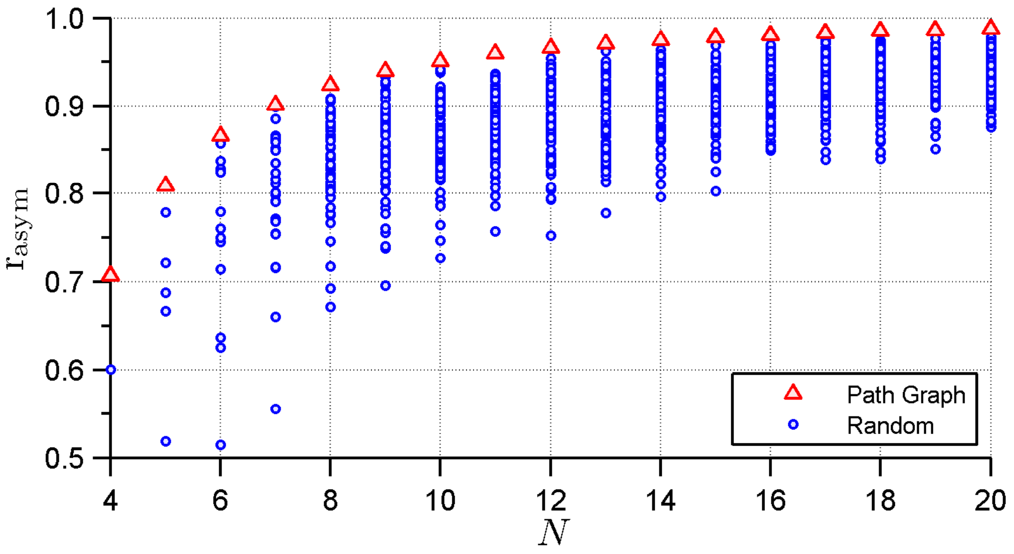


Figure 4.3 For mesh networks such as the DFA, the asymptotic convergence factor $\mathrm{r}_{\mathrm{asym}}$ generally increases with the size of the network N. The asymptotic convergence factor for path graphs (△) represent the upper bound; 100 randomly chosen configurations (○) were generated for each network size N.

4.3 Implementation: Distributed Average Consensus

In order to implement the distributed average consensus algorithm (4.23) – (4.24) described in the previous section, one needs to be aware of an assumption made implicit in its exposition. It assumes that prior to having agent i update its state, its current disagreement $\Delta_{ij}(k) := x_j(k) - x_i(k)$ with neighbour j must be reciprocated to agent j, i.e. $\Delta_{ij}(k) = -\Delta_{ji}(k)$. Another way of looking at it is that one must guarantee that the information from i is successfully transmitted to j, and vice-versa, prior to updating its state. To do this we implement a form of message-acknowledgements, but not in the traditional sense, which will be explained further on in this section. For simplicity, the agent's subscript notation is removed where it is clear.

To minimize the amount of information that is transferred, state information x is only passed in one direction between a pair of agents. Roughly speaking, i sends its state to j; j then uses this information along with its own state to compute its current disagreement Δ, which it then sends back to i. At the end of this exchange, both agents share a common disagreement Δ (differing only by a sign). To know whether an agent should send its state or its disagreement, each agent in the network is given a unique numerical identifier (ID) (this in any case is needed for other reasons); i and j will be used to represent an agent's ID, where the value of i is not j. Then,

1. if $j > i$, agent i sends its state x to j,
2. if $j < i$, agent i sends its disagreement Δ to j.

Since an agent can have more than one neighbour, it is possible for an agent to send both state x and disagreement Δ, depending on its ID and those of its neighbours.

Messages containing state x or disagreement Δ are sent back and forth between agent pairs repeatedly. The nature of this round-trip message exchange behaviour can be exploited to guarantee that messages are sucessfully communicated between agent pairs. Appended to each message is a sequence number, denoted SEQ. Each agent pair shares a sequence number that is independent of other agent pairs. This sequence number is incremented upon the successful completion of a round-trip message; only one agent in the pair increments this sequence number after sending a message and receiving a response from its other half. Once again, in order to do this, unique agent IDs are needed, where

1. if $j > i$, agent i increments its sequence number SEQ prior to starting an iteration,
2. if $j < i$, agent i does nothing with the sequence number.

Knowing this, agents can therefore predict the sequence number of the next message that it should receive. An incoming message with a correctly predictd sequence number indicates that the previously sent message was correctly received by its neighbour. This therefore acts as an implicit acknowledgement mechanism for messages that were properly received. For example, i sends a message appended with a sequence number to j; this sequence number is stored in the memory of i to verify future incoming messages. Agent j already has in memory a sequence number from a previous message that it received from i. It can therefore predict the expected sequence number in the following message from i. When j receives this message from i, it will compare the message's sequence number with its predicted value. If they do not correspond, the message will be discarded. If the predicted value is validated, j will use the data contained within this message to update its state. It will then copy the new sequence number to memory (to verify future incoming messages) and it will append this number to its next message to i. When i receives this message from j, it will verify the appended sequence number with the one stored in memory. If it is the same, the sequence number has made a round-trip and the data contained within the message is unique, which will be used to update its state. If the message cannot be verified, it is simply discarded.

To prevent against a deadlock behaviour (due to for example communication loss), a message containing the most recent data (i.e. state or disagreement) is sent at a fixed periodic interval; this can be accomplished using a timer interrupt. The use of sequence numbers ensure that messages are never re-used. By sending data periodically, we can also detect when agents join or leave the network by employing a watchdog timer on incoming messages. In the event that a watchdog is set (i.e. a message has not been received after some predefined amount of time), we classify that neighbour as being disconnected from the network and reset the appropriate cummulative disagreement variable δ to zero.

Our implementation can be divided into two components: a message sending routine (see Algorithm 1) and a message handling routine (see Algorithm 2). Each algorithm respresents a function that is executed periodically for each neighbouring agent (e.g. by means of a timer interrupt) and uses as input the most recent data. What is not shown in the Algorithms is an implementation of the watchdog timer, which is used to detect when agents join or leave the network. To see which part of a routine (denoted ①, ②, ③, and ④) is employed during sending and handling of messages, refer to Figure 4.4.

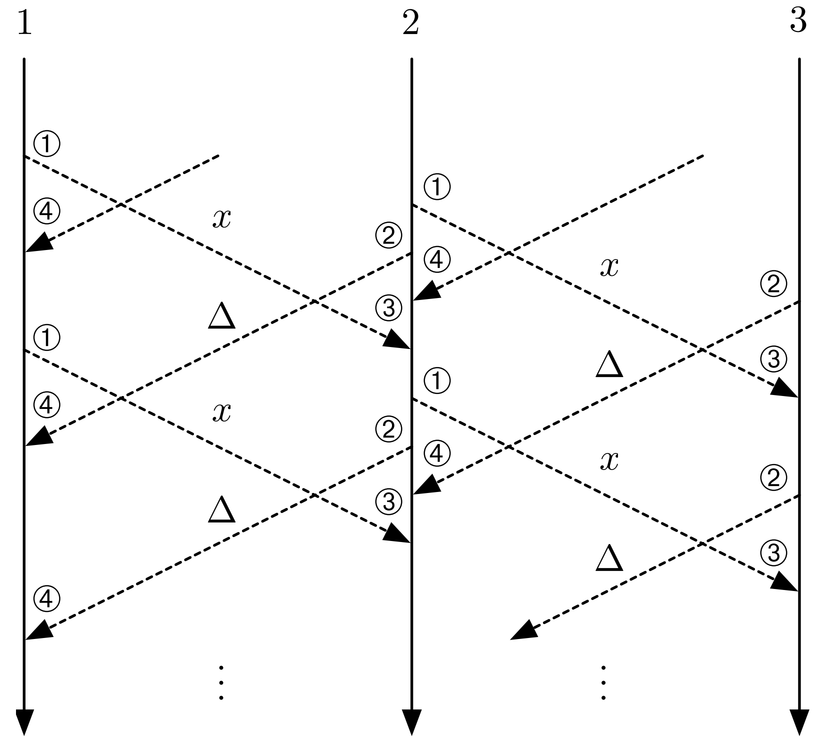


Figure 4.4 This figure illustrates the message sending and handling behaviour of a three-agent network $\{1, 2, 3\}$, connected in a line. The symbols ①, ②, ③, and ④ indicate the appropriate parts of the routines described in Algorithm 1 and Algorithm 2. Each agent $\{1, 2, 3\}$ sends a message to its neighbour(s) at a periodic interval. An agent with a lower ID than its neighbouring pair sends a state message x; an agent with a higher ID responds with its current disagreement Δ.

Simulation and Experiments

The distributed average consensus algorithm was implemented on the DFA in a manner described in the previous section. In this experiment, six agents were connected and disconnected as shown in Figure 4.5. The tuning parameter α was set to 0.165, which satisfies the condition (4.20) for all possible network topologies in this experiment. The value of

Algorithm 1 Message Sending Routine for agent i; executed at a periodic interval.

```
// Global Variables
$i$: Agent ID
$x$: State of $i$
$\Delta_j$: Current disagreement between $j$ and $i$
SEQ$_j$: Sequence number for messages exchanged between $j$ and $i$
// Input
$j$: Neighbour ID

function MsgSend($j$)
    if $j > i$ then
        DATA $\leftarrow x$                                  ▷ ①
    else
        DATA $\leftarrow \Delta_j$                           ▷ ②
    end if
    Send DATA
    Send SEQ$_j$
end function
```

this parameter is far from the optimal (4.21). Nevertheless, the experiment demonstrates our algorithm's ability to recover from real-time switching topologies (or loss of communication). These results were also compared to those obtained from our simulation environment. The results match well and thus validates our simulator, which is implemented in MATLAB. Our simulator accepts as input a Laplacian matrix and simulates the behaviour of each agent's state over a given number of iterations. It enables us to simulate a variety of network adversities, including: temporary loss of communication, switching topologies, time-varying average, and asynchronous updates. It enables us to quickly evaluate and illustrate the behaviour of our algorithm under a variety of conditions.

In a separate experiment, we again simulated a network of six agents with their states initialized as $\{1, \ldots, 6\}$ and demonstrated the ability of our algorithm to handle changes to its initial values $z_i \in \{1, \ldots, \mathrm{N}\}$ dynamically. To demonstrate this, we changed the initial value of one agent during the iterative procedure without having to restart the algorithm; the results are shown in Figure 4.6.

4.4 Implementation: Distributed Linear Least Squares

We describe here the implementation procedure for using the previously described distributed average consensus algorithm (4.23) – (4.24) to solve the distributed least squares problem (4.15) – (4.16), in order to estimate the tilt of the DFA. Each agent updates its measurement $\hat{m}_i$ periodically and independently of other agent, i.e. updates can occur

Algorithm 2 Message Handling Routine for agent i; executed at a periodic interval.

```
// Global Variables
$\alpha$: Tuning parameter
$z$: Initial state of $i$
$x$: State of $i$
$\delta_j$: Cumulative disagreement between $i$ and $j$
$\Delta_j$: Current disagreement between $i$ and $j$
SEQ$_j$: Sequence number stored at $i$ for $j$

// Inputs
$i$: Agent ID
$j$: Neighbour ID
DATA: State $x_j$ or current disagreement $\Delta_j$ of neighbour $j$
SEQ: Sequence number of the message from $j$

function MsgHandler($j$, DATA, SEQ)
    if $j < i$ then
        // DATA = $x_j$
        if SEQ$_j + 1 =$ SEQ then                                  ▷ ③
            $\Delta_j \leftarrow$ DATA $- x$
            $\delta_j \leftarrow \delta_j + \Delta_j$
            $x \leftarrow z + \alpha \sum_{j \in \mathcal{N}_i} \delta_j$
            SEQ$_j \leftarrow$ SEQ
        else
            // Do nothing
        end if
    else
        // DATA = $\Delta_j$
        if SEQ$_j =$ SEQ then                                      ▷ ④
            $\Delta_j \leftarrow -$DATA
            $\delta_j \leftarrow \delta_j + \Delta_j$
            $x \leftarrow z + \sum_{j \in \mathcal{N}_i} \delta_j$
            SEQ$_j \leftarrow$ SEQ $+ 1$
        else
            // Do nothing
        end if
    end if
end function
```

asynchronously. Between each new measurement, the matrix $\mathbf{A}$ and vector $\mathbf{b}$ are computed iteratively using our proposed distributed average consensus method. The update rate of this distributed tilt estimate is a function of the algebraic connectivity of the network, the network communication rate, and the desired number of iterations given by (4.26). For the initial conditions, vector $\mathbf{c}_i$ is initialized with agent i's sensor position,

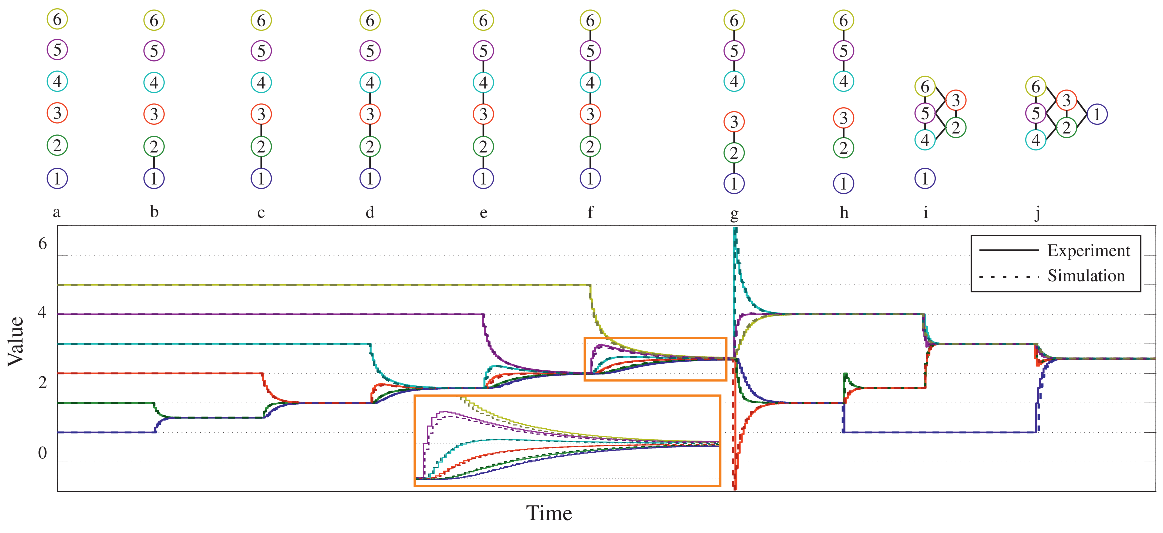


Figure 4.5 Our algorithm gracefully recovers from switching topologies. Six agents have their states initialized as $\{1, \dots, 6\}$ and start off disconnected. They are connected on by one into a line topology $(b) - (f)$. The configuration was then divided first into two (g) and after into three (h) separate clusters. The three clusters were finally assembled into a single triangle-like topology (j). The states of the agents converge in each case to the appropriate average. Also shown are simulation results (*dashed*), which correspond well with the experiments (*solid*) (see also magnification of a selected region of the plot in the lower-middle inset), thus validating our simulation environment.

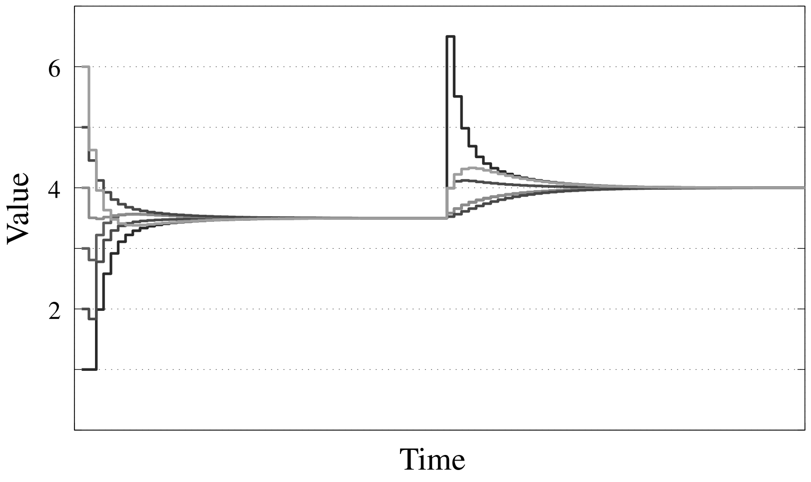
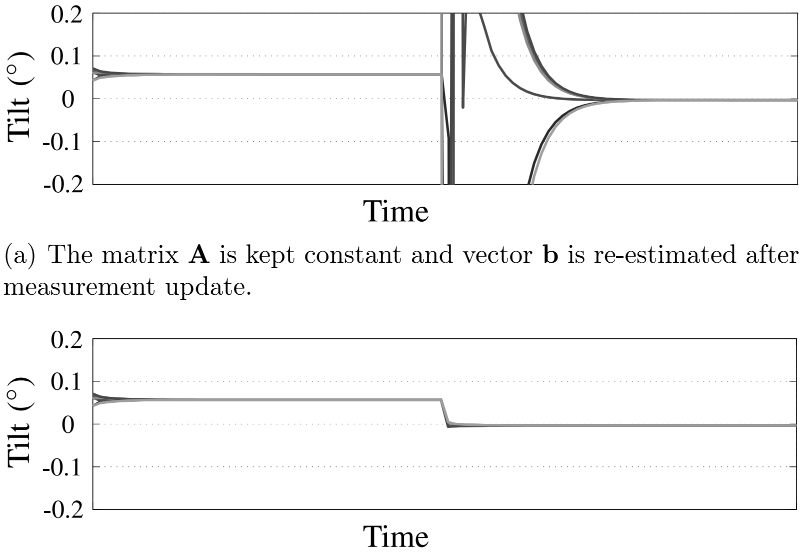


Figure 4.6 Upon converging to its average of 3.5, where the state of six agents were initialized as $\{1, \dots, 6\}$, one agent changes its initial value z_i from 1 to 4; without having to reset the algorithm, all states converge to the new average of 4.

and $\hat{m}_i$ is set to the most recent distance measurement.

For implementation purposes, as long as the configuration of the vehicle remains fixed and the number of iterations needed to estimate the state for each new set of measurements remains constant, there is no need to re-estimate $\mathbf{A}$ (which in this case would be constant) for each new set of measurements $\mathbf{m}$. One can take advantage of this by computing its value only once, and then using the result for future evaluations of the state estimate $\hat{\mathbf{s}}$. In this case, the only information that needs to be shared is $\mathbf{b}$. In doing so, the intermediate state estimates $\hat{\mathbf{s}}^*$ do not follow the smooth convergence pattern as they would if both $\mathbf{A}$ and $\mathbf{b}$ were updated, however the end result is identical, see Figure 4.7.



(a) The matrix **A** is kept constant and vector **b** is re-estimated after a measurement update.

(b) Both matrix **A** and vector **b** are re-estimated after a measurement update.

Figure 4.7 This plot compares the result of the distributed tilt estimate when **A** is kept constant (a) and when **A** is re-computed (b). Both techniques yield the same result, but the convergence is smoother in the second case.

Our proposed distributed tilt estimate was tested on the DFA in a 6-agent configuration similar to (*j*) in Figure 4.5, which has an algebraic connectivity of $\lambda_{\mathrm{N}-1} = 1.70$. In this particular implementation we used the optimal tuning parameter $\alpha^* = 0.286$, a communication rate of 20 messages per second, and four iterations for each new measurement – this amounts to a state update rate of 5 Hz. The IR distance measurement sensors of each agent was calibrated beforehand, and the vehicle was positioned and tilted by hand in the workspace of a 3D motion capture system (Vicon MX)[1] – the measurements of which we use as ground truth. The results, shown in Figure 4.8, verify the functionality of our proposed distributed tilt method and demonstrates that the estimate is relatively precise in comparison to ground truth measurements.

[1]The 3D motion capture system that is used here employs an IR wavelength that is different from the IR light emitted by the IR distance measurement sensors, resulting in unobservable interference.

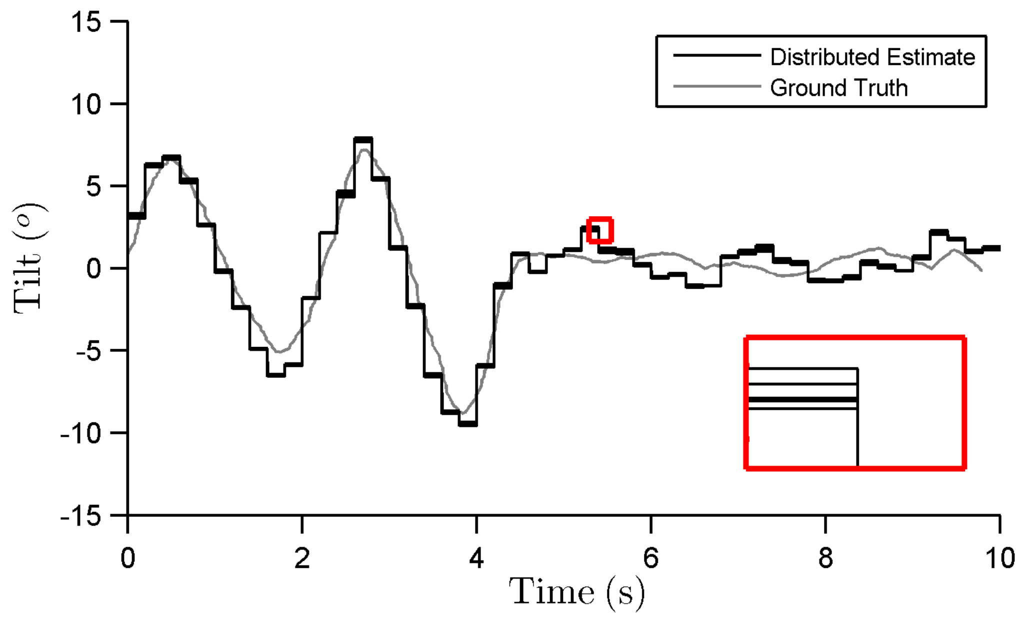


Figure 4.8 Shown here is a plot of the tilt estimate along one of the rotational degrees of freedom; the inset image shows a magnification of a particular area in the plot (□), illustrating the deviations between estimates. The distributed tilt estimate of each agent compares well with ground truth measurements.

4.5 Real-Time Distributed Tilt Estimation

We have demonstrated in the previous section that the distributed tilt estimate is non-biased and relatively precise, however the update rate is generally slow. Moreover, the estimate will be delayed from the time when the set of distance measurements are first made to the time when the agents reach a consensus about their estimate.

We would nevertheless like to use this tilt estimate for real-time purposes, such as controlled flight [27]. In order to provide a reasonably fast unbiased tilt estimate, the distributed tilt estimate can be combined with angular rate measurements obtained from a agent's 3-axis rate-gyroscope. A calibrated rate-gyroscope (i.e. removal of offset error) on its own could be used to estimate tilt by time-integration of its angular rate measurements. However this estimate is only useful short-term. Bias in the angular rate measurements (due to e.g. temperature effects and quantization error) will result in unbounded drift in the tilt estimate, see Figure 4.9. For the experimental results shown here, the configuration of the vehicle that was used to test and verify our algorithm as described in Section 4.4 is also used here.

Kalman Filter

A Kalman filter, which takes into account time-delayed information, can be used to optimally combine both the distributed tilt estimate with Euler angle rates; the latter can be obtained through the appropriate kinematic transformation of rate-gyroscope

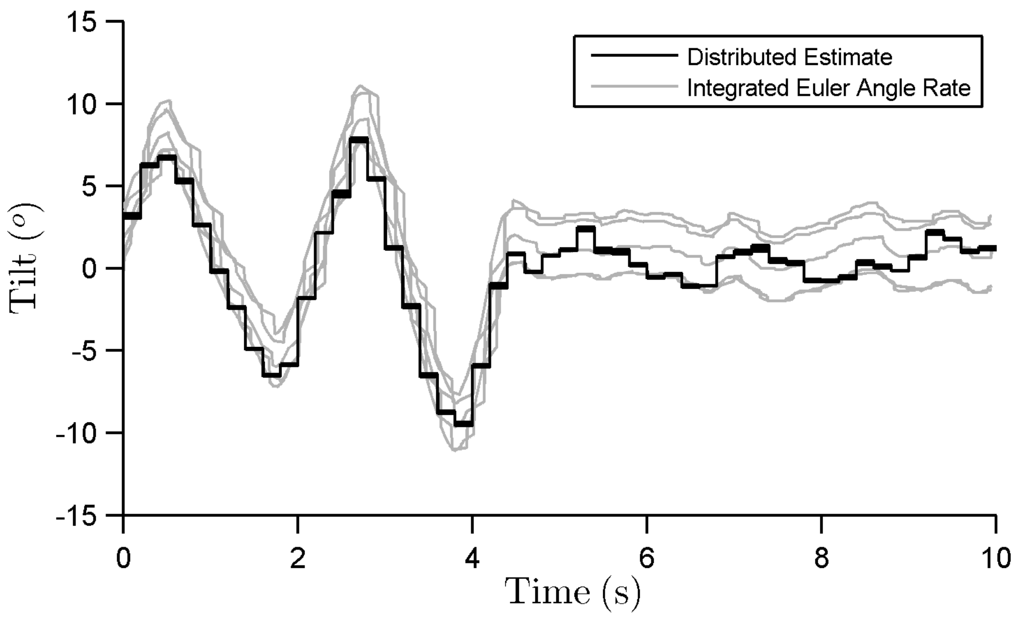


Figure 4.9 The distributed tilt estimate does not drift over time. Bias in the angular rate measurements, however, result in unbounded drift when time-integrating Euler angle rate estimates from a rate-gyroscope.

measurements (see Appendix 4.B), see Figure 4.10. For simplicity, we will present the Kalman Filter for γ only; the results for β follow accordingly. Both tilt and bias error of the rate-gyroscope are estimated by the filter.

Euler angle rate $\hat{\dot{\gamma}}$ and bias error b can be modelled, in continuous time, as:

$$\hat{\dot{\gamma}}(t) = \dot{\gamma}(t) + b(t) + v_{\dot{\gamma}}(t), \tag{4.29}$$

$$\dot{b}(t) = v_{\dot{b}}(t) \tag{4.30}$$

where $\dot{\gamma}$ is the true Euler angle rate corrupted by bias error b, and zero-mean white Gaussian noise $v_{\dot{\gamma}}$ with variance $\sigma_{\dot{\gamma}}^2 = 5.4 \times 10^{-4}\,\text{rad/s}$ determined from experiments; and the dynamics of the bias error $\dot{b}$ is modelled as zero-mean white Gaussian noise $v_{\dot{b}}$ with variance $\sigma_{\dot{b}}^2 = 0.01\,\text{rad/s}$ [28]. The intuition for the latter is that bias can be modelled as a random walk.

The distributed tilt estimate $\hat{\gamma}_d$ can be modelled as the following

$$\hat{\gamma}_d(t) = \gamma(t) + v_{\gamma}(t), \tag{4.31}$$

where γ is the true tilt of the vehicle corrupted by zero-mean white Gaussian noise v_γ with variance σ_γ^2, which was previously shown in (4.14) to be a function of the vehicle's

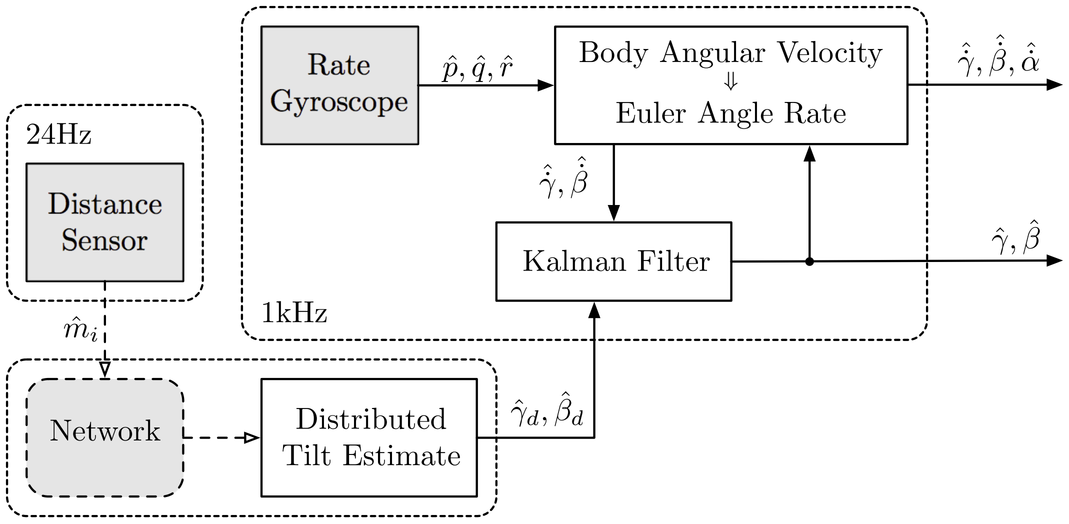

24Hz
Distance Sensor
$\hat{m}_i$
Rate Gyroscope
$\hat{p}, \hat{q}, \hat{r}$
Body Angular Velocity ⇓ Euler Angle Rate
$\hat{\dot{\gamma}}, \hat{\dot{\beta}}, \hat{\dot{\alpha}}$
$\hat{\dot{\gamma}}, \hat{\dot{\beta}}$
Kalman Filter
$\hat{\gamma}, \hat{\beta}$
1kHz
Network
Distributed Tilt Estimate
$\hat{\gamma}_d, \hat{\beta}_d$

Figure 4.10 A Kalman filter is used to combine Euler angle rates $(\hat{\dot{\gamma}}, \hat{\dot{\beta}})$, which is obtained through the appropriate kinematic transformation of rate-gyroscope measurements, and the distributed tilt estimate $(\hat{\gamma}_d, \hat{\beta}_d)$. Euler angle rates are a function of the body angular velocities (p, q, r) and Euler angles of the vehicle (see Appendix 4.B). Euler angles are approximated using the optimal estimate from the previous time step. This approximation works well in practice because the update rate of the optimal estimate is relatively fast (1 kHz).

configuration and temporal distance measurement variance, the standard deviation for which is $\sigma_m = 3.24 \times 10^{-3}$ m, dtermined from experiments.

In discrete-time, the predicted tilt and bias error (or Kalman filter state) $\hat{\mathbf{x}}_{n|n-1} = (\hat{\gamma}, \hat{b})$ and associated covariance matrix $\mathbf{P}_{n|n-1}$ for an estimate at time step n, using measurements up to time $n-1$, may be expressed using the standard Kalman filter prediction equations [29],

$$\hat{\mathbf{x}}_{n|n-1} = \mathbf{F}\hat{\mathbf{x}}_{n-1|n-1} + \mathbf{B}\mathbf{u}_n, \tag{4.32}$$

$$\mathbf{P}_{n|n-1} = \mathbf{F}\mathbf{P}_{n-1|n-1}\mathbf{F}^{\mathrm{T}} + \mathbf{Q}. \tag{4.33}$$

Here, input $\mathbf{u}_n$ is the measured Euler angle rate $\hat{\dot{\gamma}}_n$ at time step n, and the state update matrix $\mathbf{F}$, input matrix $\mathbf{B}$, and covariance matrix $\mathbf{Q}$ are respectively defined as

$$\mathbf{F} = \begin{bmatrix} 1 & -\mathrm{T}_s \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathrm{T}_s \\ 0 \end{bmatrix}, \tag{4.34}$$

$$\mathbf{Q} = \begin{bmatrix} (\mathrm{T}_s \cdot \sigma_{\dot{\gamma}})^2 & 0 \\ 0 & (\mathrm{T}_s \cdot \sigma_{\dot{b}})^2 \end{bmatrix}. \tag{4.35}$$

In this case, the prediction update period $\mathrm{T}_s = 1 \times 10^{-3}\,\mathrm{s}$ is the sampling period of the rate-gyroscope.

In the classical implementation of the Kalman filter, a correction (or update) in the state estimate is made as soon as a state measurement is available; this will in turn cause a reduction in the state's covariance estimate. In the case described here, the *measurement* at time step n is the distributed tilt estimate $\hat{\gamma}_{d,n}$. However, this estimate is only accurate at the time when the distance measurements were made, which is denoted as time step n_0, see Figure 4.11. Therefore, the update equations of the Kalman filter should reflect the fact that the distributed tilt estimates are affected by a delay of $n - n_0$.

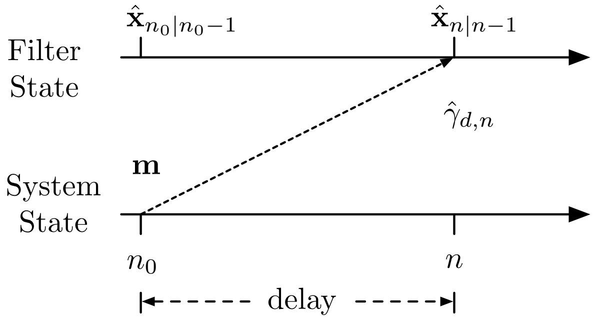


Figure 4.11 The distributed tilt estimate $\hat{\gamma}_{d,n}$ is only accurate at the time when the distance measurements $\mathbf{m}$ were made, which is denoted at time step n_0.

Methods that take into account fusion of delayed measurements have been derived in [30]. However, the computational complexity of its solution is comparable to recalculating the Kalman filter through the delay of n time steps, which is not scalable and is infeasible for implementation in an embedded system with limited computational resources. Another solution, which is proposed in [31], is to take the measurement $\hat{\gamma}_{d,n}$, which again is only accurate at time step n_0, and extrapolate this value to the present time step n. A Kalman gain, which takes into account this extrapolation, can be used to optimally combine the extrapolated measurements with the predicted states. Although this method does not guarantee optimality under all conditions (the extrapolation itself is not necessarily optimal), this method is straightforward, performs reasonably well, and is a computationally cheap way of accounting for delays.

Assuming that the optimal state $\hat{\mathbf{x}}_{n_0|n_0-1}$ from time step n_0 has been stored, the delayed measurement (or distributed tilt estimate) $\hat{\gamma}_{d,n}$ can be extrapolated to $\hat{\gamma}^{ext}_{d,n}$ at the present time step by the following:

$$\hat{\gamma}^{ext}_{d,n} = \hat{\gamma}_{d,n} + \underbrace{\mathbf{H}\hat{\mathbf{x}}_{n|n-1} - \mathbf{H}\hat{\mathbf{x}}_{n_0|n_0-1}}_{\text{extrapolation}}, \tag{4.36}$$

where the observation model is given by $\mathbf{H} = [1\ 0]$. The extrapolation is essentially made using the predicted estimates from time step n_0 to time step n. This is then used in the update phase of a Kalman filter that compensates for time-delayed measurements,

$$\hat{\mathbf{x}}_{n|n} = \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n \underbrace{\left[\hat{\gamma}^{ext}_{d,n} - \mathbf{H}\hat{\mathbf{x}}_{n|n-1}\right]}_{\text{residual}}, \tag{4.37}$$

$$\mathbf{P}_{n|n} = \mathbf{P}_{n|n-1} - \mathbf{K}_n \mathbf{H} \mathbf{P}_{n_0|n_0-1} \mathbf{M}^{\mathrm{T}}. \tag{4.38}$$

The optimal Kalman gain $\mathbf{K}_n$ in this case is given by

$$\mathbf{K}_n = \mathbf{M} \underbrace{\mathbf{P}_{n_0|n_0-1}\mathbf{H}^{\mathrm{T}} \left[\mathbf{H}\mathbf{P}_{n_0|n_0-1}\mathbf{H}^{\mathrm{T}} + \mathbf{R}\right]^{-1}}_{\text{classical Kalman gain at time step } n_0}, \tag{4.39}$$

where $\mathbf{R} = \sigma_\gamma^2$ and

$$\mathbf{M} = \prod_{i=0}^{n-n_0-1} \left(\mathbf{I} - \mathbf{K}_{n-i}\mathbf{H}\right)\mathbf{F}. \tag{4.40}$$

When the delay n is zero ($n_0 = n$), $\mathbf{M} = \mathbf{I}$ and the Kalman gain (4.39) and covariance update (4.38) reduces to the classical form.

Experiment

To validate our approach, we implemented our real-time distributed tilt estimation algorithm on the DFA. Static calibration parameters of the rate-gyroscope such as scale factor, misalignment, and non-orthogonality [32] are assumed to have been calibrated out in an earlier sensor calibration step and that the sensor's sensitivity to linear acceleration is negligible. We positioned and tilted a 6-agent vehicle by hand and used the 3D motion capture system, described in Section 4.5 as ground truth. Our approach works well in practice; rate-gyroscope drift, as well as low data rate and latency in the distributed tilt estimate are eliminated in the optimal tilt estimate, see Figure 4.12.

4.6 Conclusions

This chapter presented a generalized method for estimating the tilt of a rigid body using a network of distance measurement sensors placed at known positions on the body on interest. We demonstrated how this can be cast more generally as a linear least squares problem and proposed a method for solving this problem on a distributed system architecture, such as the Distributed Flight Array. This involves using a class of algorithms

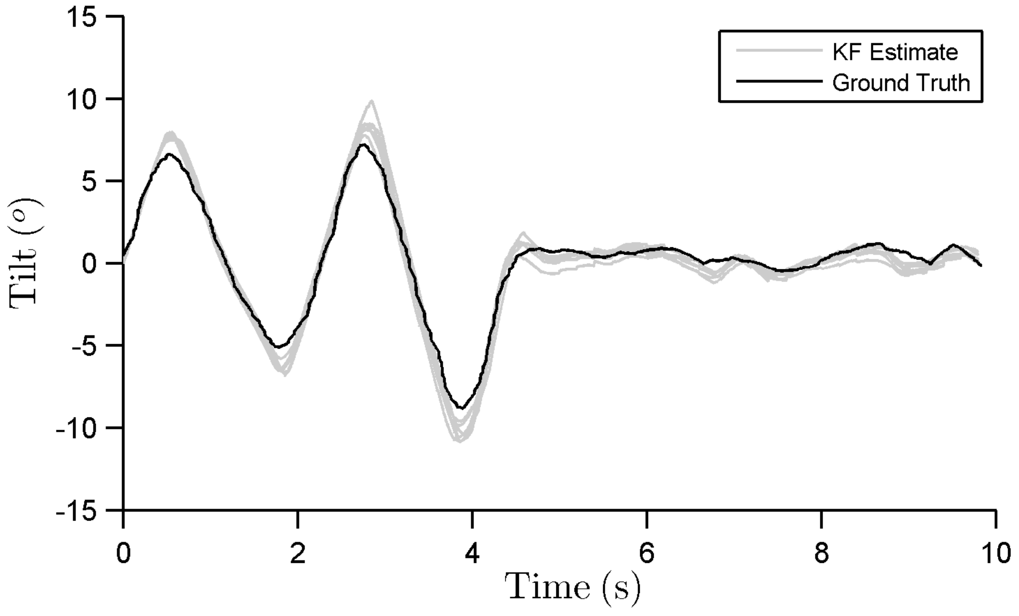


Figure 4.12 A 6-agent configuration was tilted by hand over a uniformly flat surface. The Kalman filter (KF) estimate for all six agents show good results with very little spread between estimates.

known in literature as distributed average consensus. We therefore proposed a distributed average consensus algorithm, which addresses a variety of practical issues, and we provide an implemention procedure tailored for a resource-limited embedded system, such as our testbed.

We then demonstrate with simulation and experiment the validity of our approach, that the tilt estimate approaches the linear optimal solution, and performs sufficiently well. This approach is then combined with angular rate estimates obtained from rate-gyroscope measurements in order to obtain a real-time, non-drifting tilt estimate of the vehicle, which can be used for feedback-control purposes.

Future work may build upon this method, by including an approach to detect and handle outliers in the measurements, which would occur when the flat-surface assumption is violated.

4.A Appendix: Asymptotic Convergence Factor for Path Graphs

The asymptotic convergence factor can be expressed through the eigenvalues of the Laplacian matrix [23] as

$$\mathrm{r_{asym}} = \max\{1 - \alpha\lambda_{\mathrm{N}-1}(\mathbf{L}), \alpha\lambda_1(\mathbf{L}) - 1\}, \tag{4.A.1}$$

where $\lambda_1(\mathbf{L}) \geq \lambda_2(\mathbf{L}) \geq \cdots \geq \lambda_N(\mathbf{L}) = 0$ are the ordered eigenvalues. Substituting this expression with the optimal tuning parameter α^* in (4.21) yields

$$\mathrm{r}_{\mathrm{asym}} = \frac{\lambda_1(\mathbf{L}) - \lambda_{\mathrm{N}-1}(\mathbf{L})}{\lambda_1(\mathbf{L}) + \lambda_{\mathrm{N}-1}(\mathbf{L})}. \tag{4.A.2}$$

The eigenvalues of a path graph's Laplacian matrix can be expressed analytically as [33]

$$\lambda_k = 2\left((1 - \cos\left(\frac{\pi(\mathrm{N} - k)}{\mathrm{N}}\right)\right). \tag{4.A.3}$$

Substituting (4.A.2) with (4.A.3) and employing trigonometric identities yields

$$\mathrm{r}_{\mathrm{asym}} = \cos\left(\frac{\pi}{\mathrm{N}}\right). \tag{4.A.4}$$

4.B Appendix: Transforming Rate-Gyroscope Measurements to Euler Angle Rates

The body angular velocities of the vehicle (p, q, r), which can be measured directly by each agent's onboard rate-gyroscope, can be converted to Euler angle rates $(\dot{\gamma}, \dot{\beta}, \dot{\alpha})$ through the following transformation [8]:

$$\begin{bmatrix} \dot{\gamma} \\ \dot{\beta} \\ \dot{\alpha} \end{bmatrix} = \mathbf{T}(\gamma, \beta) \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4.B.1}$$

where $\mathbf{T}(\gamma, \beta)$ is a nonlinear transformation of the form

$$\mathbf{T}(\gamma, \beta) := \begin{bmatrix} 1 & \sin\gamma\tan\beta & \cos\gamma\tan\beta \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma\sec\beta & \cos\gamma\sec\beta \end{bmatrix}. \tag{4.B.2}$$

Since the DFA operates around equilibrium, the transformation (4.B.2) can be reduced to the following linear transform $L\{\,\cdot\,\}$ assuming small angle approximations:

$$L\{T(\gamma,\beta)\} := \begin{bmatrix} 1 & 0 & \beta \\ 0 & 1 & -\gamma \\ 0 & \gamma & 1 \end{bmatrix}. \tag{4.B.3}$$

Euler angle rates $(\dot{\gamma}, \dot{\beta}, \dot{\alpha})$ are therefore a function of the Euler angles (γ, β) and the body angular velocities (p, q, r) of the vehicle.

References

[1] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Towards autonomous indoor micro VTOL," *Autonomous Robots*, vol. 18, no. 2, pp. 171–183, 2005.

[2] A. Tayebi and S. McGilvray, "Attitude stabilization of a VTOL quadrotor aircraft," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 3, pp. 562–571, 2006.

[3] B. Taylor, C. Bil, S. Watkins, and G. Egan, "Horizon sensing attitude stabilisation: A VMC autopilot," in *Proceedings of the International UAV Systems Conference*, vol. 31, Bristol, UK, 2003.

[4] P. Corke, "An inertial and visual sensing system for a small autonomous helicopter," *Journal of Robotic Systems*, vol. 21, no. 2, pp. 43–51, 2004.

[5] R. Mahony, T. Hamel, and J. M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[6] P. J. Bristeau, F. Callou, D. Vissière, and N. Petit, "The navigation and control technology inside the AR. Drone micro UAV," in *Proceedings of the International Federation of Automatic Control World Congress*, vol. 18, no. 1, Aug. 2011, pp. 1477–1484.

[7] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A system for autonomous flight using onboard computer vision," in *IEEE International Conference on Robotics and Automation.* IEEE, 2011, pp. 2992–2997.

[8] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Prentice Hall, Aug. 2004.

[9] R. Oung and R. D'Andrea, "The Distributed Flight Array: Design, implementation, and analysis of a modular vertical take-off and landing vehicle," *International Journal of Robotics Research, Under Review (Since Jan. 2013).*

[10] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proceedings of the International Symposium on Information Processing in Sensor Networks*, 2005, pp. 63 – 70.

[11] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

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

[13] F. Garin and L. Schenato, "A survey on distributed estimation and control applications using linear consensus algorithms," *Networked Control Systems*, pp. 75–107, 2011.

[14] N. A. Lynch, *Distributed Algorithms.* Morgan Kaufmann, 1996.

[15] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[16] R. Teo, D. M. Stipanovic, and C. J. Tomlin, "Decentralized spacing control of a string of multiple vehicles over lossy datalinks," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 469–473, 2010.

[17] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 443–455, 2004.

[18] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proceedings of the IEEE Conference on Decision and Control.* IEEE, 2007, pp. 5492–5498.

[19] N. Moshtagh and A. Jadbabaie, "Distributed geodesic control laws for flocking of nonholonomic agents," *IEEE Transactions on Automatic Control*, vol. 52, no. 4, pp. 681–686, 2007.

[20] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 214–226, 2006.

[21] Z. Lin, B. Francis, and M. Maggiore, "Getting mobile autonomous robots to rendezvous," *Control of Uncertain Systems: Modelling, Approximation, and Design*, pp. 119–137, 2006.

[22] R. Merris, "Laplacian matrices of graphs: A survey," *Linear Algebra and its Applications*, vol. 197, pp. 143–176, 1994.

[23] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, Sept. 2004.

[24] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Distributed sensor fusion using dynamic consensus," in *Proceedings of the International Federation of Automatic Control World Congress*, 2005, pp. 199–205.

[25] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[26] M. W. Newman, "The laplacian spectrum of graphs," Ph.D. dissertation, University of Manitoba, Winnipeg, Canada, July 2000.

[27] R. Oung, M. Picallo, and R. D'Andrea, "A parameterized control methodology for a modular flying vehicle," in *Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems*, Algarve, Portugal, Oct. 2012, pp. 532–538.

[28] R. L. Hammon, "An application of random process theory to gyro drift analysis," *IRE Transactions on Aeronautical and Navigational Electronics*, no. 3, pp. 84–91, 1960.

[29] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, ser. Dover Books on Electrical Engineering Series. Dover Publications, 2005.

[30] H. L. Alexander, "State estimation for distributed systems with sensing delay," *SPIE Data Structures and Target Classification*, vol. 1470, pp. 103–111, Aug. 1991.

[31] T. D. Larsen, N. A. Andersen, and O. Ravn, "Incorporation of time delayed measurements in a discrete-time Kalman filter," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 4, Tampa, USA, Dec. 1998, pp. 3972–3977.

[32] D. Titterton, J. Weston, D. H. Titterton, and J. L. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. IET, 2004.

[33] A. E. Brouwer and W. H. Haemers, *Spectra of Graphs.* Springer, 2011.

5 Decentralized Flight Control Strategy

One of the key challenges in modular robotics is control design. The majority of past research has focused on hardware and software challenges [1]–[3]; a summary of such work can be found in [4, 5]. The challenge in designing control algorithms for such systems is its size. Such systems can consist of hundreds or thousands of individual elements, each of which is capable of moving and interacting dynamically, leading to very rich and often unexpected behaviour (e.g. [6]–[14]). This makes such systems difficult to predict and/or control using traditional methods. To address the scalability issue, we employ distributed and decentralized control methods. Such a control system should be scalable, robust to single point failures, and adaptable to arbitrary configurations. Research in modular robotics that has taken a step in this direction includes [15] and [16].

This chapter presents a generalized tractable methodology for controlling any flight-feasible configuration of the Distributed Flight Array (DFA). We describe a parameterized, decentralized controller for an arbitrary configuration of the DFA that enables it to hover. By decentralized, we mean that the controller relies only on its local sensors for control feedback and that no information is shared between modules during flight. This results in a very scalable and straightforward implementation of the controller.

The tuning parameters for this controller can then be optimized for a given performance metric while taking into account physical constraints of the system. This decentralized controller is compared against two idealized cases: (1) a centralized architecture, where each module has direct sensor feedback from all modules in the vehicle, and (2) the centralized $\mathcal{H}_2$ optimal controller, which is used as a benchmark for performance. We show that it is possible to parameterize the configuration space of the DFA using only a few variables and demonstrate that a relatively simple set of functions can be used to map a vehicle's configuration to an appropriate set of control parameters.

Outline

Beginning with Section 5.1, we will describe the vehicle's dynamic model, similar to that encountered in micro aerial vehicles (see [17] and [18]), but generalized to an N-rotor system of arbitrary configuration. In Section 5.2 we describe how these dynamics scale with the size and configuration of the vehicle. It will be shown how the configuration of a vehicle can be parameterized by only a few variables.

The second part of this chapter will begin with Section 5.3, where the onboard estimation strategy used by the DFA will be described. In Section 5.4, we characterize the exogenous disturbances that are expected to excite our system during flight. This will become important later in optimizing the parameters of our control strategy, which is described in Section 5.5, in order to achieve best performance. Our control method, previously developed in [19], will be summarized and built upon in Section 5.5. We will then demonstrate in Section 5.6 a generalized procedure that can be used to automatically compute the parameters of our control strategy that achieves best performance subject to physical constraints.

The final part of this chapter starts in Section 5.7 by analyzing the flight performance of our vehicle for various configurations and sizes. Our analysis leads to a method, described in Section 5.8, that can be used to efficiently store, on board, the near-optimal control tuning parameters. We validate our strategy in Section 5.9 and demonstrate experimental results for up to twelve modules flying together in a variety of configurations. Concluding remarks are made in Section 5.10.

5.1 Dynamics Model

The DFA is modeled as a rigid body that is capable of producing multiple independently controlled positive thrust vectors directed upwards and perpendicularly away from the plane of the vehicle. These thrust vectors of magnitude f_i are positioned at (x_i, y_i) with respect to a body-fixed coordinate frame B, where the subscript $i \in \{1 \dots N\}$ is used to uniquely identify a module, see Figure 5.1.1. It is assumed here that each module is identical in mass and size and that each is capable of producing enough thrust to lift its own weight mg, where m is the mass of a single module and g is the acceleration due to gravity. The rigid body assumption greatly simplifies our system by discarding all inter-module dynamic behaviour, which in reality exists due to non-ideal elastic behaviour of the chassis, but is assumed to be of little consequence here.

The body frame B is positioned at the vehicle's center of mass and is aligned in a right-hand coordinate system with the principal axes of rotation, where the z-axis points upward. The orientation of B with respect to an inertial frame I is described by ZYX-Euler angles, which is commonly used in the aeronautics field to represent yaw α, pitch β, and roll γ acting first around the z-axis, followed by the y-axis, and lastly around the x-axis, respectively [20]. The orientation of B with respect to I can therefore be expressed by the rotation matrix

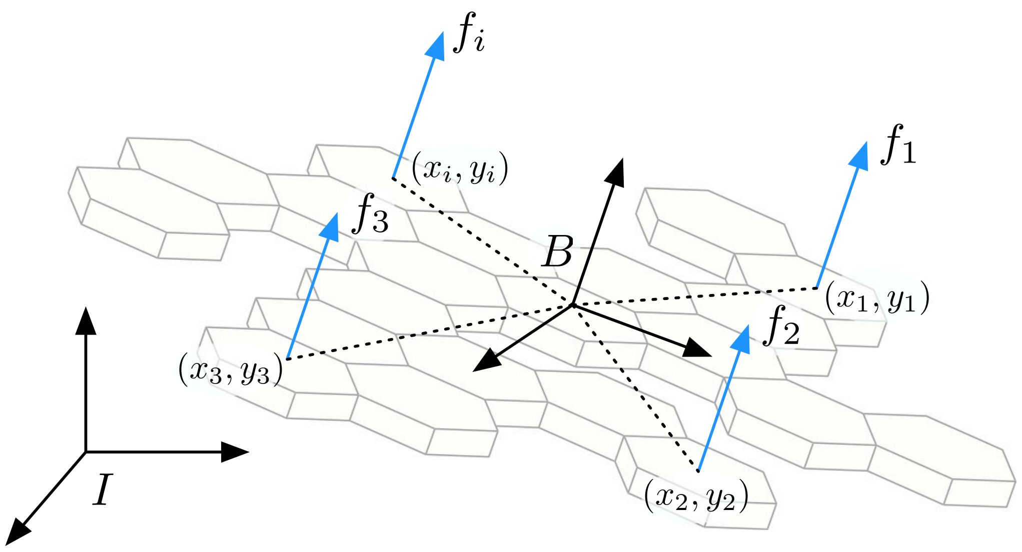


Figure 5.1.1 The DFA's body-fixed coordinate frame B is located at its center of mass and is aligned with the principal axes of rotation. The orientation of the vehicle with respect to an inertial frame I can be described by a set of ZYX-Euler angles. Control of the vehicle is achieved by varying the thrust f_i produced by module i located at (x_i, y_i).

$$
\begin{aligned}
{}^{I}_{B}\mathbf{R}_{\mathrm{zyx}} &= \mathbf{R}_{\mathrm{z}}(\alpha)\mathbf{R}_{\mathrm{y}}(\beta)\mathbf{R}_{\mathrm{x}}(\gamma), \\
&= \begin{bmatrix} \mathrm{c}_\alpha \mathrm{c}_\beta & \mathrm{c}_\alpha \mathrm{s}_\beta \mathrm{s}_\gamma - \mathrm{s}_\alpha \mathrm{c}_\gamma & \mathrm{c}_\alpha \mathrm{s}_\beta \mathrm{c}_\gamma + \mathrm{s}_\alpha \mathrm{s}_\gamma \\ \mathrm{s}_\alpha \mathrm{c}_\beta & \mathrm{s}_\alpha \mathrm{s}_\beta \mathrm{s}_\gamma + \mathrm{c}_\alpha \mathrm{c}_\gamma & \mathrm{s}_\alpha \mathrm{s}_\beta \mathrm{c}_\gamma - \mathrm{c}_\alpha \mathrm{s}_\gamma \\ -\mathrm{s}_\beta & \mathrm{c}_\beta \mathrm{s}_\gamma & \mathrm{c}_\beta \mathrm{c}_\gamma \end{bmatrix},
\end{aligned} \tag{5.1.1}
$$

where c and s are shorthand forms of cosine and sine, respectively, and

$$
\mathbf{R}_{\mathrm{z}}(\alpha) := \begin{bmatrix} \mathrm{c}_\alpha & -\mathrm{s}_\alpha & 0 \\ \mathrm{s}_\alpha & \mathrm{c}_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.1.2}
$$

$$
\mathbf{R}_{\mathrm{y}}(\beta) := \begin{bmatrix} \mathrm{c}_\beta & 0 & \mathrm{s}_\beta \\ 0 & 1 & 0 \\ -\mathrm{s}_\beta & 0 & \mathrm{c}_\beta \end{bmatrix}, \tag{5.1.3}
$$

$$
\mathbf{R}_{\mathrm{x}}(\gamma) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_\gamma & -\mathrm{s}_\gamma \\ 0 & \mathrm{s}_\gamma & \mathrm{c}_\gamma \end{bmatrix}. \tag{5.1.4}
$$

Force and Torque Generation

A module can use its rotor to generate thrust and simultaneously a reaction torque. The latter is generated mainly by the induced aerodynamic drag and, to some extent, the change in angular momentum of the rotor during flight. We assume that the thrust of each rotor is oriented in the z-axis of the vehicle, although we note that this assumption does not hold once the rotor begins to rotate and translate through the air, an effect that is known as *rotor flapping* [21].

Each rotor is directly connected to a brushless DC motor, which is controlled by a motor speed controller. The motor's speed controller effectively enables us to control the rotor's angular velocity ω_i. Experimental results have shown the steady-state thrust f_i, which is generated by a rotor i that is not translating horizontally or vertically in free air, to behave as the lumped parameter model

$$f_i = \mathrm{k}_f \omega_i^2, \tag{5.1.5}$$

where k_f is a constant that is a function of the rotor's disk area, radius, geometry, and profile, as well as air density. This parameter can easily be determined from static thrust experiments, which has the advantage that it will also naturally incorporate the effect of aerodynamic drag induced by the flow of air around the chassis.

In a similar experiment, the reaction torque τ_i acting on the chassis that is generated by a rotor i was shown to exhibit the following behaviour

$$\tau_i = \mathrm{k}_\tau \omega_i^2, \tag{5.1.6}$$

where the coefficient k_τ is also a constant that depends on the same parameters as k_f and can be determined by static thrust experiments. The torque τ_i can therefore be approximated as a linear function of thrust f_i,

$$\tau_i = c_i f_i, \tag{5.1.7}$$

where c_i is a constant of magnitude $|c_i| = \mathrm{c}$ for all i and its sign depends on the handedness of the propeller.

Rigid Body Dynamics

From first principles, the equations of motion for a rigid body of mass $N\mathrm{m}$ and mass moment of inertia $\mathbf{J}$ that is subjected to nonconservative forces $\mathbf{f} = (0, 0, \sum_i^N f_i)$ and torques $\mathbf{t} = (0, 0, \sum_i^N \tau_i)$ is given by

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{5.1.8}$$

$$N\mathrm{m}\dot{\mathbf{v}} = {}^{I}_{B}\mathbf{R}_{\mathrm{zyx}}\mathbf{f}, \tag{5.1.9}$$

$${}^{I}_{B}\dot{\mathbf{R}}_{\mathrm{zyx}} = {}^{I}_{B}\mathbf{R}_{\mathrm{zyx}}\omega^{\times}, \tag{5.1.10}$$

$$\mathbf{J}\dot{\omega} = -\omega \times \mathbf{J}\omega + \mathbf{t}. \tag{5.1.11}$$

The vectors $\mathbf{p} = (x, y, z)$ and $\mathbf{v} = (\dot{x}, \dot{y}, \dot{z})$ denote, respectively, the position and velocity of the rigid body with respect to inertial frame I; the vector $\omega = (p, q, r)$ denotes the angular velocities in the body frame, and the superscript $\times$ denotes the skew symmetric matrix of that vector, i.e.

$$\omega^{\times} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}. \tag{5.1.12}$$

Four of the six degrees of freedom (DOF) of the vehicle – altitude (distance to the ground) z, roll γ, pitch β, and yaw α – can be directly controlled by appropriately varying the thrust f_i of each rotor i around the equilibrium. The altitude (or vertical) dynamics is a function of the total control effort produced by N rotors, $\sum_{i=1}^{N} f_i$. The rolling and pitching torques of the vehicle is achieved using differential thrust; more precisely, they are a function of the control efforts acting along the y and x moment arms, $\sum_{i=1}^{N} y_i f_i$ and $-\sum_{i=1}^{N} x_i f_i$, respectively. The yawing torque around the vertical axis is the sum of all reaction torques, $\sum_{i=1}^{N} c_i f_i$. The equations of motion can be summarized as the following:

$$N\mathrm{m}\ddot{x} = (\mathrm{c}_\alpha \mathrm{s}_\beta \mathrm{c}_\gamma + \mathrm{s}_\alpha \mathrm{s}_\gamma) \sum_{i=1}^{N} f_i, \tag{5.1.13}$$

$$N\mathrm{m}\ddot{y} = (\mathrm{s}_\alpha \mathrm{s}_\beta \mathrm{c}_\gamma - \mathrm{c}_\alpha \mathrm{s}_\gamma) \sum_{i=1}^{N} f_i, \tag{5.1.14}$$

$$N\mathrm{m}\ddot{z} = \mathrm{c}_\beta \mathrm{c}_\gamma \sum_{i=1}^{N} f_i - N\mathrm{mg}, \tag{5.1.15}$$

$$I_{\mathrm{x}}\dot{p} = qr(I_{\mathrm{y}} - I_{\mathrm{z}}) + \sum_{i=1}^{N} y_i f_i, \tag{5.1.16}$$

$$I_{\mathrm{y}}\dot{q} = pr(I_{\mathrm{z}} - I_{\mathrm{x}}) - \sum_{i=1}^{N} x_i f_i, \tag{5.1.17}$$

$$I_z\dot{r} = pq(I_x - I_y) + \sum_{i=1}^{N} c_i f_i, \tag{5.1.18}$$

where (I_x, I_y, I_z) each represents the principal mass moment of inertia around the axis denoted by its subscript. Appendix 5.B shows how this physical parameter is computed.

Hover is considered to be the nominal state of operation and the point of equilibrium for the vehicle; it is defined here as the process of maintaining a constant position (x, y, z) in the air and a constant rotation around the vertical axis (or yaw angle α) with respect to the inertial frame I.

At equilibrium, the total thrust of the vehicle perfectly counteracts the acceleration due to gravity g. In the case where there are an equal number of CW and CCW rotors, the thrust required of each rotor is simply mg. In general, however, this may not be the case and one should use some secondary criteria, such as minimizing the maximum thrust produced. In this case, the nominal thrust of each rotor can be solved via least squares, see Appendix 5.C. To simplify the analysis, the vehicle will be assumed to have an equal number of CW and CCW rotors. As such, each rotor i around the nominal operating point produces a control effort $u_i = f_i - \mathrm{mg}$, in units of force. Assuming small angles, the linearized dynamics can be reduced to the following:

$$N\mathrm{m}\ddot{x} = \beta N \mathrm{mg}, \tag{5.1.19}$$
$$N\mathrm{m}\ddot{y} = -\gamma N \mathrm{mg}, \tag{5.1.20}$$
$$N\mathrm{m}\ddot{z} = \sum_{i=1}^{N} u_i, \tag{5.1.21}$$
$$I_x\ddot{\gamma} = \sum_{i=1}^{N} y_i u_i, \tag{5.1.22}$$
$$I_y\ddot{\beta} = -\sum_{i=1}^{N} x_i u_i, \tag{5.1.23}$$
$$I_z\ddot{\alpha} = \sum_{i=1}^{N} c_i u_i, \tag{5.1.24}$$

where the angular accelerations $\dot{\omega} = (\dot{p}, \dot{q}, \dot{r})$ have been converted to second derivative Euler angles $\ddot{\psi} = (\ddot{\gamma}, \ddot{\beta}, \ddot{\alpha})$ through the following kinematic relations [20] such that all equations of motion are represented with respect to the inertial frame:

$$\dot{\psi} = \mathbf{T}(\psi)\omega, \tag{5.1.25}$$
$$\ddot{\psi} = \mathbf{T}(\psi)(\dot{\omega} - \dot{\mathbf{T}}^{-1}(\psi)\dot{\psi}). \tag{5.1.26}$$

where $\mathbf{T}(\psi)$ is a nonlinear transformation of the form

$$\mathbf{T}(\psi) := \begin{bmatrix} 1 & \sin\gamma\tan\beta & \cos\gamma\tan\beta \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma\sec\beta & \cos\gamma\sec\beta \end{bmatrix}. \tag{5.1.27}$$

Since the DFA operates around the equilibrium, the transformation (5.1.27) can be reduced to the following linear transform $L\{\,\cdot\,\}$ assuming small angle approximations:

$$L\{\mathbf{T}(\psi)\} := \begin{bmatrix} 1 & 0 & \beta \\ 0 & 1 & -\gamma \\ 0 & \gamma & 1 \end{bmatrix}. \tag{5.1.28}$$

In the rest of this work we will express the equations of motion (5.1.19) – (5.1.24) more compactly as

$$\ddot{x} = \beta g, \tag{5.1.29}$$
$$\ddot{y} = -\gamma g, \tag{5.1.30}$$

$$\mathbf{M}\begin{bmatrix} \ddot{z} \\ \ddot{\gamma} \\ \ddot{\beta} \\ \ddot{\alpha} \end{bmatrix} = \mathbf{P}^{\mathrm{T}}\mathbf{u}, \tag{5.1.31}$$

where

$$\mathbf{M} = \operatorname{diag}(N\mathrm{m}, I_{\mathrm{x}}, I_{\mathrm{y}}, I_{\mathrm{z}}), \tag{5.1.32}$$
$$\mathbf{u} = (u_1, \dots, u_N), \tag{5.1.33}$$

and the matrix $\mathbf{P} \in \mathbb{R}^{N\times 4}$ contains information pertaining to the configuration of the vehicle,

$$\mathbf{P} = \begin{bmatrix} 1 & y_1 & -x_1 & c_1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & y_N & -x_N & c_N \end{bmatrix}. \tag{5.1.34}$$

5.2 Configuration Parameterization & Scaling

Let ℓ denote the characteristic length of a module and assume that the vehicle's configuration is relatively circular, then $\frac{\ell\sqrt{N}}{2}$ is comparable to the radius of the vehicle. Assuming a disk-like configuration, the vehicle's principal mass moment of inertia around the radial axis approximates that of a full disk,

$$I_d = \frac{N\mathrm{m}}{4}\left(\frac{\ell\sqrt{N}}{2}\right)^2. \tag{5.2.1}$$

In reality, however, the vehicle may take on much sparser and eccentric configurations. To capture this, two scaling parameter, ϵ_{x} and ϵ_{y}, are employed to capture the mass distribution of the vehicle along the x- and y-axis, respectively. The principal mass moment of inertia of the vehicle can thus be summarized as the following:

$$I_{\mathrm{x}} = \epsilon_{\mathrm{x}} I_d, \tag{5.2.2}$$

$$I_{\mathrm{y}} = \epsilon_{\mathrm{y}} I_d, \tag{5.2.3}$$

and for planar objects such as the DFA, the mass moment of inertia around the z-axis follows the Perpendicular Axis Theorem [22]

$$I_{\mathrm{z}} = I_{\mathrm{x}} + I_{\mathrm{y}} = (\epsilon_{\mathrm{x}} + \epsilon_{\mathrm{y}}) I_d. \tag{5.2.4}$$

The mass distribution parameters $(\epsilon_{\mathrm{x}}, \epsilon_{\mathrm{y}})$ can be used to parameterize the configuration space of the vehicle. They are expected to be close to 1 for a disk-like configuration and they are expected to get larger or smaller depending on the eccentricity of the vehicle's configuration. The upper and lower bounds of these parameters are characterized by a straight line configuration along the x- and y-axis.

By substituting (5.2.2) – (5.2.4) into (5.1.21) – (5.1.24), then normalizing the various physical parameters and control efforts as the following:

$$\hat{x}_i = \frac{x_i}{\frac{\ell\sqrt{N}}{2}}, \quad \hat{y}_i = \frac{y_i}{\frac{\ell\sqrt{N}}{2}}, \quad \hat{c}_i = \frac{c_i}{\ell}, \quad \hat{u}_i = \frac{u_i}{\mathrm{m}}, \tag{5.2.5}$$

we can gain some intuition about how the flight dynamics scale with the size of the vehicle N and how they are affected by the vehicle's configuration. Here the normalized position coordinates $\hat{x}_i$ and $\hat{y}_i$ are at most on the order of 1 for a disk-like configuration, $\hat{c}_i$ is the normalized force to torque conversion constant and is expected to be much less than 1, and $\hat{u}_i$ is the normalized control effort in units of acceleration.

Lateral dynamics are unaffected by size and/or configuration, as can be seen from (5.1.29) and (5.1.30). However, the remaining DOF are affected, which can be seen in the following normalized and linearized equations of motion:

$$\ddot{z} = \frac{1}{N}\sum_{i=1}^{N} \hat{u}_i, \tag{5.2.6}$$

$$\hat{I}_{\mathrm{x}}\ddot{\gamma} = \frac{1}{N}\sum_{i=1}^{N} \hat{y}_i\hat{u}_i, \tag{5.2.7}$$

$$\hat{I}_{\mathrm{y}}\ddot{\beta} = -\frac{1}{N}\sum_{i=1}^{N} \hat{x}_i\hat{u}_i, \tag{5.2.8}$$

$$\hat{I}_{\mathrm{z}}\ddot{\alpha} = \frac{1}{N}\sum_{i=1}^{N} \hat{c}_i\hat{u}_i, \tag{5.2.9}$$

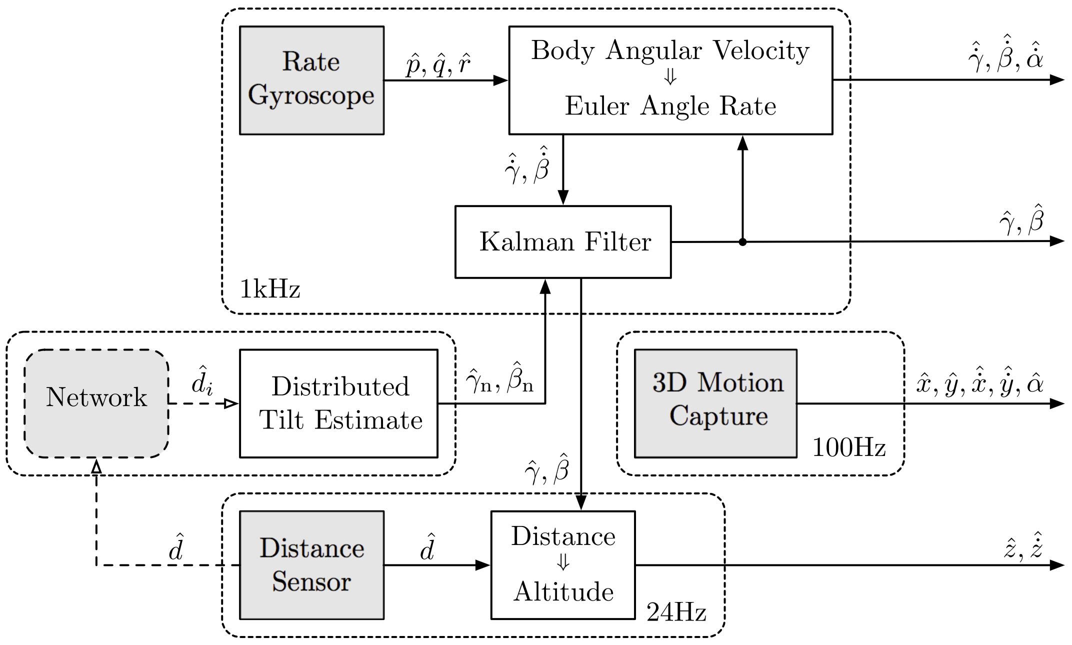
where

$$\hat{I}_{\mathrm{x}} = \frac{\epsilon_{\mathrm{x}}\ell\sqrt{N}}{8}, \quad \hat{I}_{\mathrm{y}} = \frac{\epsilon_{\mathrm{y}}\ell\sqrt{N}}{8}, \quad \hat{I}_{\mathrm{z}} = \frac{(\epsilon_{\mathrm{x}}+\epsilon_{\mathrm{y}})\ell N}{8}. \tag{5.2.10}$$

This shows that the maximum vertical acceleration is independent of N. The maximum angular accelerations in roll and pitch, however, decrease by a factor of $\sqrt{N}$, while the maximum angular acceleration in yaw decreases by a factor of N. Angular accelerations are also affected by the vehicle's configuration; angular accelerations in roll and pitch decrease by a factor of ϵ_{x} and ϵ_{y}, respectively, while yaw decreases by a combination of both mass distribution parameters.

5.3 State Estimation

To control the DFA we require an estimate of the vehicle's attitude, angular velocities, position, and translational velocities. Each module is capable of estimating a subset of these states using its onboard sensors. The remaining states are measured using offboard sensors. Figure 5.3.1 shows a block diagram of the various sensors that are used to obtain a full state estimate of our vehicle.



Rate Gyroscope
$\hat{p}, \hat{q}, \hat{r}$
Body Angular Velocity ⇓ Euler Angle Rate
$\hat{\dot{\gamma}}, \hat{\dot{\beta}}, \hat{\dot{\alpha}}$
$\hat{\gamma}, \hat{\beta}$
Kalman Filter
$\hat{\gamma}, \hat{\beta}$
1kHz
Network
$\hat{d}_i$
Distributed Tilt Estimate
$\hat{\gamma}_\mathrm{n}, \hat{\beta}_\mathrm{n}$
3D Motion Capture
$\hat{x}, \hat{y}, \hat{\dot{x}}, \hat{\dot{y}}, \hat{\alpha}$
100Hz
$\hat{\gamma}, \hat{\beta}$
$\hat{d}$
Distance Sensor
$\hat{d}$
Distance ⇓ Altitude
$\hat{z}, \hat{\dot{z}}$
24Hz

Figure 5.3.1 The onboard sensors, consisting of a 3-axis rate-gyroscope and IR distance measurement sensor, can be used to estimate Euler angle rates $(\hat{\dot{\gamma}}, \hat{\dot{\beta}}, \hat{\dot{\alpha}})$, tilt $(\hat{\gamma}, \hat{\beta})$, altitude $\hat{z}$, and vertical velocity $\hat{\dot{z}}$. Distance measurements can be shared with other modules over the inter-module communication network, which can be used to estimate tilt $(\hat{\gamma}_\mathrm{n}, \hat{\beta}_\mathrm{n})$ [23]. An offboard 3D motion capture system is used to measure XY-position/-velocity $(\hat{x}, \hat{y}, \hat{\dot{x}}, \hat{\dot{y}})$ and yaw-angle $\hat{\alpha}$.

Onboard Sensors

Each module is equipped with a 3-axis rate-gyroscope and an IR distance measurement sensor. The rate-gyroscope measures body angular velocities of the vehicle (p, q, r), which can be converted to Euler angle rates $(\dot{\gamma}, \dot{\beta}, \dot{\alpha})$ through the transformation in (5.1.25). Euler angle rates $(\dot{\gamma}, \dot{\beta}, \dot{\alpha})$ are therefore a function of the Euler angles (γ, β) and the body angular velocities (p, q, r). Euler angles are approximated using the most recent estimate from the previous time step. This approximation works well in practice because the update rate of our Euler angle estimate is relatively fast (i.e. 1 kHz).

Static calibration parameters of the rate-gyroscope such as scale factor, misalignment, and non-orthogonality [24] are assumed to have been calibrated out in an earlier sensor calibration step and that the sensor's sensitivity to linear acceleration is negligible. Upon start-up, while the vehicle remains on the ground and the motors are turned off, each module estimates its sensor bias by performing a running average of its measurements over a few seconds.

By integrating the roll and pitch Euler angle rates, we can obtain a local tilt estimate that has good short-term characteristics. However, a bias in the rate measurement (due to e.g. temperature effects and quantization error) will result in unbounded drift in the tilt estimate, which will be particularly noticeable in the translation dynamics of the vehicle, see (5.1.29) and (5.1.30). To correct for this, a method was developed in [23] to use the IR distance measurement sensors to estimate the vehicle's tilt (i.e. roll γ and pitch β) by exchanging information between neighbouring modules over the vehicle's inter-module communication network. This method assumes that the vehicle flies over a flat horizontal surface. Assuming that the sensors are calibrated, this estimate is non-biased, drift-free, and relatively precise, however the update rate is slow and depends on the algebraic connectivity of the network [25]. A Kalman filter is used to optimally combine both the distributed tilt estimate with the local rate-gyroscope measurements in [26].

In addition to estimating tilt, the IR distance measurement sensors can be used to estimate altitude z, again assuming that the surface over which it flies is flat and horizontal. These sensors point downwards, measuring the distance from the module to a point on the ground. Altitude is a function of this distance d, the sensor's position (r_x, r_y) with respect to the vehicle's body frame B, and the tilt of the vehicle. More precisely, altitude z of the vehicle is the scalar projection of a vector ${}_B\mathbf{k} = (r_x, r_y, -d)$ in the vehicle's body frame, pointing from the center of rotation (or origin of the body frame B) of the vehicle to the point of measurement, onto the unit normal ${}_B\hat{\mathbf{n}}$ of the vehicle's plane (also known as the Hesse Normal form),

$$-{}_B\hat{\mathbf{n}} \cdot {}_B\mathbf{k} = z. \tag{5.3.1}$$

The unit normal ${}_B\hat{\mathbf{n}}$ is obtained by rotating the z-axis unit vector in the inertial frame ${}_I\hat{\mathbf{z}} = (0, 0, 1)$ to the body-frame B via the rotation matrix (5.1.1),

$$\begin{aligned} {}_B\hat{\mathbf{n}} &= {}^I_B\mathbf{R}_{\text{zyx}} \cdot {}_I\hat{\mathbf{z}} && (5.3.2) \\ &= \begin{bmatrix} \sin\beta \\ -\cos\beta\sin\gamma \\ -\cos\beta\cos\gamma \end{bmatrix}. && (5.3.3) \end{aligned}$$

Making the appropriate substitutions into (5.3.1) and solving for altitude z yields

$$z = d\cos\beta\cos\gamma + r_x \sin\beta - r_y \cos\beta\sin\gamma. \tag{5.3.4}$$

It is clear from (5.3.4) that the altitude estimate has a nonlinear dependency on the roll γ and pitch β angles of the vehicle. At equilibrium, however, this can be linearized to the following:

$$L\{z\} = d + r_x\beta - r_y\gamma. \tag{5.3.5}$$

An estimate of the vehicle's altitude $\hat{z}$ can be computed using (5.3.5), using the latest IR distance sensor measurement $\hat{d}$, the known position of the sensors (r_x, r_y), and the latest tilt estimate $(\hat{\gamma}, \hat{\beta})$. The discrete-time derivative of the altitude estimate can be used to estimate the vertical velocity of the vehicle $\hat{\dot{z}}$.

External Sensors

With its existing set of sensors, the DFA is incapable of accurately estimating its lateral position and velocity in flight, nor its yaw-angle. To measure these states, and to simultaneously evaluate our control and estimation algorithms, the DFA is flown in the workspace of a 3D motion capture system (Vicon MX), which is capable of measuring the 3D pose (i.e. position and orientation) of an object with respect to an intertial coordinate frame in real-time. The system functions by illuminating this workspace with IR light[1] and visually tracks uniquely placed IR-reflective markers on the object; the baseline of these markers determine the precision of the system's pose estimate. From these markers the pose of the object, in this case the DFA, is computed by dedicated hardware at a rate of 200 Hz and this information is sent directly over ethernet to a computer (base station), which then forwards this information to each module over a wireless interface at a rate of 100 Hz. In future revisions of the vehicle, however, one can eliminate the need for a 3D motion capture system with additional onboard sensors, such as a global positioning system [27], machine-vision [28, 29], etc.

Wireless transmission of information is inherently prone to transmission losses. To account for this, we use a model-based predictor to estimate the states at any given time and we use measurements only when they become available. More precisely, let $\hat{\mathbf{x}}[k]$ represent the state estimate of a vehicle at time step k consisting of the position, velocity, Euler angles, and Euler angle rates; let the elements of the vector $\mathbf{u}[k]$ consist of the command efforts u_i from all modules. A prediction of the state at time step $k+1$ using an estimate of the state up to time step k is given by

$$\hat{\mathbf{x}}[k+1|k] = \mathbf{f}(\hat{\mathbf{x}}[k|k], \mathbf{u}[k]), \tag{5.3.6}$$

where $\mathbf{f}$ is a function describing the linear dynamics (5.1.29) – (5.1.31). The state is

[1] The 3D motion capture system that is used in our experiments uses an IR wavelength that is different from the IR light emitted a module's IR distance measurement sensor, resulting in little or no interference.

updated whenever a measurement $\tilde{\mathbf{x}}$ becomes available,

$$\hat{\mathbf{x}}[k+1|k+1] = \tilde{\mathbf{x}}[k+1|k+1]. \tag{5.3.7}$$

However, in the event of temporary measurement failure, such as a transmission loss in a wireless link, the estimator simply pushes forward the previous estimate,

$$\hat{\mathbf{x}}[k+1|k+1] = \hat{\mathbf{x}}[k+1|k]. \tag{5.3.8}$$

The use of a predictor enables the system to operate in situations where measurements arrive at unpredictable instances in time and thus increases the robustness of the system.

5.4 Expected Exogenous Disturbances

The expected exogenous disturbances can be divided into two categories: (1) process noise and (2) measurement noise. We experimentally characterized the power spectral density (PSD) of each disturbance for our system and demonstrate that they can all be modeled as zero-mean white noise.

The PSD $S_{\nu\nu}$ for a discrete ergodic signal ν of sample size N, which is assumed to be periodic (i.e. $\nu[i+lN] = \nu[i] \quad \forall l \in \mathbb{Z}$), is the discrete Fourier transform of the signal's autocorrelation function [30]:

$$S_{\nu\nu}(\Omega_k) = \sum_{n=0}^{\mathrm{N}-1} R_{\nu\nu}[n] e^{-j\Omega_k n} \text{ for } k = \{0, 1, \ldots, \mathrm{N}-1\}, \tag{5.4.1}$$

where $\Omega_k = 2\pi k/\mathrm{N}$ is the normalized frequency and $R_{\nu\nu}[n]$ is the autocorrelation function of an ergodic signal ν computed with wrap-around, which has units of the signal squared, and is defined as

$$R_{\nu\nu}[n] = \frac{1}{\mathrm{N}} \sum_{m=0}^{\mathrm{N}-1} \nu[m]\nu[m-n]. \tag{5.4.2}$$

Process Noise

The primary source of process noise comes from the turbulent flow of air that is generated by the rotor and aerodynamic properties of the chassis. We model this type of disturbance as being spatially *uncorrelated* across the vehicle. In an outdoor environment, one could also take into account spatially *correlated* disturbances, such as a wind gusts using a Dryden wind turbelence model [31]. To characterize the spatially uncorrelated disturbance, a single module was rigidly attached to a 6-axis force-torque sensor (ATI Industrial Automation Mini40) with a sampling frequency of 7 kHz to measure the normalized thrust (in units of acceleration) produced at equilibrium; thrust is normalized by the mass m of a single module.

As expected, the PSD (see Figure 5.4.1) of this disturbance is relatively constant across a large frequency band, except for a significant amount of power contained at 0.16 radians per sample (or 1, 144 rad/s), which is approximately twice the angular velocity of our 2-bladed rotor (5,460 RPM). Since the PSD is not completely constant, it may not be appropriate to assume that the process noise due to the rotor is white. However, most of this power lies in a frequency band substantially higher than the expected closed-loop dynamics of the system. It is therefore reasonable to assume a zero-mean white noise spectrum with a standard deviation of $\sigma_p = 1.16\,\mathrm{m/s^2}$ in the frequency range of interest.

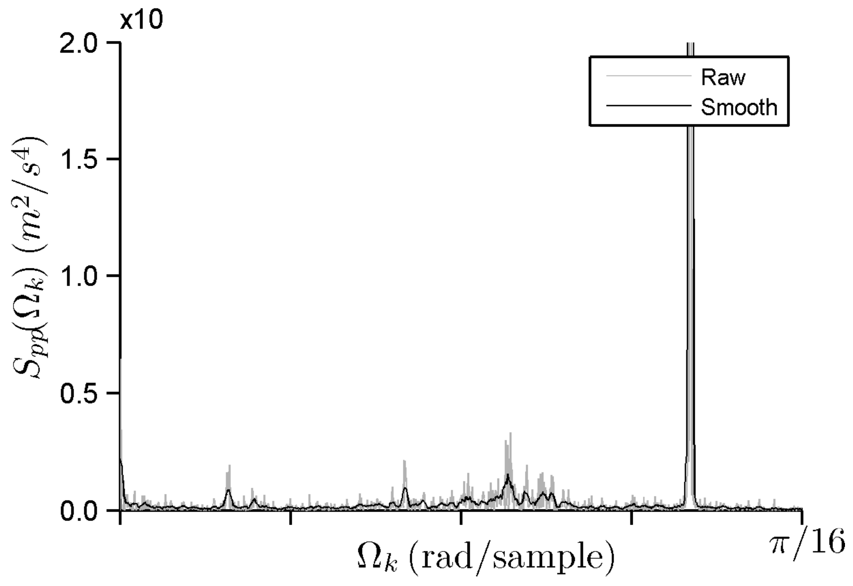


Figure 5.4.1 The PSD of the rotor-generated disturbance at equilibrium thrust shows that the main portion of power is contained at 0.16 rad/sample (or 1, 144 rad/s), which is twice the angular velocity of the module's 2-blade rotor. Measurements in this experiment were sampled at 7 kHz.

This disturbance, which is independently generated by each rotor, manifests itself as process noise on the state of the vehicle. This can be seen by considering (5.1.31); the variances in process noise for those states is given by

$$\begin{bmatrix} \sigma^2_{\ddot{z}} \\ \sigma^2_{\ddot{\gamma}} \\ \sigma^2_{\ddot{\beta}} \\ \sigma^2_{\ddot{\alpha}} \end{bmatrix} = \mathbf{M}^{-1}\mathbf{P}^{\mathrm{T}}\mathrm{Var}[\mathbf{u}]\mathbf{P}(\mathbf{M}^{-1})^{\mathrm{T}}, \tag{5.4.3}$$

where $\mathrm{Var}[\mathbf{u}] = \sigma_p^2$. The standard deviations resulting from this expression are given by the following:

$$\sigma_{\ddot{z}} = \frac{\sigma_p}{\mathrm{m}\sqrt{N}}, \tag{5.4.4}$$

$$\sigma_{\ddot{\gamma}} = \frac{\sigma_p}{\mathrm{m}\sqrt{\sum_{i=1}^{N} y_i^2}}, \tag{5.4.5}$$

$$\sigma_{\ddot{\beta}} = \frac{\sigma_p}{\mathrm{m}\sqrt{\sum_{i=1}^{N} x_i^2}}, \tag{5.4.6}$$

$$\sigma_{\ddot{\alpha}} = \frac{N|c_i|\sigma_p}{\mathrm{m}\sqrt{\sum_{i=1}^{N}(x_i^2 + y_i^2)}}. \tag{5.4.7}$$

As expected, process noise decreases as the size of the vehicle gets larger.

It is worth mentioning that one type of disturbance that is neglected in this model due to modeling difficulties is the transverse airflow caused by neighbouring rotors [32]. However, their effects are assumed to be negligible.

Measurement Noise

The PSD for each sensor (i.e. rate-gyroscope, IR distance measurement sensor, and 3D motion capture system) was computed using data obtained from experiments.

Rate-Gyroscope The PSD of the measurements obtained from the rate-gyroscope is shown in Figure 5.4.2. Measurements were taken at 700 Hz while sitting at rest. The PSD shows a reasonably flat spectrum, with a very gradual decrease above 0.18 radians per sample (or 125.6 rad/s). This can be attributed to the sensor's built-in low-pass filter, which has a cut-off frequency that was manually set to 20 Hz (or 125.6 rad/s) – an order of magnitude larger than the expected closed-loop dynamics of the system.

The variance of the Euler angle rate estimate is approximated by sending $\mathrm{Var}[\hat{\omega}] = (\sigma_p^2, \sigma_q^2, \sigma_r^2)$ through the transformation in (5.1.25) around the equilibrium, which simply results in $\mathrm{Var}[\hat{\dot{\psi}}] = \mathrm{Var}[\hat{\omega}]$. The Euler angle rate estimates are modeled as having a zero-mean white noise spectrum with a standard deviation of $\sigma_{\dot{\gamma}} = \sigma_{\dot{\beta}} = 5.4 \times 10^{-4}$ rad/s around the x- and y-axis, respectively, and $\sigma_{\dot{\alpha}} = 5.7 \times 10^{-4}$ rad/s around the z-axis.

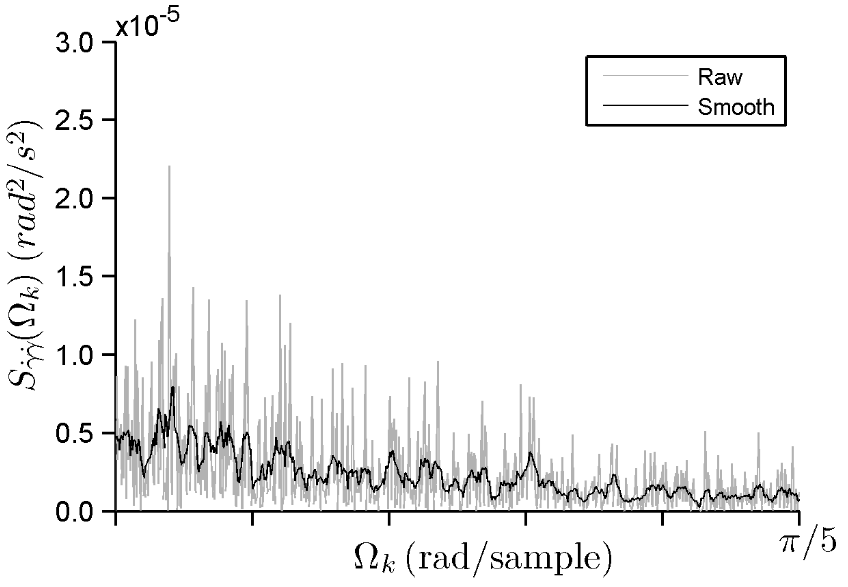


Figure 5.4.2 The PSD of the measurements obtained from the 3-axis rate-gyroscope is relatively flat, with a very gradual decrease above 0.18 rad/sample, which is the cut-off frequency of the sensor's internal low-pass filter. Measurements were taken at 700 Hz. The response shown here is for angular rate measurements in roll $\dot{\gamma}$; angular rate measurements taken for the other two DOF show a similar response.

IR Distance Measurement Sensor The PSD of the measurements obtained from the IR distance measurement sensor is shown in Figure 5.4.3. Static measurements were taken at 1 m distance (i.e. the hovering height of the vehicle) with a sampling frequency of 100 Hz. The PSD shows a relatively flat response, except for the concentration of power contained at approximately $\Omega_k = \pi/2$ radians per sample, which corresponds to the refresh rate of the sensor (24 Hz) and is again larger than the expected closed-loop dynamics of the system. The altitude estimate is therefore modeled as having a zero-mean white noise spectrum with a standard deviation of $\sigma_z = 5.8 \times 10^{-3}$ m.

As mentioned in Section 5.3, in addition to estimating altitude, IR distance measurements are used to estimate tilt (γ, β). Due to its linear relationship (5.3.5), the PSD of these estimates also show a flat response and can therefore be modeled as a zero-mean white noise spectrum. The standard deviations of these measurements $(\sigma_\gamma, \sigma_\beta, \sigma_z)$, however, will depend on the physical configuration of the vehicle [23].

3D Motion Capture System The PSD of the measurements obtained from the 3D motion capture system, with the tracked object at rest, is shown in Figure 5.4.4. Measurements were taken at 200 Hz. The PSD is shown to have a relatively flat response across the entire spectrum, for all DOF. Position, velocity, and yaw angle measurements of the vehicle are therefore modeled as having a zero-mean white noise spectrum with a standard deviation of $\sigma_x = \sigma_y = 3.0 \times 10^{-4}$ m, $\sigma_{\dot{x}} = \sigma_{\dot{y}} = 3.0 \times 10^{-2}$ m/s, and $\sigma_\alpha = 1.6 \times 10^{-3}$ rad, respectively.

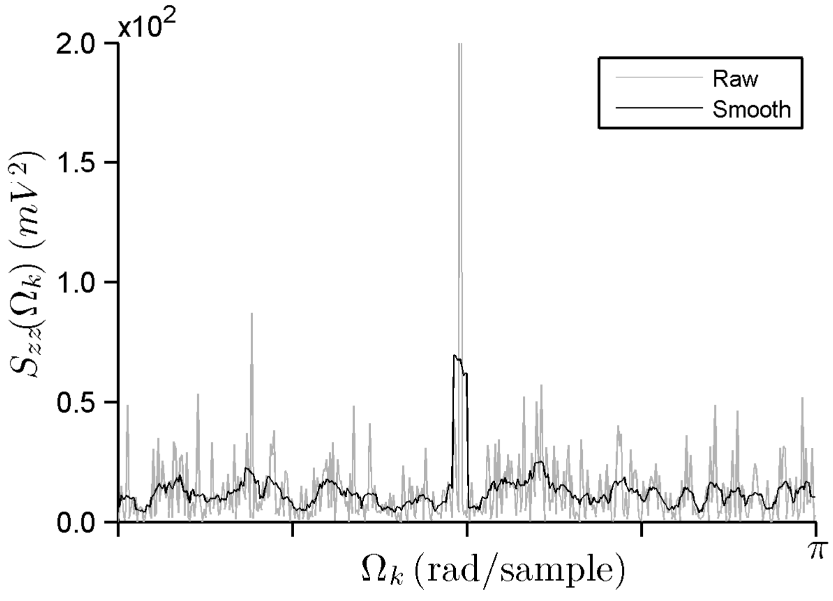


Figure 5.4.3 The PSD of the measurements obtained from the IR distance measurement sensor is relatively flat, except for the peak at $\Omega_k = \pi/2$, which corresponds to the refresh rate of the sensor (24 Hz). Measurements were taken at 100 Hz.

To summarize, all exogenous disturbances expected to act on our system can in practice be modeled as zero-mean white noise. The standard deviations for the estimate or measurement of each state is listed in Table 5.4.1.

5.5 Control Strategy

The system can be thought of as a rigid body of varying shape and size with multiple inter-communicating control units, each of which is capable of independently controlling its own actuator, see Figure 5.5.1. A relatively straightforward, parameterized controller is presented here to control the system around the equilibrium (or hover). Although this approach may seem to limit the vehicle's maneuverability, VTOL vehicles with linear controllers designed around equilibrium have been demonstrated to be quite agile and capable of performing aggressive maneuvers [33]–[36]. Nevertheless, nonlinear controllers that are exponentially stable almost everywhere in SO(3) exist [37] and can in principal be applied here as well.

With full state feedback it is possible to design, for example, a linear quadratic regulator [38] for a given configuration. This method, however, has some drawbacks: (1) the controller itself is essentially a black box to the designer and provides little intuition on the closed-loop dynamics of the system; and (2) the size and computational complexity of the controller increases with the size of the vehicle and is therefore not scalable. Due to these limitations, we have devised a scalable control strategy based on loop-shaping [39] for a

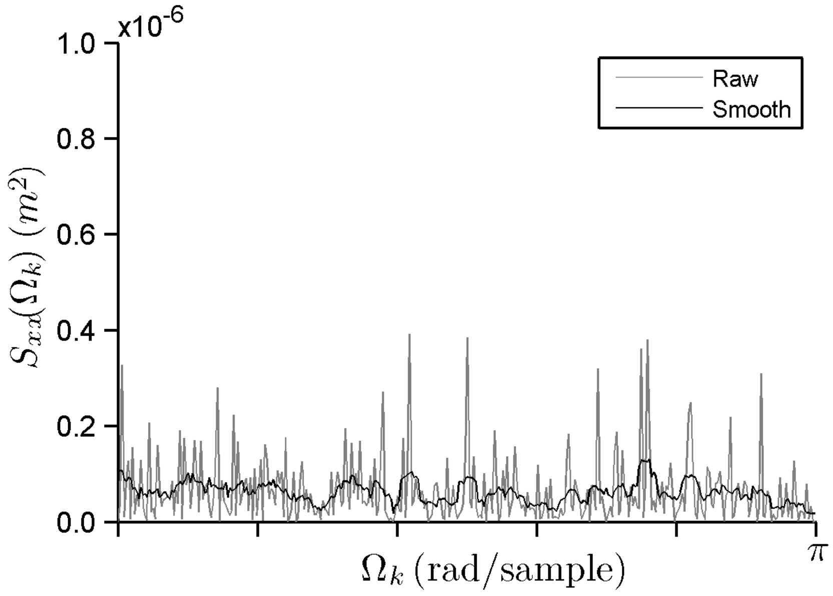


Figure 5.4.4 The PSD of the measurements taken by the 3D motion capture system is relatively flat over the entire spectrum. The response shown here is for the position x; the responses for the other DOF are similiar. Measurements were taken at 200 Hz.

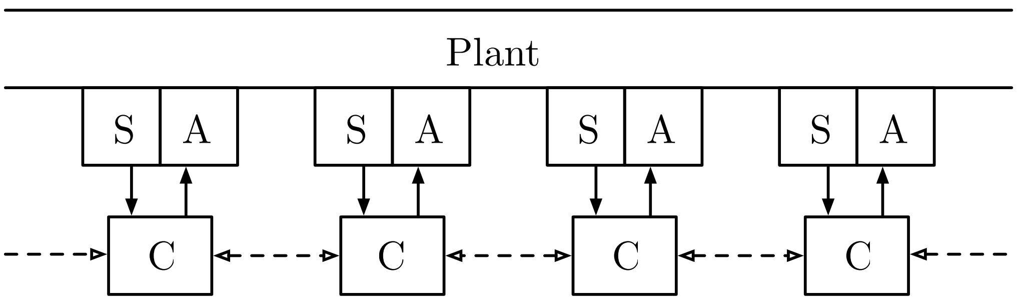


Figure 5.5.1 The system can be modeled as an ad hoc mesh of interconnected controllers; shown here is a one-dimensional representation. Each control unit (or module) C uses its onboard sensors S to independently control its actuator A in order to maintain overall stability of the vehicle. Only adjacent modules may communicate with one another.

modular VTOL vehicle using a finite number of tuning parameters. This was previously presented in [40], the results of which are summarized below.

We employ a strongly cascaded control architecture, see Figure 5.5.2. The lowest level control loop is the motor speed controller. Next is the altitude and attitude controller, which are the four DOF that can be directly controlled. The highest level control loop is the XY-position controller.

Actuator Dynamics

The transfer function that takes the desired control effort $u_{d,i}$ to the actual control effort u_i, around the nominal operating point, can be modeled as a linear time-invariant (LTI) system. We use a non-parametric frequency response identification method to character-

Table 5.4.1 Standard Deviations of all State Estimates

	Standard Deviation
σ_x	$3.0 \times 10^{-4}\,\mathrm{m}$
σ_y	$3.0 \times 10^{-4}\,\mathrm{m}$
σ_z	$5.8 \times 10^{-3}\,\mathrm{m}$
$\sigma_{\dot{x}}$	$3.0 \times 10^{-2}\,\mathrm{m/s}$
$\sigma_{\dot{y}}$	$3.0 \times 10^{-2}\,\mathrm{m/s}$
$\sigma_{\dot{z}}$	$5.8 \times 10^{-1}\,\mathrm{m/s}$
$\sigma_\gamma, \sigma_\beta$	see [23]
σ_α	$1.6 \times 10^{-3}\,\mathrm{rad}$
$\sigma_{\dot{\gamma}}$	$5.4 \times 10^{-4}\,\mathrm{rad/s}$
$\sigma_{\dot{\beta}}$	$5.4 \times 10^{-4}\,\mathrm{rad/s}$
$\sigma_{\dot{\alpha}}$	$5.7 \times 10^{-4}\,\mathrm{rad/s}$
$\sigma_{\ddot{x}}, \sigma_{\ddot{y}}$	$0\,\mathrm{m/s^2}$
$\sigma_{\ddot{z}}, \sigma_{\ddot{\gamma}}, \sigma_{\ddot{\beta}}, \sigma_{\ddot{\alpha}}$	see (5.4.4) – (5.4.7)

ize this system [41]. A sinusoidally varying desired control effort, converted to angular velocity via (5.1.5), was sent to the motor speed controller with an offset equivalent to the equilibrium thrust and an amplitude representative of the operating region. The thrust resulting from the commanded input was measured using a 6-axis force-torque strain gauge sensor with a resolution of 0.01 N at a sampling frequency an order of magnitude higher than the maximum input frequency. This procedure was repeated for inputs with frequencies ranging from 0.1 to 100 rad/s. As shown in Figure 5.5.3, the thrust response can be approximated as a first-order LTI system up to 45 rad/s,

$$\frac{u_i(s)}{u_{d,i}(s)} = \frac{\omega_r}{s + \omega_r}, \tag{5.5.1}$$

where the rotor's bandwidth ω_r is approximately 30 rad/s.

Altitude and Attitude Controller

Consider a control strategy of the form:

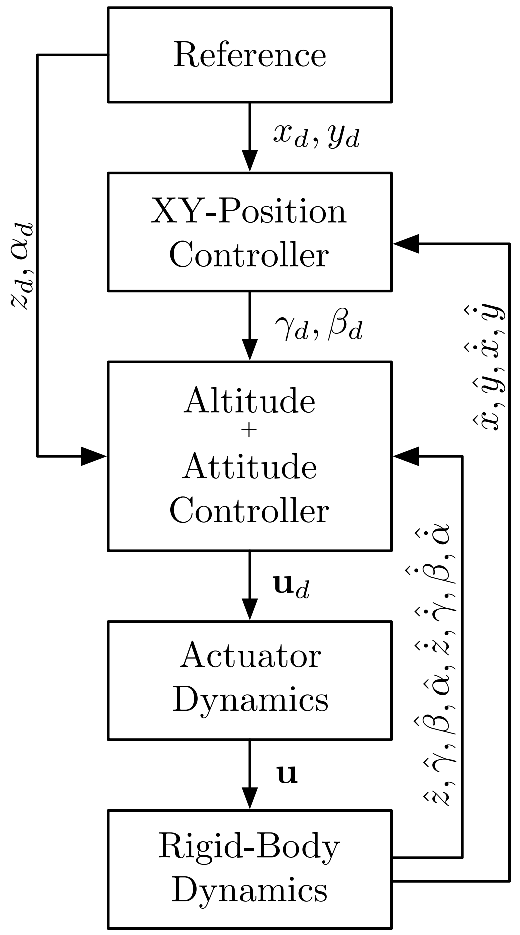

Reference
x_d, y_d
XY-Position Controller
γ_d, β_d
z_d, α_d
Altitude + Attitude Controller
$\mathbf{u}_d$
Actuator Dynamics
$\mathbf{u}$
Rigid-Body Dynamics
$\hat{z}, \hat{\gamma}, \hat{\beta}, \hat{\alpha}, \hat{\dot{z}}, \hat{\dot{\gamma}}, \hat{\dot{\beta}}, \hat{\dot{\alpha}}$
$\hat{x}, \hat{y}, \hat{\dot{x}}, \hat{\dot{y}}$

Figure 5.5.2 A strongly cascaded control architecture is employed, consisting of three layers: (1) low-level actuator dynamics; (2) mid-level altitude and attitude control; and (3) high-level XY-position control.

$$\mathbf{u}_d = \mathbf{Q} \begin{bmatrix} f_z \\ f_\gamma \\ f_\beta \\ f_\alpha \end{bmatrix}, \tag{5.5.2}$$

where the elements of the vector $\mathbf{u}_d \in \mathbb{R}^{N\times 1}$ are the desired control efforts imparted by each module; $\mathbf{Q} \in \mathbb{R}^{N\times 4}$ is a function of the vehicle's configuration and can be designed such that $\mathbf{P}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}_{4\times 4}$. Recall that $\mathbf{P}$ captures the vehicle's configuration (5.1.34). In doing so, the DOF of the closed-loop system become decoupled. The functions $(f_z, f_\gamma, f_\beta, f_\alpha)$ essentially describe the closed-loop dynamics of the system. Note that this is only possible if $\mathbf{P}$ is full column rank, which physically means that the configuration is flight-feasible.

Assuming that an estimator is used to obtain full state estimate of the system, one

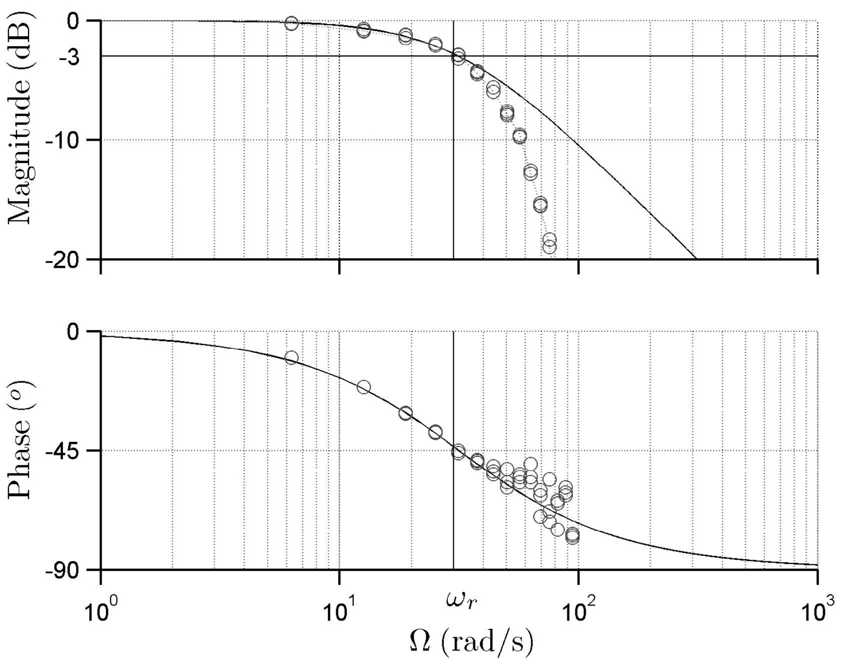


Figure 5.5.3 The frequency response of the motor for commanded input control effort $u_{d,i}$ to actual output control effort u_i (see (5.5.1)) approximates a first-order transfer function up to approximately 45 rad/s. The system's bandwidth (or -3 dB frequency) is approximately 30 rad/s. At higher frequencies, the measured response diverges from the approximation due to higher order effects such as time delay.

can force the closed-loop response of each DOF to behave as a second-order mass-spring-damper.

$$f_z = -N\mathrm{m}(2\omega_z\zeta_z\hat{\dot{z}} + \omega_z^2(\hat{z} - z_d)), \tag{5.5.3}$$

$$f_\gamma = -I_\mathrm{x}(2\omega_\gamma\zeta_\gamma\hat{\dot{\gamma}} + \omega_\gamma^2(\hat{\gamma} - \gamma_d)), \tag{5.5.4}$$

$$f_\beta = -I_\mathrm{y}(2\omega_\beta\zeta_\beta\hat{\dot{\beta}} + \omega_\beta^2(\hat{\beta} - \beta_d)), \tag{5.5.5}$$

$$f_\alpha = -I_\mathrm{z}(2\omega_\alpha\zeta_\alpha\hat{\dot{\alpha}} + \omega_\alpha^2(\hat{\alpha} - \alpha_d)). \tag{5.5.6}$$

The hat notation is used to denote the estimate of that state and the subscript d denotes the desired state. This strategy contains two design parameters per DOF: (1) a closed-loop natural frequency $\omega_s > 0$, and (2) a damping ratio $\zeta_s > 0$, where the subscript $s \in \{z, \gamma, \beta, \alpha\}$ denotes the DOF. One of the features of this control strategy is that it minimizes the shear forces and moments between modules [19].

XY-**Position Controller**

As previously mentioned in Section 5.1, the lateral DOF (x, y) are indirectly controlled through the roll γ and pitch β angles of the vehicle, see (5.1.29) and (5.1.30). Since these two DOF are also decoupled, they can be designed to exhibit a second-order response. Let the functions f_x and f_y represent the closed-loop accelerations in x and y, respectively,

$$f_x = -2\omega_x\zeta_x\hat{\dot{x}} - \omega_x^2(\hat{x} - x_d), \tag{5.5.7}$$

$$f_y = -2\omega_y\zeta_y\hat{\dot{y}} - \omega_y^2(\hat{y} - y_d). \tag{5.5.8}$$

Making use of the tilt to lateral acceleration relationships in (5.1.29) and (5.1.30), this can be converted to the two desired angles in roll γ_d and in pitch β_d of (5.5.4) and (5.5.5),

$$\gamma_d = -\frac{f_y}{g} = \frac{2\omega_y\zeta_y\hat{\dot{y}} + \omega_y^2(\hat{y} - y_d)}{g}, \tag{5.5.9}$$

$$\beta_d = \frac{f_x}{g} = -\frac{2\omega_x\zeta_x\hat{\dot{x}} + \omega_x^2(\hat{x} - x_d)}{g}. \tag{5.5.10}$$

Implementation

Assuming full state feedback of the vehicle, which we denote as $\tilde{\mathbf{y}}$, our controller can be written as

$$\mathbf{u}_d = -\mathbf{QMK}_o\tilde{\mathbf{y}}, \tag{5.5.11}$$

where the vector $\tilde{\mathbf{y}}$ consists of position, Euler angles, translational velocities, and Euler angle rates,

$$\tilde{\mathbf{y}} = (x, y, z, \gamma, \beta, \alpha, \dot{x}, \dot{y}, \dot{z}, \dot{\gamma}, \dot{\beta}, \dot{\alpha}). \tag{5.5.12}$$

This is scaled by the control gain matrix, which contains 12 unique physical parameters,

$$\mathbf{K}_o = \begin{bmatrix} 0 & 0 & \omega_z^2 & 0 & 0 & 0 & 0 & 0 & 2\omega_z\zeta_z & 0 & 0 & 0 \\ 0 & -\omega_\gamma^2\omega_y^2 & 0 & \omega_\gamma^2 & 0 & 0 & 0 & -\omega_\gamma^2 2\omega_y\zeta_y & 0 & 2\omega_\gamma\zeta_\gamma & 0 & 0 \\ \omega_\beta^2\omega_x^2 & 0 & 0 & 0 & \omega_\beta^2 & 0 & \omega_\beta^2 2\omega_x\zeta_x & 0 & 0 & 0 & 2\omega_\beta\zeta_\beta & 0 \\ 0 & 0 & 0 & 0 & 0 & \omega_\alpha^2 & 0 & 0 & 0 & 0 & 0 & 2\omega_\alpha\zeta_\alpha \end{bmatrix}. \tag{5.5.13}$$

The result is scaled by the vehicle's total mass and principal mass moment of inertia, contained in the matrix $\mathbf{M}$, which results in the closed-loop dynamics of the vehicle. The matrix $\mathbf{Q}$ maps these dynamics to the desired control effort of each module, see (5.5.2).

In the case of the DFA, each module i has a full state estimate of the vehicle denoted by

$\mathbf{y}_i$. It is therefore possible to implement our control strategy in a *decentralized* manner, i.e. modules do not need to communicate during flight. This provides us with a lower benchmark in terms of performance for comparing distributed algorithms in future work.

Thus each module i commands its motor with a desired control effort of the form

$$u_{d,i} = -\mathbf{K}_i \mathbf{y}_i, \tag{5.5.14}$$

where

$$\mathbf{K}_i = \mathbf{q}_i \mathbf{M} \mathbf{K}_o, \tag{5.5.15}$$

where $\mathbf{q}_i$ denotes row i of matrix $\mathbf{Q}$. The vector of desired control efforts is thus given by

$$\mathbf{u}_d = \mathbf{K}\mathbf{y}, \tag{5.5.16}$$

where

$$\mathbf{K} = [\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_N]^{\mathrm{T}}, \tag{5.5.17}$$
$$\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]^{\mathrm{T}}. \tag{5.5.18}$$

Assuming that the state estimates of each module is identical, the resulting control effort would be equivalent to (5.5.11) and there will be no shear force or moment acting along the interface of adjacent modules caused by the control effort. However, since state estimates will naturally differ between modules due to uncorrelated sensor noise, there will be an apparent shear force and moment acting along the interface of adjacent modules caused by the control effort alone.

The controller described here has been designed based on continuous-time system dynamics, with no consideration to the sampling rate of the system. Performance losses due to this approximation are considered to be negligible since the sampling rate of all the sensors is significantly faster than the closed-loop system dynamics. The controller itself operates at a rate of 100 Hz, limited only by the maximum command rate of the actuator.

The following three sections deal specifically with the control parameters and can be skipped to Section 5.9 without any loss of continuity. More concretely, we will describe in Section 5.6 how one can compute the optimal control tuning parameters and examine how these optimal parameters are affected by the size and configuration of the vehicle in Section 5.7. In Section 5.8, we demonstrate how these parameters can be efficiently stored on a system with limited computational resource such as the DFA.

5.6 Optimizing Performance

The control tuning parameters in (5.5.13) can be optimized for flight performance while satisfying physical constraints of the system. What follows is a description of a general procedure with implementation details pertaining to the DFA.

Performance Objective

The objective is to find control tuning parameters that minimize our closed-loop system's output (or error) $\mathbf{z}$ power due to an exogenous disturbance $\mathbf{w}$. This can be quantified by computing the $\mathcal{H}_2$ norm $||\cdot||_2$ of our closed-loop system, which we denote as $\mathbf{G}(s)$. In continuous-time, the $\mathcal{H}_2$ norm is defined as

$$||\mathbf{G}(s)||_2^2 = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \mathrm{tr}(\mathbf{G}^*(j\omega)\mathbf{G}(j\omega))d\omega, \tag{5.6.1}$$

where the term $\mathrm{tr}(\mathbf{G}^*(j\omega)\mathbf{G}(j\omega))$ is the sum of the squared magnitudes of all the elements of $\mathbf{G}(j\omega)$ [38]. The $\mathcal{H}_2$ norm can be interpreted as an average gain of the system, performed over all the elements of the matrix transfer function $\mathbf{G}(s)$ over all frequencies, or roughly speaking it is the root-mean square value of the measured response due to white noise input [39]. High performance in this context means minimizing the $\mathcal{H}_2$ norm of our system,

$$\arg\min_{\mathbf{K}_o} ||\mathbf{G}(s)||_2 \quad \text{s.t. physical constraints.} \tag{5.6.2}$$

Searching for the optimal control tuning parameters $\mathbf{K}_o$ means finding the minimum of this constrained multivariable function, which can be implemented with standard software tools like MATLAB's Optimization Toolbox or `CVX` [42].

Our linear time invariant (LTI) closed-loop feedback system $\mathbf{G}(s)$ takes the form shown in Figure 5.6.1, where $\mathbf{w}$ denotes a vector of exogenous input disturbances with a fixed or bounded power spectrum; $\mathbf{z}$ represents the weighted system output (or error), which we want to minimize; the signals $\mathbf{u}_d$ denotes the desired control effort; $\mathbf{y}$ denotes the measured states, see (5.5.18); and $\mathbf{K}$ is given by (5.5.17). The state-space description of the generalized plant is given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_1\mathbf{w} + \mathbf{B}_2\mathbf{u}_d, \tag{5.6.3}$$

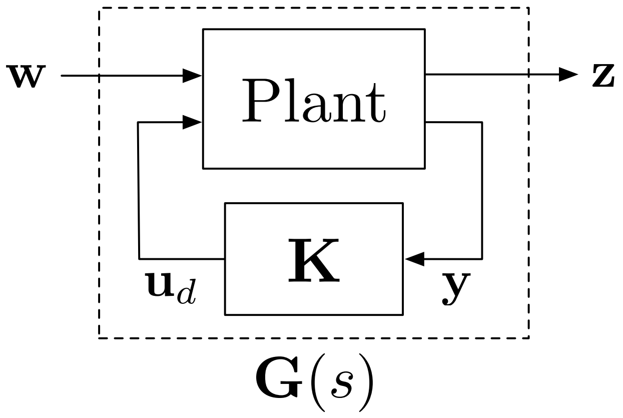


Figure 5.6.1 General block diagram for computing the $\mathcal{H}_2$ norm of a system, where **w** denotes the exogenous inputs to the system and **z** respresents the weighted response resulting from these inputs; the signal $\mathbf{u}_d$ and **y** denote the desired control effort and measured states, respectively.

$$\mathbf{z} = \mathbf{C}_1\mathbf{x}, \tag{5.6.4}$$

$$\mathbf{y}_i = \mathbf{C}_2\mathbf{x} + \mathbf{D}_i\mathbf{w}, \tag{5.6.5}$$

where all matrices are real matrices of compatible dimensions and the state consists of position, Euler angles, translational velocities, Euler angle rates, and the control effort produced by each rotor,

$$\mathbf{x} = (x, y, z, \gamma, \beta, \alpha, \dot{x}, \dot{y}, \dot{z}, \dot{\gamma}, \dot{\beta}, \dot{\alpha}, u_1, \ldots, u_N). \tag{5.6.6}$$

The state transition matrix **A** for nominal flight dynamics of the DFA, including motor dynamics as described in Section 5.5, is given by

$$\mathbf{A} = \begin{bmatrix} 0_{6\times 6} & \mathbf{I}_{6\times 6} & 0_{6\times N} \\ \mathcal{A}_{2\times 6} & 0_{2\times 6} & 0_{2\times N} \\ 0_{4\times 6} & 0_{4\times 6} & \mathbf{M}^{-1}\mathbf{P}^{\mathrm{T}} \\ 0_{N\times 6} & 0_{N\times 6} & -\omega_r \mathbf{I}_{N\times N} \end{bmatrix}, \tag{5.6.7}$$

where

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & \mathrm{g} & 0 \\ 0 & 0 & 0 & -\mathrm{g} & 0 & 0 \end{bmatrix}. \tag{5.6.8}$$

The desired control effort $\mathbf{u}_d$ is fed into the matrix $\mathbf{B}_2$, which scales these values by the rotor's bandwidth ω_r,

$$\mathbf{B}_2 = \begin{bmatrix} 0_{12\times N} \\ \omega_r \mathbf{I}_{N\times N} \end{bmatrix}. \tag{5.6.9}$$

The feedback vector $\mathbf{y}_i$ of module i is a function of the system's states, which does not include control effort, therefore

$$\mathbf{C}_2 = \begin{bmatrix} \mathbf{I}_{12\times 12} & 0_{12\times N} \end{bmatrix}. \tag{5.6.10}$$

The expected exogenous disturbances $\mathbf{w}$, which are described in Section 5.4, consist of unit variance process noise $\mathbf{w}_p$ and measurement noise $\mathbf{w}_m$,

$$\mathbf{w} = (\mathbf{w}_p, \mathbf{w}_m). \tag{5.6.11}$$

Process noise $\mathbf{w}_p$ may consist of correlated and uncorrelated disturbances. Here we only consider the effect that rotor disturbances have on the states of our system, since correlated disturbances such as wind gusts are not a factor when flying indoors in a laboratory environment. This is given by

$$\mathbf{w}_p = (w_{\ddot{x}}, w_{\ddot{y}}, w_{\ddot{z}}, w_{\ddot{\gamma}}, w_{\ddot{\beta}}, w_{\ddot{\alpha}}), \tag{5.6.12}$$

and is scaled by the matrix $\mathbf{B}_1$ using the appropriate standard deviations listed in Table 5.4.1,

$$\mathbf{B}_1 = \left[\begin{array}{c|c} 0_{6\times 6} & \\ \mathrm{diag}(\sigma_{\ddot{x}}, \sigma_{\ddot{y}}, \sigma_{\ddot{z}}, \sigma_{\ddot{\gamma}}, \sigma_{\ddot{\beta}}, \sigma_{\ddot{\alpha}}) & 0_{(12+N)\times 12N} \\ 0_{N\times 6} & \end{array}\right]. \tag{5.6.13}$$

Measurement noise $\mathbf{w}_m$ consists of an individual sensor noise vector for each module i,

$$\mathbf{w}_m = (\mathbf{w}_{m,1}, \ldots, \mathbf{w}_{m,N}), \tag{5.6.14}$$

where

$$\begin{aligned}\mathbf{w}_{m,i} = (&w_{x,i}, w_{y,i}, w_{z,i}, w_{\gamma,i}, w_{\beta,i}, w_{\alpha,i}, \ldots \\ &w_{\dot{x},i}, w_{\dot{y},i}, w_{\dot{z},i}, w_{\dot{\gamma},i}, w_{\dot{\beta},i}, w_{\dot{\alpha},i}),\end{aligned} \tag{5.6.15}$$

and is scaled by the matrix $\mathbf{D}_i$ using the appropriate standard deviation listed in Table 5.4.1,

$$\mathbf{D}_i = \begin{bmatrix} 0_{12\times 6} & \mathcal{D}_1 & \ldots & \mathcal{D}_N \end{bmatrix}, \tag{5.6.16}$$

where,

$$\mathcal{D}_j = \begin{cases} 0_{12\times 12} & \text{if } j \neq i \\ \text{diag}(\sigma_x, \sigma_y, \sigma_z, \sigma_\gamma, \sigma_\beta, \sigma_\alpha, \ldots & \\ \quad \sigma_{\dot{x}}, \sigma_{\dot{y}}, \sigma_{\dot{z}}, \sigma_{\dot{\gamma}}, \sigma_{\dot{\beta}}, \sigma_{\dot{\alpha}}) & \text{otherwise.} \end{cases}$$

The output response $\mathbf{z}$, which is to be minimized, is a function of the states weighted by

$$\mathbf{C}_1 = \left[\begin{array}{cccccc|c} \mathrm{w_x} & 0 & 0 & 0 & 0 & 0 & \\ 0 & \mathrm{w_y} & 0 & 0 & 0 & 0 & \\ 0 & 0 & \mathrm{w_z} & 0 & 0 & 0 & 0_{4\times(6+N)} \\ 0 & 0 & 0 & 0 & 0 & \mathrm{w}_\alpha & \end{array}\right]. \tag{5.6.17}$$

Since the objective of the vehicle is to hover, only position and yaw-angle is considered. It is sensible to set the position weights $(\mathrm{w_x}, \mathrm{w_y}, \mathrm{w_z})$ to the same value; this value is normalized to a constant value of 1. There exists then just one free design parameter: the weight w_α on yaw-angle in units of meters per radian. It is difficult to find a suitable expression for this weight in terms of physical parameters. Experiments have shown $\frac{1}{35}$ m/rad demonstrates reasonable performance, with the deviation in yaw-angle α reasonable in comparison to the position error.

Frequency dependent weights, for both the exogenous input and regulated output, could also be used. For example, had our exogenous disturbances not been zero-mean white noise, a filter could have been designed to exhibit the same characteristic response given a zero-mean white noise input. This would then be absorbed by the generalized plant. One could also consider penalizing high frequency actuation to reduce wear and tear of the actuator or because it may not make sense to ask for performance past the point where the model integrity begins to degrade.

Physical Constraints

Constraints are added to ensure that the parameters obtained from the optimization work within the physical limits of the system. They are also used to reduce the search space of the optimization routine; for example, the parameter search space is reduced to $\omega_s > 0$ and $\zeta_s > 0$.

A physical limit that we want to avoid is saturation of the control effort. The maximum available control effort can be found through experiment; this is approximately 3.5 N of thrust for our system. We also want to ensure that the shear forces and moments resulting from differences in control effort do not disconnect modules from the vehicle. Modules are held together by a set of magnets. Experiments with two modules demonstrate that the connection strength is weakest around an axis running horizontally along the edge of an interface. Assuming one module is fixed, the connection strength of the magnets f_m should counteract the moment that is created by the combination of a module's control effort f_i and gravitational acceleration in order to remain connected. Figure 5.6.2 shows a free body diagram of a single module connected on one side to a fixed module. Although modules are not necessarily fixed, this provides a conservative estimate of the control effort needed to disconnect a module from the vehicle. For simplicity, assume that the centre of mass is at the geometric centre of a module. The following condition must be satisfied in order for a module to remain connected:

$$f_m \cdot (2\mathrm{d}_1 + \mathrm{d}_2) > \left| (f_i - \mathrm{mg}) \frac{\ell}{2} \right|, \tag{5.6.18}$$

where d_1 is the vertical distance from magnet to the top/base of the module and d_2 is the relative distance between both magnets. This expression can be re-arranged to obtain the control effort $u_i = f_i - \mathrm{mg}$ effort that is necessary to decouple two connected modules.

Depending on the strength of the magnets, the magnetic constraint may be more restrictive than control effort saturation. Let $\mathrm{U}_{\max}$ denote the maximum control effort that satisfies both constraints.

In the case of the DFA, it is sufficient if the control effort lies below $\mathrm{U}_{\max}$ *most of the time*. This can be formulated as N inequality constraints, where some multiple of the output control effort's standard deviation σ_{u_i} must lie below $\mathrm{U}_{\max}$. While this does not guarantee that all control signals will remain below $\mathrm{U}_{\max}$, it does quantify the probability of saturation.

Once again, the $\mathcal{H}_2$ system norm can be employed and this time used to obtain the standard deviation σ_{u_i} of the output control effort u_i. It can be shown that the $\mathcal{H}_2$ norm of a single-output system is equivalent to the root-mean-squared (RMS) response of that system due to white noise input. Since this is taken at equilibrium, the RMS response is the standard deviation of the control effort.

Let us denote the system used to compute the control effort's standard deviation of module i as $\mathbf{G}_{c,i}(s)$. This system is identical to $\mathbf{G}(s)$ (see (5.6.3) – (5.6.5)), except that

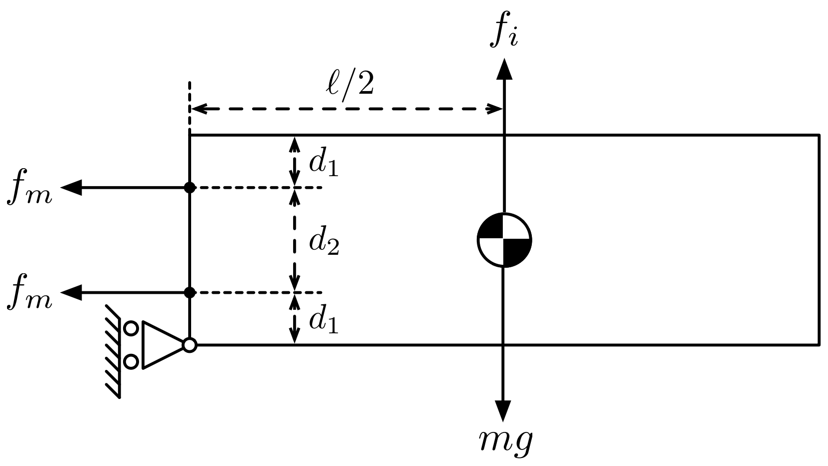


Figure 5.6.2 Consider a conservative scenario where one module is attached to one other module, which is fixed (not shown). The weakest link is subject to a moment created around the edge-axis of a module (pointing out of the page). The connection strength of the magnets f_m should counteract the moment created by the combination of module's control effort f_i and gravitational acceleration g.

the weighted output response is the control effort of module i

$$\mathbf{z}_{c,i} = \mathbf{C}_{1_{c,i}}\mathbf{x}, \tag{5.6.19}$$

where

$$\mathbf{C}_{1_{c,i}} = \left[\begin{array}{c|c} 0_{1\times 12} & \mathcal{C}_{1\times N} \end{array}\right], \tag{5.6.20}$$

where the elements of the vector $\mathcal{C}$ are

$$c_j = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise.} \end{cases} \tag{5.6.21}$$

The standard deviation of the control effort produced by module i is therefore

$$\sigma_{u_i} = ||\mathbf{G}_{c,i}(s)||_2. \tag{5.6.22}$$

Our analysis until now has been performed in continuous-time; our system, however, is implemented in discrete-time. Therefore σ_{u_i} will need to be converted to its discrete-time equivalent in order to properly compare it with the constraint $\mathrm{U}_{\max}$ on the control effort. A relationship between the discrete and continous time $\mathcal{H}_2$ norms is given by [43]

$$||\mathbf{S}_\mathrm{c}||_2 = \frac{1}{\sqrt{\mathrm{T}_s}}||\mathbf{S}_\mathrm{d}||_2, \tag{5.6.23}$$

where $\mathbf{S}_\mathrm{c}$ is a continous-time system and $\mathbf{S}_\mathrm{d}$ is the discretization of this system using zero-order hold with a sampling frequency of T_s. Therefore, the physical constraint of the system is given by

$$||\mathbf{G}_{c,i}(s)||_2 < \frac{\mathrm{U}_\mathrm{max}}{\sqrt{\mathrm{T}_s}} \quad \text{for } i \in \{1, \ldots, N\}. \tag{5.6.24}$$

5.7 Control Parameter Analysis

Using the optimization procedure described the previous section, Section 5.6, the tuning parameters for our parameterized controller were computed using MATLAB's optimization toolbox for various configurations ranging from 4 to 20 modules in size. To provide some statistics, the procedure was implemented on 100 randomly chosen configurations for each vehicle size.

Performance versus Size

We compared the performance of the DFA using our parameterized controller with optimized parameters, implemented in a decentralized architecture, to that of two other controllers in simulation:

1. **Parameterized Controller in a Centralized Architecture**: This is the parameterized controller implemented in a centralized architecture; a controller aggregates and averages sensor measurements from all of the modules, then commands each module with the appropriate control effort. Although not scalable, this is considered the best case scenario when using our paramterized controller.
2. $\mathcal{H}_2$ **Optimal**: This is the $\mathcal{H}_2$ optimal controller [44], which takes into account physical constraints, implemented for our system. It assumes full state feedback of the vehicle and a centralized control architecture, see Appendix 5.D for implementation details. This is considered here to be the best performing controller for our system.

The $\mathcal{H}_2$ optimal controller is expected to perform better than our parameterized control strategy because it is a dynamic controller that also grows with the size of the vehicle N. In contrast, our parameterized controller is a static gain matrix (5.5.13) containing just

12 parameters (two for each DOF) for all vehicle sizes. Traditional control techniques such as $\mathcal{H}_2$ optimal control are impractical for handling the incredible size and computational complexity of such systems. In this case, the system matrices describing the input-output state behaviour of N interconnected modules will be approximately $N \times N$, and most matrix arithmetic operations will require $\mathcal{O}(N^3)$ floating point operations [45]. This makes traditional robust and optimal controller design prohibitively expensive (slow) for a large number of modules, never mind the difficulties of implementing such a controller on a resource-constrained embedded system such as the DFA. We demonstrate here that there is a trade-off between performance and scalability. In addition, thanks to its structure, our controller provides physical intuition about the closed-loop system. The same cannot be said about the $\mathcal{H}_2$ optimal controller, which is essentially a black box to the designer.

The results in Figure 5.7.1 show that flight performance improves (i.e. decreasing $||\mathbf{G}(s)||_2$) as the size of the vehicle increases. This is to be expected for a system with no correlated process noise such as wind gusts and we demonstrated previously in (5.4.4) – (5.4.7) that the effect of spatially uncorrelated process noise decreases as the system grows. With decreasing exogenous disturbances acting on our system as the size of the vehicle gets larger, we would expect to see a decrease in the system's output response.

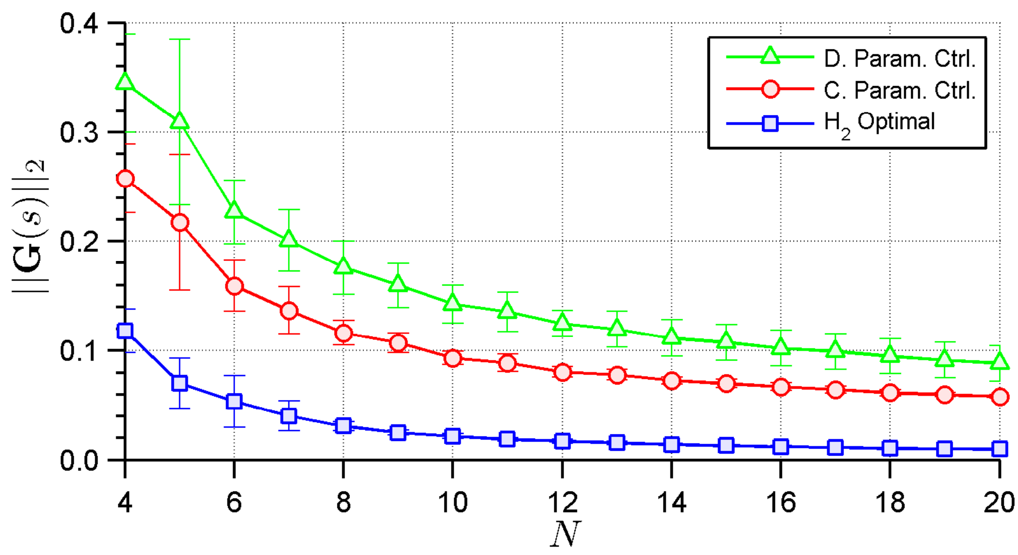


Figure 5.7.1 The performance of the DFA improves (i.e. decreasing $||\mathbf{G}(s)||_2$) as the size of the vehicle N increases. This is to be expected because of the effective decrease in uncorrelated process noise acting on our system as the size of the vehicle gets larger. The $\mathcal{H}_2$ optimal controller is shown to perform better in comparison to the parameterized controller, for both the centralized and decentralized architecture.

The error bars in the figure denote the variability in performance for similar sized vehicles, which suggests that performance is also a function of vehicle configuration. We showed previously in (5.2.6) – (5.2.9) that the vehicle's ability to accelerate is indeed a function of the vehicle's configuration,

$$\ddot{z} \propto 1, \quad \ddot{\gamma} \propto \frac{1}{\epsilon_{\mathrm{x}}}, \quad \ddot{\beta} \propto \frac{1}{\epsilon_{\mathrm{y}}}, \quad \ddot{\alpha} \propto \frac{1}{\epsilon_{\mathrm{x}} + \epsilon_{\mathrm{y}}}. \tag{5.7.1}$$

The result of this and the control effort constraint imposed on our optimization procedure has the effect of varying the performance for similar sized vehicles.

In the limit, as N goes to infinity, the effect that uncorrelated process noise has on the states of our system goes to zero and the resulting performance is equivalent to a closed-loop system with measurement noise acting as the only exogenous disturbance.

Control Parameters versus Size and Configuration

In Figure 5.7.2 we see that the optimal control tuning parameters are a function of the vehicle's size and configuration. This is to be expected since our minimization procedure is a function of the expected exogenous disturbances, which are in turn functions of vehicle's size and configuration.

As previously discussed, increasing vehicle size N has the effect of decreasing the expected exogenous disturbance, namely uncorrelated process noise, see (5.4.4) – (5.4.7). This has two immediate effects on our closed-loop system: (1) the $\mathcal{H}_2$-norm of the performance objective decreases, which is clearly illustrated in Figure 5.7.1; and (2) the expected variance in control effort decreases. These two effects coupled with the physical constraints has the effect of varying optimal control parameters with respect to vehicle size N, as well as configuration for similar sized vehicles. In general, with a reduction in disturbances, one can expect the optimization to choose a more aggressive controller.

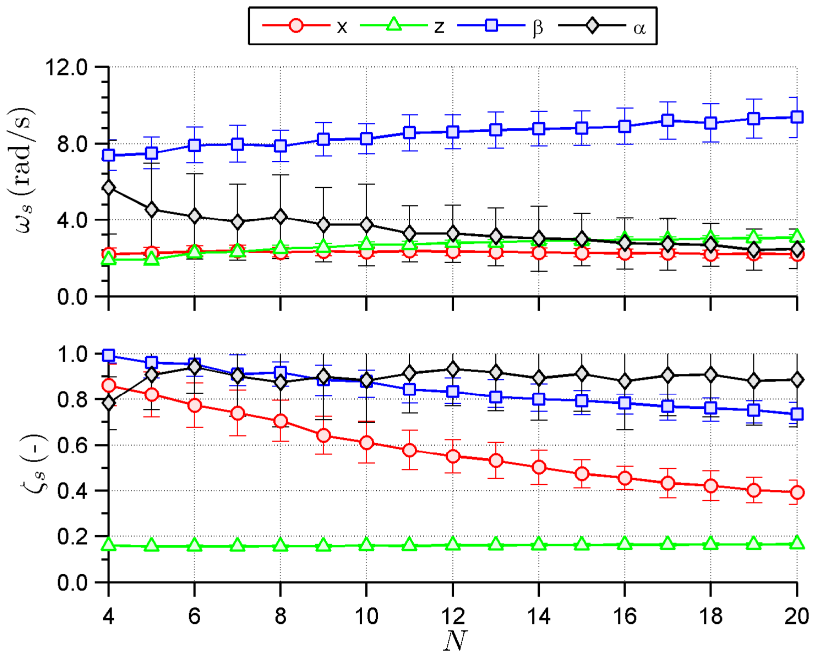
In Figure 5.2(b), we see that the change in optimal control parameters with respect to the vehicle's configuration is most pronounced in the DOF of roll γ and pitch β. This is to be expected since the ability for a vehicle to roll γ and pitch β, as well as the effect in which uncorrelated process noise has on these DOF is a function of its configuration, see (5.7.1) and (5.4.5) – (5.4.6), respectively.

5.8 Onboard Control Parameter Storage

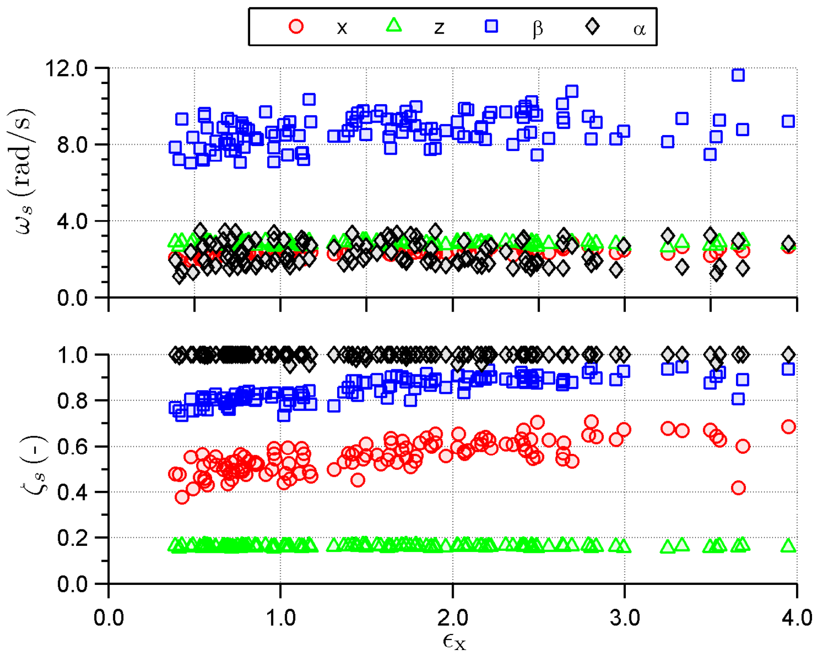
Due to the limited computational resources available on board the DFA, it would be impractical to compute or store the optimal parameters for our parameterized controller using the generalized optimization procedure described in Section 5.6. Instead, we make use of our analysis in Section 5.7 and show how these parameters can be compressed and efficiently stored on our embedded system.

Mapping Configuration to Control Parameter Space

One solution is to design a function that maps the size N and configuration space $(\epsilon_{\mathrm{x}}, \epsilon_{\mathrm{y}})$ of the DFA to its control parameter space (ω_s, ζ_s), where $s \in \{x, y, z, \gamma, \beta, \alpha\}$. Due to



(a) As the size of the vehicle increases, uncorrelated process noise effectively decreases and the control tuning parameters adjust to further reduce the $\mathcal{H}_2$ norm of the system. With a reduction in the expected exogenous input disturbance, the optimization may choose a more aggressive controller. The DOF in y and γ are not shown because they follow the same trend as x and β, respectively.



(b) The closed-loop optimal parameters for a set of 250 unique DFA configurations of a vehicle containing 12 modules. Most parameters maintain a relatively constant value except for the DOF in x and β. The DOF in y and γ are not shown, but follow a similar trend in ϵ_y – the results are symmetrical.

Figure 5.7.2 The control tuning parameters resulting from our optimization procedure are a function of the vehicle's size (a) and configuration (b).

symmetry of the problem, one can reduce the configuration space to just one of the two parameters. One way to obtain such a map is to surface-fit a function via least-squares to a dataset of optimal parameters – one function for each parameter (see Figure 5.2(b)). In general, the control parameters can be modeled as an exponential function with respect to size N and configuration ϵ_{x}. Some parameters (e.g. ω_x, ω_y, ζ_z, and ζ_α) can simply be assigned constant values, see Figure 5.7.2. For the range of data that has been generated (i.e. up to 100 unique configurations for each vehicle size), it is sufficient to use a first- and second-order polynomial function to model the values of the optimal parameters. For example, the parameters $p = \{\omega_z, \omega_\alpha\}$ can be modeled by a first-order polynomial of the form:

$$p = \mathrm{a}N + \mathrm{b}, \tag{5.8.1}$$

where a and b are constants. The remaining parameters, $p = \{\omega_\gamma, \omega_\beta, \zeta_x, \zeta_y, \zeta_\gamma, \zeta_\beta\}$, can be modeled as a first-order polynomial in N and second-order polynomial in ϵ_{x},

$$p = \mathrm{a}N + \mathrm{b}\epsilon_{\mathrm{x}}^2 + \mathrm{c}\epsilon_{\mathrm{x}} + \mathrm{d}\epsilon_{\mathrm{x}}N + \mathrm{e}, \tag{5.8.2}$$

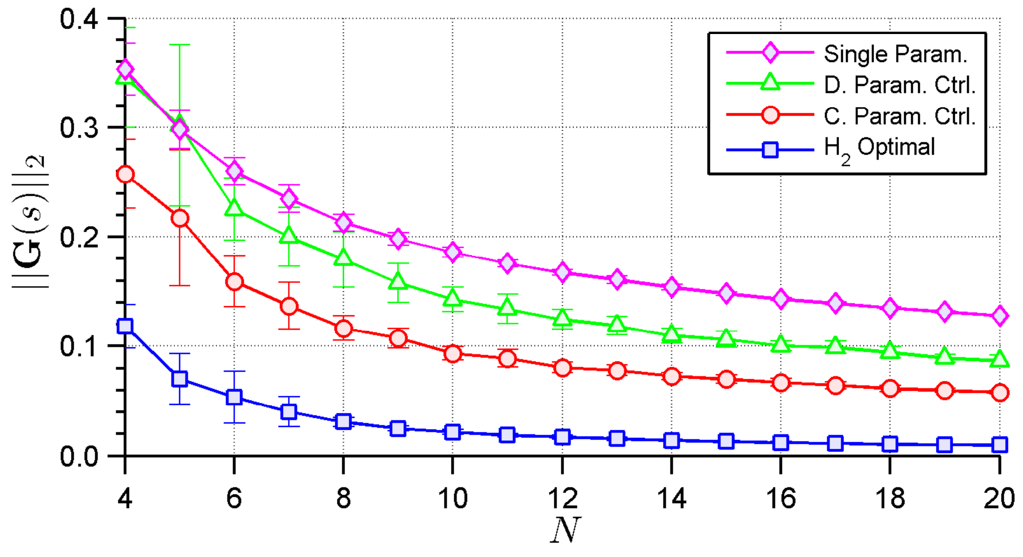
where $\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}$ are all constants.

The set of surface-fitted functions can then be tested against an independent dataset of optimal parameters. We can assess the fit by comparing the $\mathcal{H}_2$ norm of a system that uses the *fitted parameters* with one that uses the *optimal parameters.*

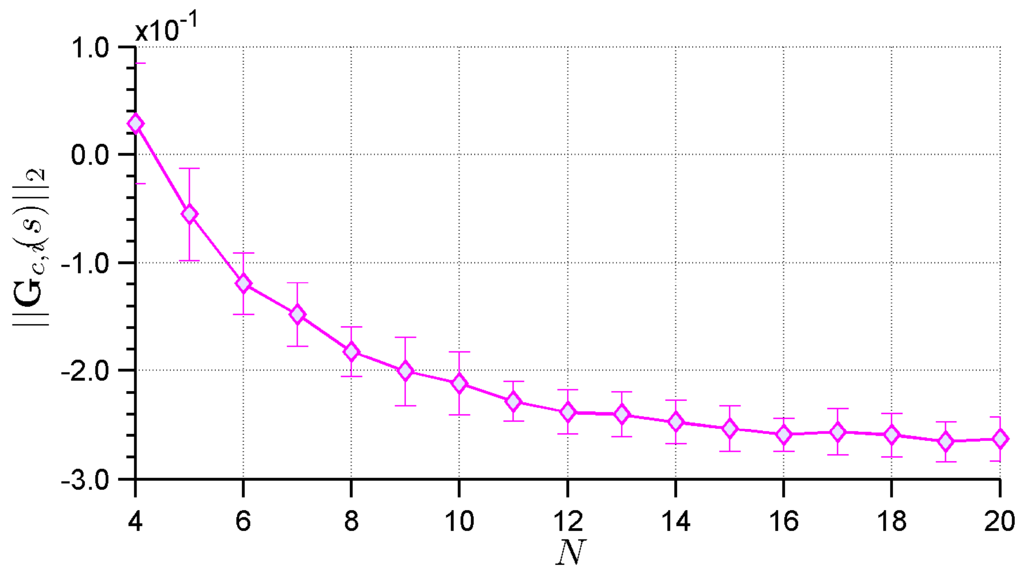
In summary, the only information that is needed in order to obtain reasonable control parameters for flying an arbitrary configuration of the vehicle is the size and configuration $(N, \epsilon_{\mathrm{x}})$ of the vehicle. This information can be computed before taking flight, see Appendix 5.B. The drawback to this method, however, is that there is no guarantee that the fitted parameters will not violate the vehicle's physical constraints, see Section 5.6. For practical purposes, however, this may be of little concern since these constraints are conservative.

Single Set of Parameters

An even simpler solution is to use a single set of parameters for all sizes and configurations. This is motivated by the fact that, practically speaking, control parameters change relatively little with respect to size and configuration, see Figure 5.7.2. For the majority of configurations, the performance will be far from optimal. However, from a practical point of view the performance may be sufficient. Based on our analysis in Section 5.7, the optimal parameters for large vehicles, in comparison to smaller ones, are relatively aggressive. Using such parameters on smaller sized vehicles will likely violate physical constraints. This, in fact, can be verified in simulation. On the other hand, choosing optimal parameters for small-size vehicles will in general not violate physical constraints for larger vehicles (see Figure 5.1(b)), however the performance will be relatively sluggish.



(a) Although worse than the optimal, the performance using a single set of parameters still improves with increasing vehicle size.



(b) Using control tuning parameters designed for small-sized vehicles will in general not violate physical constraints (below zero) for larger vehicles.

Figure 5.8.1 Performance (a) and constraint (b) statistics using a single set of parameters for various vehicle sizes and configurations. The single set of parameters used in this dataset are the mean optimal parameters for a 4-module vehicle. The dataset used here consists of 100 randomly chosen configurations for each vehicle size.

In the case of the DFA, the performance using a single set of parameters improves with size (see Figure 5.1(a), and simulation results show that the physical constraints are not violated. In this case one can choose, for example, the mean optimal parameters for a 4-module vehicle to be used for all vehicle sizes and configurations.

5.9 Experiments

We have successfully demonstrated modular flight for up to 12 modules, both indoors and outdoors, in a variety of configurations. In doing so, we validated the the linear model proposed in Section 5.1, the control strategy described in Section 5.5, and made use of the procedure for computing the optimal parameters described in Section 5.6.

Procedure

For the experiments performed indoors, the vehicle was operated in the workspace of a 3D motion capture system, see Section 5.3. This system is used for XY-position/velocity and yaw-angle feedback. It is also used as our source of ground truth for evaluating flight performance.

A variety of configurations were tested, some randomly assembled, others hand-picked in order to test the limits of the system, see Figure 5.9.1. After manually assembling the vehicle, commands such as *take-off* and *land* were transmitted to the vehicle over a wireless joystick interface. All control functions took place on board the vehicle. A single set of control parameters, as explained in Section 5.8, was used to fly all of the configurations tested in this experiment.

Immediately before take-off, each module executes a 3-second static calibration routine. During this period, a running average of the measurements obtained from its 3-axis rate-gyroscope is taken as its bias since the mean angular velocity is known to be zero. Upon completing its calibration routine, all modules simultaneously start its rotor and increases its angular velocity to equilibrium thrust.

Feedback from the 3D motion camera system is sent to the modules at a rate of 100 Hz. This is also the rate of the control loop. Each configuration was flown for approximately 30 seconds. The vehicle was then manually landed by joystick.

Nominal Performance Evaluation

Due to the lack of an integrator in the position controller, the vehicle does not necessarily hover at the commanded reference position. Rather, it has been observed to hover at a repeatable offset position. The cause of this offset may be due to propeller quality, deviations in manufacturing and assembly of the modules, and/or 3D motion capture system frame-misalignment. Since the effect is repeatable, they can be calibrated out. However, for the purpose of these experiments, this is not important and this calibration is left out to simplify experimental procedure.

The standard deviation σ_s of the regulated states $s \in \{x, y, z, \alpha\}$ is used as a metric for evaluating the flight performance of the vehicle,

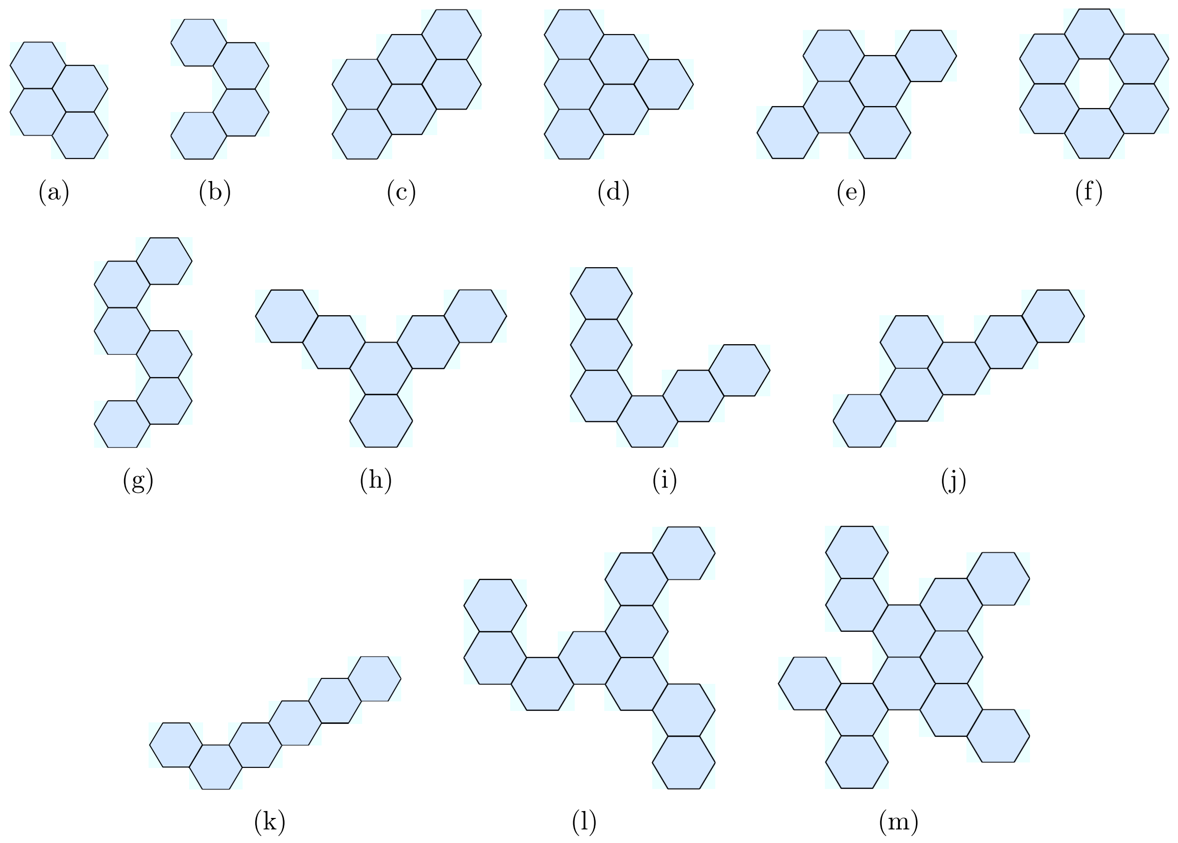


Figure 5.9.1 A variety of configurations, ranging from 4 to 12 modules, were used to evaluate the flight performance (listed in Table 5.9.1) of the vehicle indoors.

$$\sigma_s := \sqrt{\frac{1}{\mathrm{K}} \sum_{k=1}^{\mathrm{K}} (\hat{s}[k] - \bar{s})^2}, \tag{5.9.1}$$

where K is the total number of samples k of the measured state $\hat{s}$, and $\bar{s}$ is the state average taken over all samples. Measurements were made at a rate of 200 Hz.

Each configuration was flown over an undisturbed interval of 10 seconds. The standard deviation of the regulated states for each configuration (see Figure 5.9.1) is listed in Table 5.9.1.

For some configurations, namely the ones shown in Figure 5.1(a) and Figure 5.1(c), modules tend to *jitter*. This is clearly demonstrated in Extension 1. The cause of this has mainly to do with relatively high control gains (or tuning parameter values) used in flying the vehicle. This jitter, or high velocity content, is not necessarily minimized in the optimization procedure. Moreover, the single set of parameters used for flying all of the configurations does not neccessarily imply that all vehicles should behave in the same way.

Table 5.9.1 Standard Deviation of the Regulated States for Various Configurations in Flight

Config.	x (m)	y (m)	z (m)	α (rad)
(a)	0.025	0.022	0.028	0.012
(b)	0.052	0.042	0.036	0.035
(c)	0.011	0.011	0.010	0.002
(d)	0.024	0.026	0.018	0.004
(e)	0.054	0.044	0.027	0.023
(f)	0.020	0.029	0.018	0.004
(g)	0.014	0.013	0.013	0.023
(h)	0.019	0.035	0.014	0.012
(i)	0.011	0.027	0.014	0.006
(j)	0.021	0.029	0.030	0.031
(k)	0.030	0.033	0.031	0.027
(l)	0.031	0.026	0.017	0.027
(m)	0.015	0.031	0.017	0.004

Disturbance Rejection

In a different experiment, the vehicle was flown in the configuration shown in Figure 5.1(d) and was disturbed by pushing it in the lateral direction. Figure 5.9.2 shows this 25-second dataset, with the disturbance applied at 5-seconds and 18-seconds. The results demonstrate the tracking capability of the controller.

Outdoor Flight

Although performance evaluation was made indoors in a controlled environment, the system performs reasonably well outdoors, see Figure 5.9.3. In this case, joystick was used as feedback to control the lateral DOF (through roll and pitch angles) and yaw-angle.

5.10 Conclusions

This chapter documents the complete solution, spanning a flight dynamics model, estimation, control, and experimental results of a modular vertical take-off and landing vehicle, namely the Distributed Flight Array. With this hardware testbed, we have successfully demonstrated coordinated flight for an assortment of vehicle configurations and sizes. The results from our experiments validate the models used and proposed scalable control strategy. It has also given us insight into how the system can be improved.

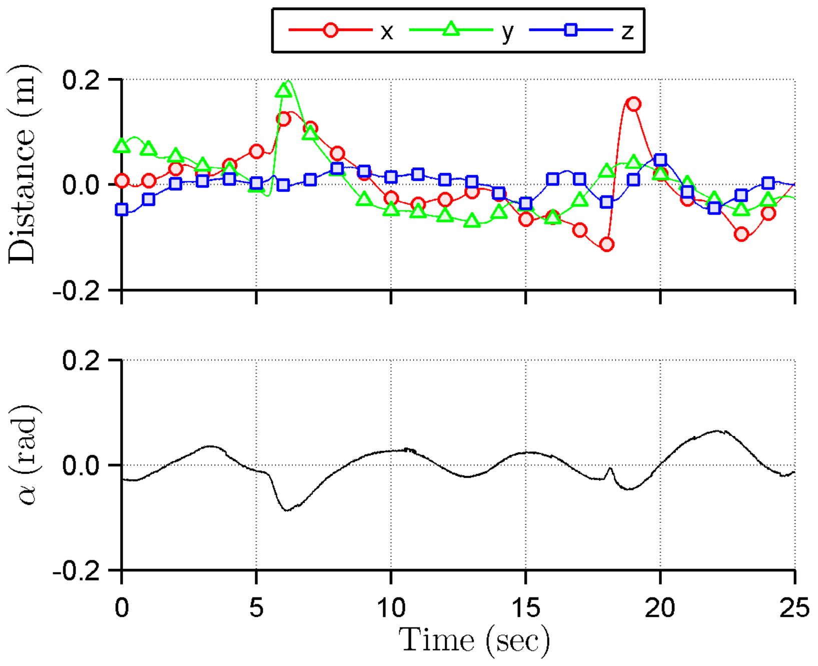


Figure 5.9.2 The vehicle is shown to recover from a disturbance applied to it by pushing it along the y-axis and then the x-axis, shown at 5-seconds and 18-seconds, respectively.

One of the main contributions of this work is a parameterized method for controlling a modular vertical take-off and landing vehicle of arbitrary size and configuration, as well as an approach to automatically compute the parameters of this controller that maximizes flight performance subject to the system's physical constraints. The control strategy we describe is scalable, using a finite set of parameters for any vehicle size and configuration. Our approach to computing the control parameters is general enough such that it can easily be extended to include different disturbance models apart from those already considered and it is straightforward to consider performance metrics apart from the one we used (i.e. $\mathcal{H}_2$ system norm).

In this work we compared our control strategy to other traditional centralized control methods in simulation, and although it demonstrates slightly lower performance, it gives us much more intuition into the behaviour of our system. More importantly, however, is that it lends its simplicity in enabling us to create a function that maps the configuration space of our vehicle to its control parameter space. We show that this map can be designed as a polynomial function. Thus, computing the control tuning parameters for *any* flight-feasible configuration is now nothing more than solving a few straightfoward equations.

The work described here provides a lower benchmark in terms of flight performance for this type of vehicle. Apart from sharing distance measurements between neighbours in order to estimate the vehicle's tilt, no other information is shared during flight. It will be interesting to quantify the benefits we obtain by sharing information in such a system and to evaluate the trade-offs, for example performance versus cost of communication. We would also like to explore coupled distributed estimation and control strategies, i.e.

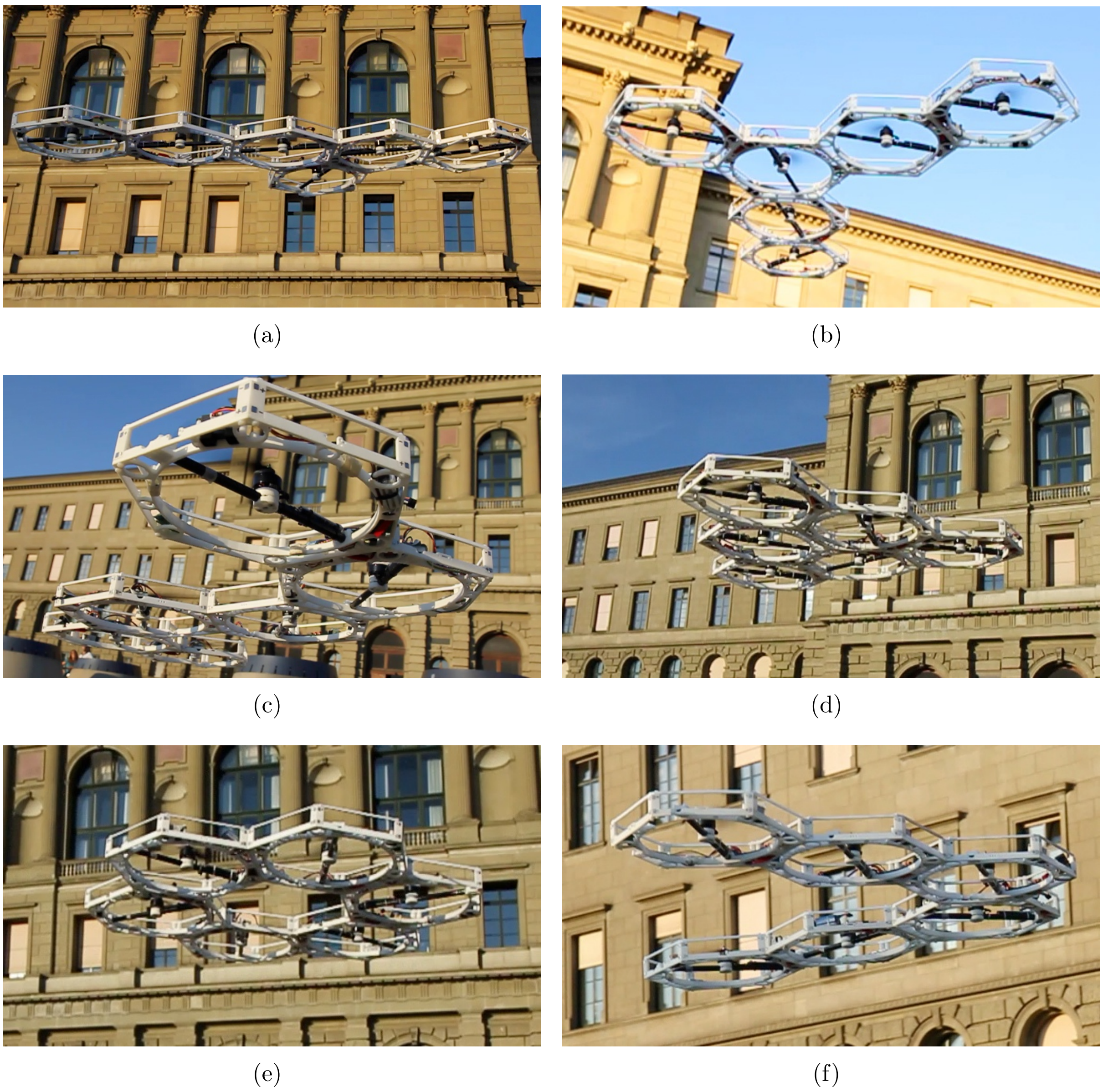


Figure 5.9.3 A variety of 6-module vehicle configurations flying outdoors using a joystick as position and yaw-angle feedback.

strategies that do not a priori split the task into first estimation and then control, which is what is currently implemented on the existing system. This scheme is supported by the results in [46]; they will have to be extended, however, to impose various physical constraints such as limited communication bandwidth.

Apart from improving the estimator and controller, flight performance can be improved with a stiffer chassis and a more rigid mechanical interface between modules, otherwise a more detailed physical model. One of the major assumptions in our model is that the system is thought to behave as a single rigid body. Experiments show this to be clearly false. However, the design of the system was not without any consideration. Apart from the requirements of a stiff chassis and one that is relatively robust such that it can be used reliably as an experimental platform, a major challenge was to keep the weight of the system to a minimum in order to maximize flight time and control authority. Such constraints result in a tradeoff in design requirements. The choice of both mechanical and electrical design only came about after various design iterations and improvements.

5.A Appendix: Physical Parameters of a Module

The physical parameters of a DFA module are listed in Table 5.A.1. All distance measurements are listed with respect to the geometric centre of a module (see Figure 5.A.1), where the origin along the vertical axis sits at the base of the module.

Table 5.A.1 Physical Parameters of a Module

Symbol	Description	Value
ℓ	Characteristic length	$0.250\,\mathrm{m}$
-	Position of the centre of mass along the x-axis	$0.009\,\mathrm{m}$
-	Position of the centre of mass along the y-axis	$-0.0196\,\mathrm{m}$
-	Position of the centre of mass along the z-axis	$0.0179\,\mathrm{m}$
-	Position of the distance measurement sensor along the x-axis	$-0.1086\,\mathrm{m}$
-	Position of the distance measurement sensor along the y-axis	$-0.0652\,\mathrm{m}$
-	Position of the distance measurement sensor along the z-axis	$0\,\mathrm{m}$
m	Mass	$0.255\,\mathrm{kg}$
$\mathrm{I_x}$	Mass moment of inertia around the x-axis	$0.00179\,\mathrm{kg\cdot m^2}$
$\mathrm{I_y}$	Mass moment of inertia around the y-axis	$0.00185\,\mathrm{kg\cdot m^2}$
$\mathrm{I_z}$	Mass moment of inertia around the z-axis	$0.00322\,\mathrm{kg\cdot m^2}$
$\lvert\mathrm{c}\rvert$	Force to torque conversion constant	$0.0164\,\mathrm{m}$

The module's centre of mass was measured simply by hanging a module from three orthogonal sides off a gimbal. The intersection of all three lines drawn along the gravitational axis through the gimbal is the module's centre of mass.

The mass moment of inertia I_a for $a \in \{\mathrm{x},\mathrm{y},\mathrm{z}\}$ was computed by swinging a module around the axis a of a gimbal connected to a rotary encoder and measuring the period of oscillation. The period of oscillation T for a physical pendulum is given by [47]

$$T = 2\pi\sqrt{\frac{\mathrm{I}_a}{\mathrm{mgh}}}, \tag{5.A.1}$$

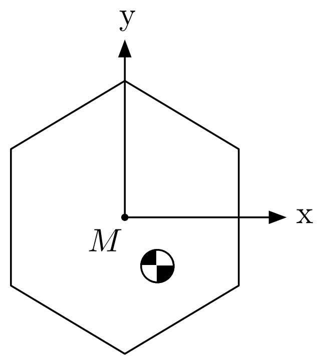


Figure 5.A.1 The module's coordinate frame M is located at the geometric centre of the chassis. The origin along the vertical axis (pointing out of the page) sits at the base of the module. The centre of mass is located off centre, see Table 5.A.1 for details.

where m is the mass of a module, g is the acceleration due to gravity, and h is the distance of the centre of mass from the axis of rotation. It is straightforward to solve this expression for I_a. The Parallel Axis (Huygens-Steiner) Theorem [48] was used transform this value to the module's coordinate frame M. This method is unable to measure off-diagonal terms in the mass moment of inertia tensor, which we assume to be negligible.

5.B Appendix: Physical Parameters of the Vehicle

Given the size and configuration of the vehicle, as well as the known physical parameters of a module (see Appendix 5.A), the mass and principal mass moment of inertia of the vehicle can be computed.

Configuration Database

Each module is capable of determinining the network topology and, in turn, the physical configuration of the vehicle that it may be connected to. This is accomplished by having each module flood the network with its own module identifier (ID), neighbour ID(s), and communication interface ID(s), see Figure 5.B.1; the latter is crucial in determining the orientation and physical position of a module with respect to another. Upon receiving such information from another module, each module stores this data in a database, similar in structure to a Laplacian matrix [49]. The Laplacian matrix can be used to find many properties of the network, such as connectivity and the number of spanning trees. The size of this matrix $N \times N$ denotes the size of the network (or the number of modules in the vehicle).

This approach is by no means efficient; it is nevertheless simple to implement and reliable for the number of modules that we work with (up to 12). More efficient techniques similar to the breadth-first *Localization Algorithm* proposed in [50] could, however, be implemented.

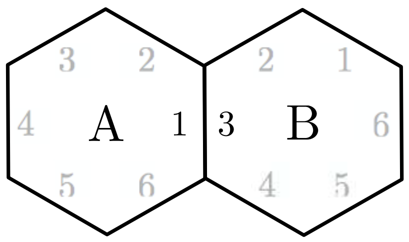


Figure 5.B.1 For any module-to-module connection, there are two pieces of information needed in order to determine the position and orientation of a module with respect to another: (1) module IDs, shown here as A and B; and (2) communication interface ID, shown here as 1 on module-A and 3 on module-B.

Centre of Mass

Let the position vector $\mathbf{r}_k = (x_k, y_k, z_k)$ denote the centre of mass of a module k with respect to a module's coordinate frame M, see Figure 5.B.2. This information is straightforward to obtain from the configuration database introduced previously. The vehicle's centre of mass (which is also the location of the vehicle's body coordinate frame B) with respect to a module's coordinate frame M is the average of all these positions $\frac{1}{N}\sum_{k=1}^{N} \mathbf{r}_k$.

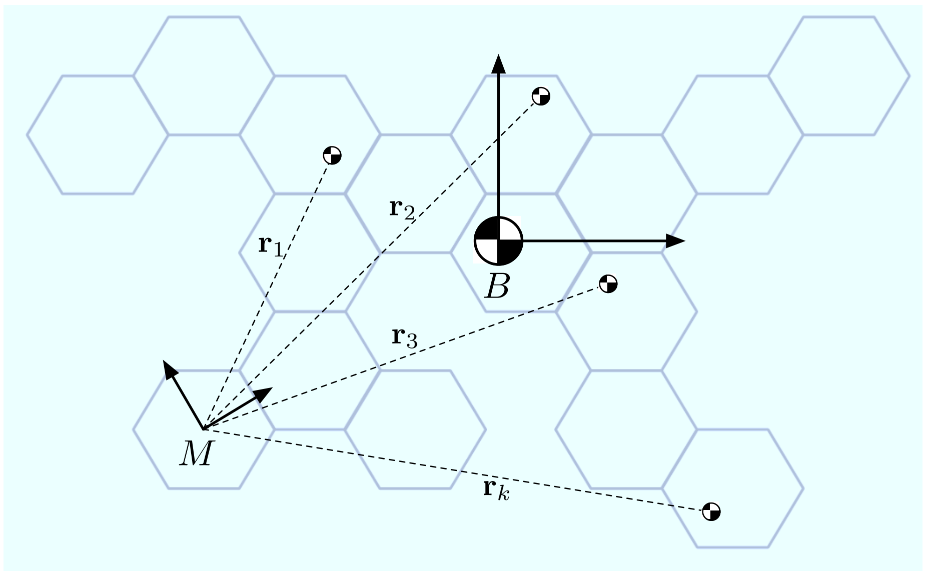


Figure 5.B.2 The vehicle's centre of mass B with respect to a module M can be computed by taking the average of all position vectors $\mathbf{r}_k$, which point from M to a module k's centre of mass.

Mass

The total mass of the vehicle is simply the product of the total number of modules N and the mass of a single module m,

$$\sum_{k=1}^{N} \mathrm{m} = N\mathrm{m}. \tag{5.B.1}$$

Principal Mass Moment of Inertia

Each module computes the principal mass moment of inertia with respect to an agreed upon module M, for example the module with the lowest ID. In this way, we can ensure that all modules compute the same principal mass moment of inertia, since the x and y axes are ambiguous.

The inertia tensor ${}_{M}\mathbf{J}$ of the vehicle with respect to the coordinate frame of module M is computed by repeated use of the Parallel Axis Theorem [48] over the total number of modules N. The elements ${}_{M}\mathbf{J}_{i,j}$ of this inertia tensor for row i and column j is given by:

$${}_{M}\mathbf{J}_{i,j} = \sum_{k=1}^{N} \left[\mathrm{I}_{i,j} + \mathrm{m}\left(||\mathbf{r}_k||^2 \delta_{i,j} - r_{k,i} r_{k,j}\right)\right], \tag{5.B.2}$$

where $\delta_{i,j}$ is the Kronecker delta, $\mathrm{I}_{i,j}$ are the elements a module's inertia tensor, which is given by $\mathrm{diag}(\mathrm{I_x}, \mathrm{I_y}, \mathrm{I_z})$ (off diagonal terms are assumed to be negligible), and $r_{k,i}$ and $r_{k,j}$ are respectively the i-th and j-th element of vector $\mathbf{r}_k$.

The inertia tensor ${}_{M}\mathbf{J}$ is then transformed to the vehicle's center of mass (also the location of the vehicle's body coordinate frame B) by once again applying the Parallel Axis Theorem. We denote this new inertia tensor as ${}_{B}\mathbf{J}$. The vehicle's *principal* mass moment of inertia $(I_\mathrm{x}, I_\mathrm{y}, I_\mathrm{z})$ can then be computed from the eigendecomposition of the inertia tensor ${}_{B}\mathbf{J}$, which is given by

$$\begin{bmatrix} I_\mathrm{x} \\ I_\mathrm{y} \\ I_\mathrm{z} \end{bmatrix} = \mathbf{V}^{-1} \cdot {}_{B}\mathbf{J} \cdot \mathbf{V}, \tag{5.B.3}$$

where the matrix $\mathbf{V}$ is a rotation matrix that rotates the coordinate frame of the inertia tensor ${}_{B}\mathbf{J}$ to the principal axes of rotation.

5.C Appendix: Nominal Thrust for an Over-Actuated System

In order to hover, the vehicle must satisfy four conditions: (1) the total thrust must counteract the acceleration due to gravity g, and (2) – (4). the net moment of the vehicle in roll, pitch, and yaw must be zero. In the case of the DFA, this implies having at least four modules with a pair of rotors that rotate CW and another pair that rotates CCW.

In general, however, the system may be over-actuated. In which case one should use a secondary criteria, such as minimizing the maximum amount of thrust produced. This can be formulated as a least squares squares problem, where the solution is in fact optimal (all thrusts are equal) when N is an even number. The system of equations can be written as

$$\begin{bmatrix} N\mathrm{mg} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \mathbf{P}^{\mathrm{T}} \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}, \tag{5.C.1}$$

where $\mathbf{P}$ is the configuration matrix given by (5.1.34). The solution to this requires taking the Moore-Penrose (or pseudoinverse) as follows:

$$\begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} = \underbrace{\mathbf{P}(\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}}_{\mathbf{Q}} \begin{bmatrix} N\mathrm{mg} \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{5.C.2}$$

5.D Appendix: $\mathcal{H}_2$ Optimal Controller

The $\mathcal{H}_2$ optimal controller used in Section 5.7 for performance comparison was computed using MATLAB's Robust Control Toolbox. The state-space description of the generalized plant in this case is given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_1\mathbf{w} + \mathbf{B}_2\mathbf{u}_d, \tag{5.D.1}$$

$$\mathbf{z} = \mathbf{C}_1\mathbf{x} + \mathbf{D}_{11}\mathbf{w} + \mathbf{D}_{12}\mathbf{u}_d, \tag{5.D.2}$$

$$\mathbf{y} = \mathbf{C}_2\mathbf{x} + \mathbf{D}_{21}\mathbf{w} + \mathbf{D}_{22}\mathbf{u}_d, \tag{5.D.3}$$

where all matrices are real matrices of compatible dimensions and the state consists of position, Euler angles, translational velocities, Euler angle rates, and the control effort produced by each rotor,

$$\mathbf{x} = (x, y, z, \gamma, \beta, \alpha, \dot{x}, \dot{y}, \dot{z}, \dot{\gamma}, \dot{\beta}, \dot{\alpha}, u_1, \ldots, u_N) \in \mathbb{R}^{12+N}. \tag{5.D.4}$$

The state transition matrix $\mathbf{A}$ for nominal flight dynamics of the DFA, including motor dynamics as described in Section 5.1, is given by

$$\mathbf{A} = \begin{bmatrix} 0_{6\times 6} & \mathbf{I}_{6\times 6} & 0_{6\times N} \\ \mathcal{A}_{2\times 6} & 0_{2\times 6} & 0_{2\times N} \\ 0_{4\times 6} & 0_{4\times 6} & \mathbf{M}^{-1}\mathbf{P}^{\mathrm{T}} \\ 0_{N\times 6} & 0_{N\times 6} & -\omega_r \mathbf{I}_{N\times N} \end{bmatrix}, \tag{5.D.5}$$

where

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & \mathrm{g} & 0 \\ 0 & 0 & 0 & -\mathrm{g} & 0 & 0 \end{bmatrix}. \tag{5.D.6}$$

The desired control efforts contained in vector $\mathbf{u}_d \in \mathbb{R}^{N\times 1}$ is fed into the matrix $\mathbf{B}_2$, which scales these values by the rotor's bandwidth ω_r,

$$\mathbf{B}_2 = \begin{bmatrix} 0_{12\times N} \\ \omega_r \mathbf{I}_{N\times N} \end{bmatrix}. \tag{5.D.7}$$

The state feedback vector $\mathbf{y} \in \mathbb{R}^{12\times 1}$ of the vehicle is a function of the system's state $\mathbf{x}$, not including the control efforts, therefore

$$\mathbf{C}_2 = \begin{bmatrix} \mathbf{I}_{12\times 12} & 0_{12\times N} \end{bmatrix}. \tag{5.D.8}$$

The expected exogenous disturbances contained in vector $\mathbf{w} \in \mathbb{R}^{18\times 1}$, which are described in Section 5.4, consist of unit variance process noise $\mathbf{w}_p$ and measurement noise $\mathbf{w}_m$,

$$\mathbf{w} = (\mathbf{w}_p, \mathbf{w}_m), \tag{5.D.9}$$

where

$$\mathbf{w}_p = (w_{\ddot{x}}, w_{\ddot{y}}, w_{\ddot{z}}, w_{\ddot{\gamma}}, w_{\ddot{\beta}}, w_{\ddot{\alpha}}), \tag{5.D.10}$$

$$\mathbf{w}_m = (w_x, w_y, w_z, w_\gamma, w_\beta, w_\alpha, \ldots \tag{5.D.11}$$

$$w_{\dot{x}}, w_{\dot{y}}, w_{\dot{z}}, w_{\dot{\gamma}}, w_{\dot{\beta}}, w_{\dot{\alpha}}). \tag{5.D.12}$$

The exogenous disturbance vector is scaled by the matrix $\mathbf{B}_1$ and the matrix $\mathbf{D}_{21}$ using the appropriate standard deviations listed in Table 5.4.1 of Section 5.4,

$$\mathbf{B}_1 = \left[\begin{array}{c|c} 0_{6\times 6} & \\ \mathrm{diag}(\sigma_{\ddot{x}}, \sigma_{\ddot{y}}, \sigma_{\ddot{z}}, \sigma_{\ddot{\gamma}}, \sigma_{\ddot{\beta}}, \sigma_{\ddot{\alpha}}) & 0_{(12+N)\times 12} \\ 0_{N\times 6} & \end{array}\right], \tag{5.D.13}$$

$$\mathbf{D}_{21} = \left[\begin{array}{cc} 0_{12\times 6} & \mathcal{D} \end{array}\right], \tag{5.D.14}$$

where,

$$\mathcal{D} = \mathrm{diag}(\sigma_x, \sigma_y, \sigma_z, \sigma_\gamma, \sigma_\beta, \sigma_\alpha, \ldots \tag{5.D.15}$$

$$\sigma_{\dot{x}}, \sigma_{\dot{y}}, \sigma_{\dot{z}}, \sigma_{\dot{\gamma}}, \sigma_{\dot{\beta}}, \sigma_{\dot{\alpha}}). \tag{5.D.16}$$

The output response $\mathbf{z}$, which is to be minimized through the synthesis of the $\mathcal{H}_2$ optimal controller, consist of the system's states and control effort weighted by the following two matrices, respectively:

$$\mathbf{C}_1 = \left[\begin{array}{cccccc|c} \mathrm{w_x} & 0 & 0 & 0 & 0 & 0 & \\ 0 & \mathrm{w_y} & 0 & 0 & 0 & 0 & \\ 0 & 0 & \mathrm{w_z} & 0 & 0 & 0 & 0_{4\times(6+N)} \\ 0 & 0 & 0 & 0 & 0 & \mathrm{w}_\alpha & \end{array}\right], \tag{5.D.17}$$

$$\mathbf{D}_{12} = \left[\begin{array}{c} 0_{4\times N} \\ \mathrm{w}_u \mathbf{I}_{N\times N} \end{array}\right]. \tag{5.D.18}$$

To compare the performance of this control method with the parameterized control strategy described in Section 5.6, the states are weighted similarly; the position weights $(\mathrm{w_x}, \mathrm{w_y}, \mathrm{w_z})$ are all set to 1 and the weight on the yaw-angle w_α is set to $\frac{1}{35}$ m/rad. Weight on the control effort, denoted w_u, is adjusted such that the control effort does not exceed the physical constraints, which can be formulated in the same way as described in Section 5.6.

There are no feed-through terms for the exogenous input $\mathbf{w}$ to the weighted output response $\mathbf{z}$ in (5.D.2), nor are there any for the control effort in (5.D.3). Hence, $\mathbf{D}_{11}$ and $\mathbf{D}_{22}$ are zero.

References

[1] M. Yim, D. G. Duff, and K. D. Roufas, "PolyBot: a modular reconfigurable robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 2000, pp. 514–520.

[2] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-TRAN: Self-reconfigurable modular robotic system," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 431–441, 2002.

[3] B. Salemi, M. Moll, and W. M. Shen, "SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3636–3641.

[4] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [Grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, March 2007.

[5] K. Gilpin and D. Rus, "Modular robot systems," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 38–55, 2010.

[6] D. D. Givone and R. P. Roesser, "Multidimensional linear iterative circuits – general properties," *IEEE Transactions on Computers*, vol. 100, no. 10, pp. 1067–1073, 1972.

[7] D. Swaroop and J. K. Hedrick, "String stability of interconnected aystems," *IEEE Transactions on Automatic Control*, vol. 41, no. 3, pp. 349–357, 1996.

[8] R. Horowitz and P. Varaiya, "Control design of an automated highway system," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 913–925, 2000.

[9] J. P. Lynch and K. H. Law, "Decentralized control techniques for large-scale civil structural systems," in *Proceedings of the International Modal Analysis Conference*, 2002.

[10] R. D'Andrea and G. E. Dullerud, "Distributed control design for spatially interconnected systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 9, pp. 1478–1495, 2003.

[11] J. M. Fowler and R. D'Andrea, "A formation flight experiment," *IEEE Control Systems*, vol. 23, no. 5, pp. 35–43, 2003.

[12] G. E. Stewart, D. M. Gorinevsky, and G. A. Dumont, "Feedback controller design for a spatially distributed system: The paper machine problem," *IEEE Transactions on Control Systems Technology*, vol. 11, no. 5, pp. 612–628, 2003.

[13] S. Jiang, P. G. Voulgaris, L. E. Holloway, and L. A. Thompson, "Distributed control of large segmented telescopes," in *American Control Conference.* IEEE, 2006, p. 6.

[14] M. Cantoni, E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan, "Control of large-scale irrigation networks," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 75–91, 2007.

[15] P. Varshavskaya, L. P. Kaelbling, and D. Rus, "Learning distributed control for modular robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2004, pp. 2648–2653.

[16] R. Fitch and Z. Butler, "Million module march: Scalable locomotion for large self-reconfiguring robots," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 331–343, 2008.

[17] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 5. IEEE, 2004, pp. 4393–4398.

[18] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1279–1291, 2012.

[19] R. Oung, A. Ramezani, and R. D'Andrea, "Feasibility of a Distributed Flight Array," in *Proceedings of the IEEE Conference on Decision and Control*, Shanghai, China, Dec. 2009, pp. 3038–3044.

[20] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Prentice Hall, Aug. 2004.

[21] J. G. Leishman, *Principles of Helicopter Aerodynamics*, 2nd ed. Cambridge University Press, 2006.

[22] K. F. Riley, P. Hobson, and S. J. Bence, *Mathematical Methods for Physics And Engineering: A Comprehensive Guide.* Cambridge University Press, 2006.

[23] M. Kriegleder, R. Oung, and R. D'Andrea, "Distributed altitude and attitude estimation from multiple distance measurements," in *Proceedings of the IEEE/RSJ*

International Conference of Intelligent Robots and Systems, Algarve, Portugal, Oct. 2012, pp. 3626–3632.

[24] D. Titterton, J. Weston, D. H. Titterton, and J. L. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. IET, 2004.

[25] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sept. 2004.

[26] M. Kriegleder, R. Oung, and R. D'Andrea, "Distributed altitude and attitude estimation using a network of distance measurement sensors," *To be submitted*, 2013.

[27] J. H. Kim, S. Sukkarieh, and S. Wishart, "Real-time navigation, guidance, and control of a UAV using low-cost sensors," in *Field and Service Robotics.* Springer, 2006, pp. 299–309.

[28] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A system for autonomous flight using onboard computer vision," in *IEEE International Conference on Robotics and Automation.* IEEE, 2011, pp. 2992–2997.

[29] P. J. Bristeau, F. Callou, D. Vissière, and N. Petit, "The navigation and control technology inside the AR. Drone micro UAV," in *Proceedings of the International Federation of Automatic Control World Congress*, vol. 18, no. 1, Aug. 2011, pp. 1477–1484.

[30] P. Stoica and R. L. Moses, *Introduction to Spectral Analysis.* Prentice Hall Upper Saddle River, NJ, 1997, vol. 51.

[31] F. Hoblit, *Gust Loads on Aircraft: Concepts and Applications.* AIAA, 1988.

[32] J. G. Leishman, "Rotorcraft aerodynamics," *Encyclopedia of Aerospace Engineering*, 2010.

[33] H. Huang, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering," in *IEEE International Conference on Robotics and Automation.* IEEE, 2009, pp. 3277–3282.

[34] O. Purwin and R. D'Andrea, "Performing aggressive maneuvers using iterative learning control," in *IEEE International Conference on Robotics and Automation.* IEEE, 2009, pp. 1731–1736.

[35] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Proceedings of the IEEE International Conference on Robotics and Automation.* IEEE, 2011, pp. 763–770.

[36] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[37] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *IEEE Conference on Decision and Control*, Dec. 2010, pp. 5420–5425.

[38] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods.* Prentice Hall Englewood Cliffs, 1990, vol. 1.

[39] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design.* Wiley, 2007, vol. 2.

[40] R. Oung and R. D'Andrea, "The Distributed Flight Array," *Mechatronics*, vol. 21, no. 6, pp. 908–917, Sept. 2011.

[41] H. Rake, "Step response and frequency response methods," *Automatica*, vol. 16, no. 5, pp. 519–526, 1980.

[42] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming," http://cvxr.com/cvx, Aug. 2012.

[43] W. Gawronski, *Advanced Structural Dynamics and Active Control of Structures.* Springer Verlag, 2004.

[44] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis, "State-space solutions to standard H2 and H∞ control problems," *IEEE Transactions on Automatic Control*, vol. 34, no. 8, pp. 831–847, 1989.

[45] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Johns Hopkins University Press, 1996, vol. 3.

[46] C. Langbort, R. Chandra, and R. D'Andrea, "Distributed control of heterogeneous systems interconnected over an arbitrary graph," *IEEE Transactions on Automatic Control*, vol. 9, no. 49, pp. 1502–1519, 2004.

[47] G. Fowles, *Analytical Mechanics*, ser. Holt-Saunders international editions: Physics. Holt, Rinehart, and Winston, 1977.

[48] T. R. Kane and D. A. Levinson, *Dynamics, Theory and Applications.* McGraw Hill, 1985.

[49] R. Merris, "Laplacian matrices of graphs: A survey," *Linear Algebra and its Applications*, vol. 197, pp. 143–176, 1994.

[50] K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu, "Miche: Modular shape formation by self-disassembly," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 345–372, 2008.

6 Conclusions, Perspectives, and Future Directions

This dissertation described the design and implementation of the Distributed Flight Array (DFA); a mobile modular robotics platform capable of coordinated mobility, both on the ground and in the air. In particular, it examines the challenges associated with controlling the joint behaviour of such a system, and within these challenges it provides some solutions. Such a testbed – from its concept phase, throughout its development, and to its use – abstracts many of the problems likely to appear in the next generation of multi-agent dynamic control systems. It has and will continue to provide a unique platform where traditionally independent aspects – control theory, communication, and computation – are tightly integrated. The methods developed in overcoming some of the various challenges may be of use to such highly integrated systems, which are likely to dominate the technological landscape in the near future.

In the following, we take a retrospective glance at the work accomplished, some of the lessons learned, and in a broad sense provide future directions.

General Design Considerations and Multi-Agent System Development

When it comes to hardware, test all assumptions and fail often. More difficult, however, is in realizing one's own assumptions as they may simply be taken for granted. For example, in designing Revision 1 of the DFA (see Chapter 3), it was assumed that the selected commercial off-the-shelf (COTS) propeller would behave similarily from one propeller to another, and that the same magnitude of thrust and torque would be produced regardless of the propeller's handedness. This was in fact not the case, and had a more thorough set of tests with the propellers been performed, one may have realized these irregularities before it was too late.

It is often said that individual agents or components of a multi-agent system can be simple in design. Nevertheless, developing such a system, where there are multiple components interacting, is not any less difficult – if anything it is more so. What was initially lacking was a set of automated testing procedures for quickly testing sub-systems on each agent. In particular, when developing methods where agents were required interact with one another, there were instances where the cause of the problem was not clear. For example, there were instances where the DFA was incapable of maintaining level-flight, see Chapter 5. There are a number of reasons for this, including failure in inter-module communication leading to an incomplete topology matrix, sensor malfunction, and actuator malfunction (see Chapter 3); each of these problems have been encountered on more than one occasion. The problem is in isolating the main cause of failure. In a system with a small number of agents, one can often get away without having to automate a test. There is a tradeoff that has to be made in the time to create an automated testing procedure and having to repeat the procedure manually. Generally speaking, a large multi-agent system is likely to require more infrastructure than its single-agent-equivalent for development purposes.

Cyber-Physical Systems Architecture

Despite an agent's relative simplicity, they are nevertheless highly integrated systems, combing both computational and physical elements. To successfully execute the design of such a system, a well thought-out design architecture is needed; taking not only the single agent into account, but all agents together. Before this can take place, however, one can greatly benefit from knowing all of the physical limitations and constraints existing in a design, which are not always so obvious. One unforseen limitation was the saturation of the rate-gyroscope measurements, discussed in Chapter 3, when operating at hovering thrust in Revision 1 of the DFA. The cause of this had primarily to do with a combination of poorly choosen propeller, motor mounting design, and chassis material.

Although it may seem that microprocessor speeds are even increasing, it is not a substitute for poor software architecture. As discussed in Chapter 3, the time-triggered architecture is a powerful framework for real-time embedded systems. In particular, for systems with many peripherals on a single microcontroller such as the DFA, this architecture greatly facilitated development by being able to quickly isolate faults at an agent-level. Isolating faults in an *event*-triggered architecture, where there are multiple sources of events, is far from trivial.

Throughout the design and development process it became clear that a streamlined and more processor-cycle-efficient solution could be achieved if sensing and estimation, communication handling functions, and high-level control was managed by their own processor. Dividing the various processes physically like this also helps with development by more clearly isolating faults. Future cyber-physical systems, which integrate sensing, communication, actuation, and computation on a single device may benefit from a embedded system's architecture of the form shown in Figure 6.0.1. Such an architecture could be implemented in future revisions of the DFA with very few changes to the firmware, thanks to the time-triggered architecture employed there.

Although the DFA was designed to operate autonomously, it was clear during the initial states of development that remote control of the vehicle was needed. What was originally considered to be a temporary solution, requiring a COTS wireless solutions, later became a much larger problem that required two solutions, see Chapter 3. The

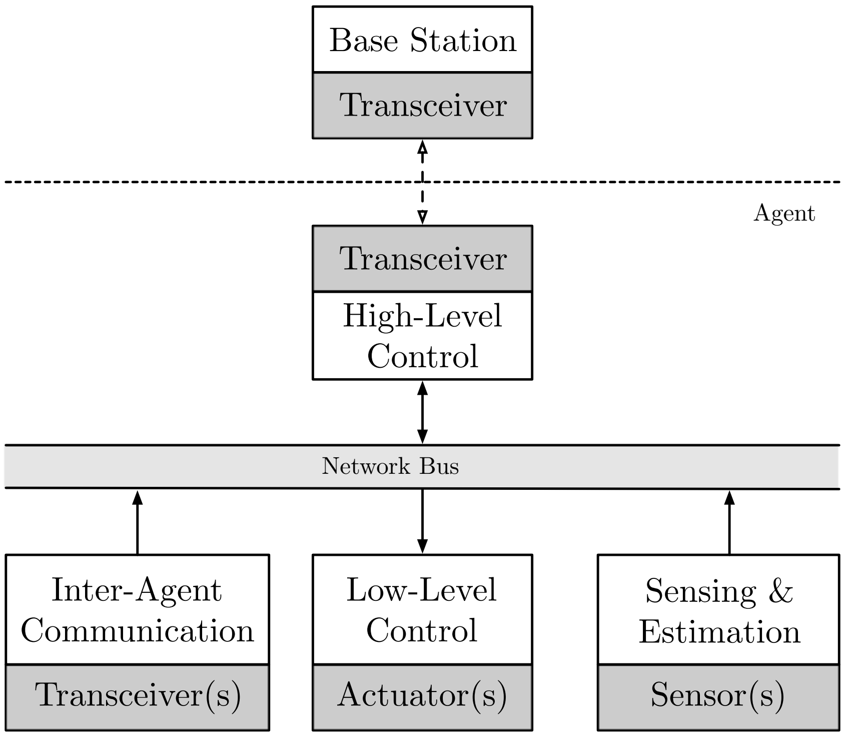


Figure 6.0.1 An embedded system's architecture that physically separates sensing, communication, low-level control with actuator interface, and high-level control from one another has a variety of advantages; important to a developer is the ability to more easily identify faults.

wireless spectrum available for local area networks has a physical limit – a communication bandwidth limit – which may be taken for granted. The symptoms, such as latency and packet loss, are not often observed until there is a significant amount of users in the same area. Such seemingly random occurances can be misleading to a developer. Active wireless technologies, such as wireless communication devices and wireless sensing technologies, should be thoroughly tested with multiple units – ideally the same number of units intended for simultaneous use.

Today's active wireless sensing technology [1] was not necessarily designed for multiple agents (or sensors) operating simultaneously in the same environment. This refers in particular to the infrared distance measurement sensors on board each DFA module, but also to a broad class of sensors ranging from ultrasonic rangefinders to motion sensing devices (e.g. Microsoft Kinect). The active signals eminating from such devices tend to interfere with one another, often leading to misrepresented data. To be useful in the future, such sensors need to be designed to operate effectively together or methods need to be developed to handle the operation of such sensors to eliminate interference, such as the work of [2] for the Microsoft Kinect.

Augmented Perception using Sensor Networks

We demonstrated in Chapter 4 that by using a network of distance measurement sensors, one can infer the tilt of a rigid body, which would otherwise be impossible to do using just one of these sensors alone. The addition of a communication network, in this case, added a new dimension or perception modality to the sensors.

It would therefore be interesting as future work to explore other ways of exploiting the communication network and the existing suite of sensors in order to perceive states that would otherwise be difficult or impossible to obtain using just a single sensor alone. For example, consider using the IR transceivers to not only infer heading to another module, as is described in Appendix B with the help of its rate-gyroscope, but to also infer distance. This could be accomplished with two or more modules, where the transceivers coordinate with others in the network to triangulate the position of another module.

Another example would be to use the distance measurement sensors to keep the vehicle flying within an invisible boundary. The detection method for this boundary could be a step in the surface over which the vehicle flies. Given prior knowledge of the shape and size of the boundary, and an agent's ability to coordinate with its peers, it may be possible to estimate the position of the vehicle with respect to the centre of the enclosure.

Multi-Agent Reliability

It is often said that a multi-agent system has the potential to be more reliable than its single-agent-equivalent – if an agent is lost or becomes defective, another simply takes its place. The irony of this statement, however, is perhaps the lacking notion of multi-agent reliability (or robustness) in literature. Just because an algorithm may be distributed does not inherently lead to reliability. There is therefore a need for an analytical tool that could determine the robustness properties of an algorithm and to synthesize robust algorithms systematically.

Prior to this, however, is a need to address the variety of practical failure modes. This has been explored extensively by the computer science and networking community. Some of the more practical faults that are likely to appear in real world systems are those categorized by Byzantine failures [3]. This type of fault encompasses two failure modes:

Omission Failure An agent may cease to communicate with other agents (e.g. system crash, link failure) [4]. This type of failure is addressed in Chapter 4 where we discuss a robust implementation of distributed average consensus.

Commission Failure An agent may process a request incorrectly, corrupting its local information, and possibly send an incorrect response to a request.

There are a variety of methods in which to handle failure. What enables the distributed algorithm in Chapter 4 to recover from Omission failure was the clever use of a buffering variable. Another method could be to divide the problem into local-agent fault detection [5] and distributed algorithm. In these cases, robustness is achieved through information redundancy – every agent receives a sufficient amount of information to detect a faulty agent and does something in order to recover from its effects.

Network Topology

It may be of interest to more thoroughly investigate the role of network topology in distributed systems. Both the distributed tilt estimation method described in Chapter 4 and the decentralized control strategy described in Chapter 5, to some extent, rely on knowing the physical topology of the vehicle, which can be derived from its network topology. More precisely, each agent needs to know its position with respect to some agreed upon reference frame. An agent can compute this provided the vehicle's physical topology. To agree upon a reference frame in a distributed manner, however, is not possible. It was shown in [6, 7] that agreement is possible on the position of a reference frame but not on its orientation. As such, the methods described in the two previously mentioned chapters require some knowledge about the entire system, and one can argue that these methods are therefore not completely distributed or decentralized.

One can see the importance of this by considering an instance during which the topology of a vehicle changes (i.e. disassembly or assembly of the DFA during flight). Before the next control step, the topology needs to be recomputed. Depending on the network size and connectivity, this transition period where the vehicle is not controlled may render itself incapable of recovering from its state after the controller sets in. In this case, it may be interesting to find ways of handling such situations gracefully.

What may be of interest is to design algorithms that simply do not depend on the complete topology of the system, in a similar fashion to the work described in [8]. On the other hand, computing the topology may be reasonable for some applications. In such cases, algorithms have a clear advantage provided the additional information about the network. For example, it is possible to impose an artificial network on top of the existing network topology in order to improve the algorithm's performance [9, 10]. In this case, it may be of interest to investigate the possibility of designing a topology for a distributed algorithm that achieves a desired performance.

Redundancy

When a multi-agent system has more agents than needed to perform a given task, the system becomes redundant. In the case of the DFA, when there are more than four modules in the vehicle, the system becomes overactuated for flight. This leads to some interesting questions on how resources should be optimally allocated in a distributed system. For example, in the case of the DFA, it may be of interest to investigate the type of configurations that result in increased stability of the vehicle, or the type of configurations that result in its ability to perform very aggressive maneuvers.

Redundancy adds this unique capability for agents to take control of another agent in the event that this other agent's system has been compromised. It may be of interest to investigate the design requirements of such a system and how this exchange of control could take place. Another area of interest, but also likely to be very challenging, would be to achieve reconfiguration in flight. To disassemble, one can use a heurisitic methodology of simply controlling sub-clusters of the vehicle.

Self-Assembly

The bulk of this dissertation described estimation and control strategies for abritrary configurations of the DFA. Nothing yet has been said about how modules assemble themselves to achieve such configurations. Mobile multi-agent systems, such as the DFA, are a classic testbed for decentralized or distributed self-assembly algorithms, known more commonly in literature as multi-agent rendezvous problems [11]–[15]. We provide a sketch of how this might be accomplished in Appendix B.

References

[1] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots.* MIT press, 2004.

[2] F. Faion, S. Friedberger, A. Zea, and U. D. Hanebeck, "Intelligent sensor-scheduling for multi-Kinect-tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 3993–3999.

[3] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[4] N. A. Lynch, *Distributed algorithms.* Morgan Kaufmann, 1996.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[6] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots-formation and agreement problems," in *Proceedings of the 3rd International Colloquium on Structural Information and Communication Complexity*, 1996, pp. 313–330.

[7] ——, "Distributed anonymous mobile robots: Formation of geometric patterns," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1347–1363, 1999.

[8] P. Varshavskaya, L. P. Kaelbling, and D. Rus, "Learning distributed control for modular robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2004, pp. 2648–2653.

[9] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, Sept. 2004.

[10] R. Olfati-Saber, "Ultrafast consensus in small-world networks," in *Proceedings of the American Control Conference.* IEEE, 2005, pp. 2371–2378.

[11] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, “Distributed memoryless point convergence algorithm for mobile robots with limited visibility,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 818–828, 1999.

[12] Z. Lin, B. Francis, and M. Maggiore, “Getting mobile autonomous robots to rendezvous,” *Control of Uncertain Systems: Modelling, Approximation, and Design*, pp. 119–137, 2006.

[13] J. Lin, A. S. Morse, and B. D. O. Anderson, “The multi-agent rendezvous problem. Part 1: The synchronous case,” *SIAM Journal on Control and Optimization*, vol. 46, no. 6, pp. 2096–2119, 2007.

[14] ——, “The multi-agent rendezvous problem. Part 2: The asynchronous case,” *SIAM Journal on Control and Optimization*, vol. 46, no. 6, pp. 2120–2147, 2007.

[15] S. Martınez, J. Cortes, and F. Bullo, “On robust rendezvous for mobile autonomous agents,” in *International Federation of Automatic Control World Congress*, 2005.

A

Appendix: Driving Kinematics

Much of the work described in this dissertation focused on issues required for coordinated flight. The Distributed Flight Array (DFA), however, is also capable of moving on the ground using its set of omni-directional wheels, see Chapter 3. This appendix briefly discusses the driving aspects of the DFA.

Outline

Section A.1 provides the kinematic transforms for a 3-wheel omni-directional drive vehicle; in particular it provides the transforms necessary to move a single DFA module, as well as a random assembly of modules. Section A.2 describes some of the physical limitation, which need to be handled when coordinating the movement of DFA modules on the ground.

A.1 Kinematics

Single Module

Let the body-fixed coordinate frame of a module, denoted M, be located at its geometric center aligned as shown in Figure A.1.1. The translational velocity components ($\dot{x}_{\mathrm{M}}$, $\dot{y}_{\mathrm{M}}$) and angular velocity $\dot{\alpha}_{\mathrm{M}}$ of a module with respect to an inertial coordinate frame is captured by the vector ${}_{\mathrm{M}}\mathbf{v} = (\dot{x}_{\mathrm{M}}, \dot{y}_{\mathrm{M}}, \dot{\alpha}_{\mathrm{M}})$.

The velocity of a module can be controlled by varying the individual velocities of each wheel (v_1, v_2, v_3). Given that the wheels are separated by $\pi/3$, the kinematic relationship is the following:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = {}^{\mathrm{M}}_{\mathrm{v}}\mathbf{T}(\mathrm{d}, \theta) \cdot {}_{\mathrm{M}}\mathbf{v}, \tag{A.1.1}$$

where d = 115 mm and $\theta = 15°$ are physical parameters of a module that captures the position of a wheel (i.e. closest to x-axis) with respect to the module's body coordinate frame (see Figure A.1.1), and

$${}^{\mathrm{M}}_{\mathrm{v}}\mathbf{T}(\mathrm{d}, \theta) := \begin{bmatrix} -\mathrm{s}\theta & \mathrm{c}\theta & \mathrm{d} \\ -\mathrm{s}(\frac{\pi}{3} - \theta) & -\mathrm{c}(\frac{\pi}{3} - \theta) & \mathrm{d} \\ \mathrm{s}(\frac{2\pi}{3} - \theta) & \mathrm{c}(\frac{2\pi}{3} - \theta) & \mathrm{d} \end{bmatrix}, \tag{A.1.2}$$

where c and s are shorthand forms of cosine and sine, respectively.

Assembly of Modules

Let B denote the body-fixed coordinate frame of a vehicle (i.e. an assembly of two or more modules). The translational velocity components (${}_{\mathrm{B}}\dot{x}$, ${}_{\mathrm{B}}\dot{y}$) and angular velocity of a vehicle with respect to an inertial coordinate frame is captured by the vector ${}_{\mathrm{B}}\mathbf{v} = ({}_{\mathrm{B}}\dot{x}, {}_{\mathrm{B}}\dot{y}, {}_{\mathrm{B}}\dot{\alpha})$.

The velocity of a vehicle can be controlled by appropriately varying the velocities of each module i located at $\mathbf{r}_i = (r_{x,i}, r_{y,i})$ with respect to the vehicle's body coordinate frame B. This can be accomplished by transforming the desired velocity in the vehicle's body coordinate frame to a module's body coordinate frame, which is captured in the following expression:

$${}_{\mathrm{M}}\mathbf{v} = {}^{\mathrm{B}}_{\mathrm{M}}\mathbf{T}(\mathbf{r}_i, \phi_i) \cdot {}_{\mathrm{M}}\mathbf{v}, \tag{A.1.3}$$

where ϕ_i is the orientation of module i with respect to the vehicle's body coordinate frame B (see Figure A.1.2), and

$${}^{\mathrm{B}}_{\mathrm{M}}\mathbf{T}(\mathbf{r}_i, \phi_i) := \begin{bmatrix} -\mathrm{c}\phi_i & \mathrm{s}\phi_i & r_{x,i} \cdot \mathrm{s}\phi_i - r_{y,i} \cdot \mathrm{c}\phi_i \\ -\mathrm{s}\phi_i & \mathrm{c}\phi_i & r_{x,i} \cdot \mathrm{c}\phi_i + r_{y,i} \cdot \mathrm{s}\phi_i \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.1.4}$$

Thus, given the position and orientation of a module with respect to an agreed upon vehicle coordinate frame, it is possible for the modules to control the joint velocity-

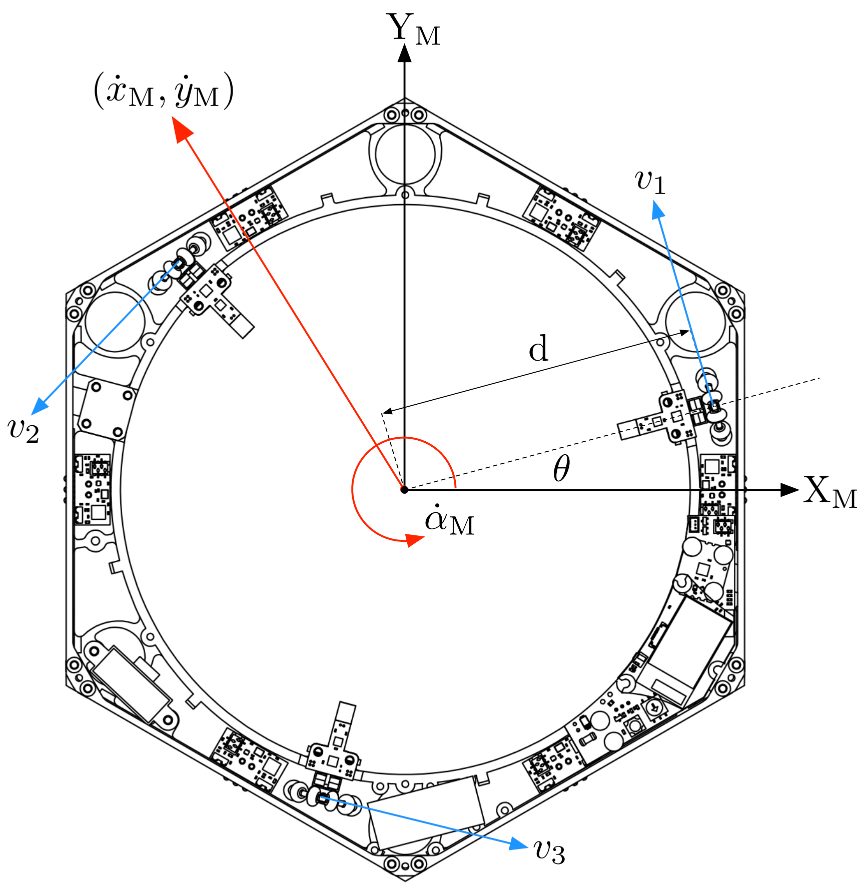


Figure A.1.1 A DFA module's body-fixed coordinate frame M is located at its geometric center and is aligned as shown in the figure above. Omni-directional wheels are positioned at a distance of d away from the center and are numbered counterclockwise, starting from the x-axis. The nearest wheel to the module's x-axis is positioned θ away from this axis; wheels are separated by $\pi/3$.

behaviour of the vehicle in a decentralized manner. These parameters can be computed given the physical topology of the vehicle, which can be determined using various means [1, 2].

A.2 Physical Constraints

The motors that drive the wheels impose a physical limit on the maximum velocity of a module and in turn the the maximum velocity of a vehicle. Let the maximum velocity of a wheel be denoted $\mathrm{v}_{\mathrm{max}}$, and the desired velocity heading of a module with respect to its coordinate frame M be given by the angle β. Then the maximum permissible translation speed in that heading is given by

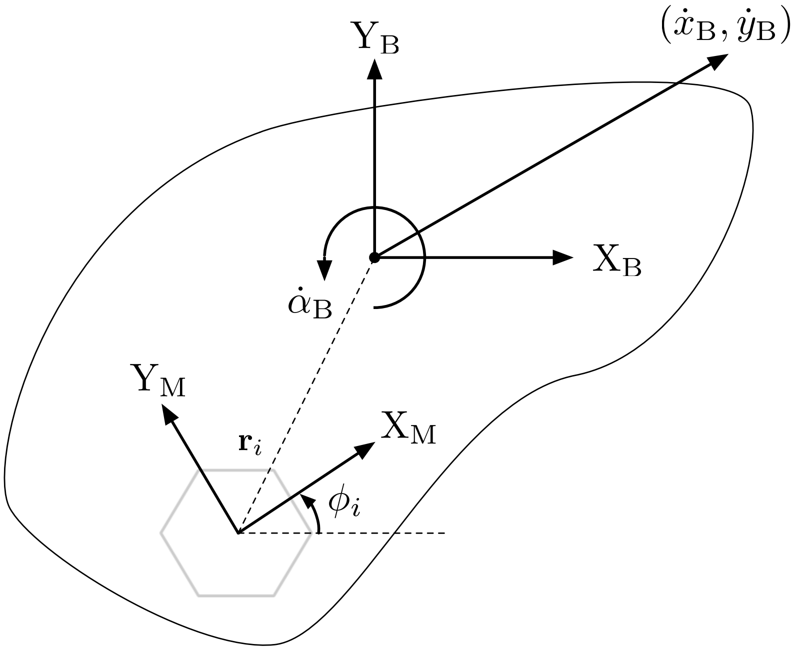


Figure A.1.2 The velocity of a vehicle $(\dot{x}_B, \dot{y}_B, \dot{\alpha}_B)$ can be controlled by varying the velocities of each module $(\dot{x}_M, \dot{y}_M, \dot{\alpha}_M)$, where a module i is positioned $\mathbf{r}_i$ away from the vehicle's coordinate frame B and rotated by ϕ_i.

$$V_{\max} = \min \left\{ \frac{\mathrm{v}_{\max} - d \cdot \dot{\alpha}}{{}^{\mathrm{M}}_{\mathrm{v}}\mathbf{T}_{j1} \cdot \cos\beta + {}^{\mathrm{M}}_{\mathrm{v}}\mathbf{T}_{j2} \cdot \sin\beta}, \quad j \in \{1, 2, 3\} \right\}, \tag{A.2.1}$$

where ${}^{\mathrm{M}}_{\mathrm{v}}\mathbf{T}_{jk}$ denote the element at row j and column k of the matrix ${}^{\mathrm{M}}_{\mathrm{v}}\mathbf{T}$. The maximum permissible angular speed is given by

$$\dot{\alpha}_{\max} = \frac{\mathrm{v}_{\max}}{\mathrm{d}}. \tag{A.2.2}$$

The preceeding expressions summarize the speed envelope of a module. Figure A.2.1 illustrates the speed envelope with respect to different heading angles and angular velocities. The speed envelope for an assembly of modules will depend on its physical topology. The maximum speed can be computed by stepping through each module in the vehicle (or assembly) and using an expression similar to (A.2.1), while taking into account the position and orientation of a module with respect to the body coordinate frame of the vehicle. In terms of rotation-only, the maximum permissible angular velocity of a vehicle will in general decrease as the size of the vehicle increases.

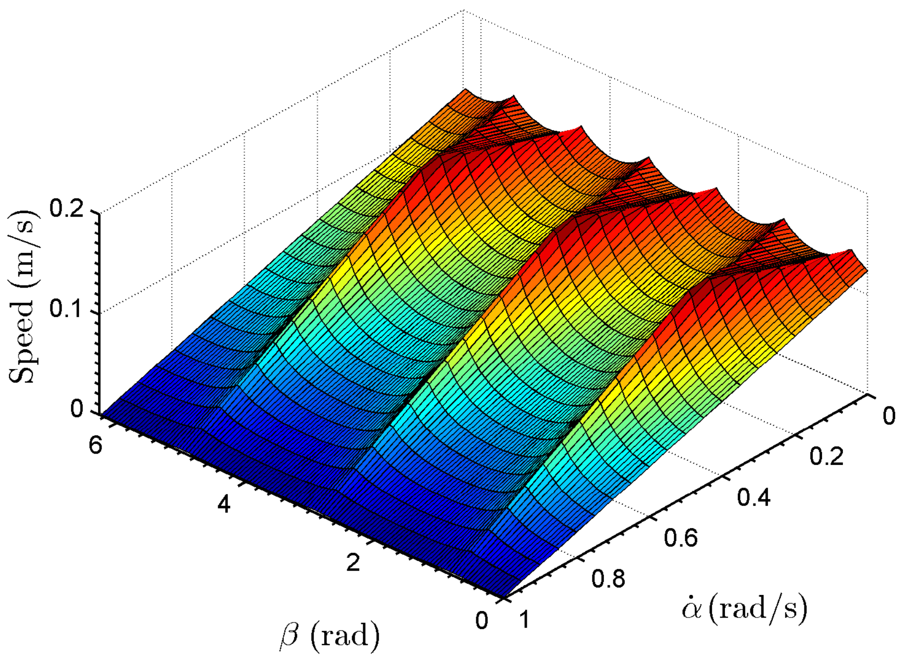


Figure A.2.1 Speed envelope of a module covering the entire range of heading angles β and angular velocities $\dot{\alpha}$ that range from zero to its maximum permissible value defined in (A.2.2).

A.3 Angular Velocity Control

Due to variety of non-idealities, such as manufacturing and wheel slip, simply employing the kinematic relations of Section A.1 and physical constraints of Section A.2 will not guarantee a straight-line motion when commanded to do so. To improve on this behaviour, one can employ the existing onboard rate-gyroscope to control the angular rate of a module. In practice, it is sufficient to use a proportional controller with a feed-forward term for angular rate. This minimizes deviation from a reference when commanded for example to move along a straight line. Nevertheless, due to rate-gyro bias it will still result in some rotation.

References

[1] K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu, "Miche: Modular shape formation by self-disassembly," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 345–372, 2008.

[2] R. Oung and R. D'Andrea, "The Distributed Flight Array: Design, implementation, and analysis of a modular vertical take-off and landing vehicle," *International Journal of Robotics Research, Under Review (Since Jan. 2013).*

B

Appendix: Self-Assembly

Much of the work described in this dissertation focuses on issues after the assembly of modules. Since the modules on their own are mobile on the ground, and an assembly of modules are mobile in the air, it would be interesting as future work to investigate autonomous self-assembly methods. This appendix provides an outline for how this can be accomplished using the existing set of onboard sensors on a module, see Chapter 3.

Outline

Section B.1 discusses how modules (or agents) can use its existing set of onboard peripherals to detect other agents within its vicinity. We then provide an outline for a generalized self-assembly algorithm in Section B.2.

B.1 Sensing Other Modules

There are no sensors available on a DFA module that will directly measure the position of another module with respect to itself. Instead, a module can employ its line-of-sight infrared transceivers in combination with its onboard rate-gyroscope to estimate the heading of other modules that are within range of its transceivers.

This is a two part method:

1. A module freely transmits at a periodic interval a packet of information over its transceivers (e.g. module identification number)

2. A module rotates in place, receiving the same transmitted packet from other modules. It uses its rate-gyroscope to estimate the relative angular heading from which the packet was received.

This method effectively provides a module with the relative heading of other modules within its vicinity. In reality, the transceivers are not line-of-sight, but rather transmit and receive signals through a cone of roughly 5°, see Chapter 3. One can simply use the mean-likelihood estimate of the signals received over this cone. Another issue is that the heading angle estimated by integrating rate-gyroscope measurements will be subject to drift, due to bias errors. This, however, can be minimized since a module can stop rotating at any time to recalibrate its measurements.

The following provides a sketch of a decentralized multi-agent algorithm, which makes use of the previously described heading information, that will cause agents in the system to move towards one another. Analysis of this method is left as future work.

B.2 Self-Assembly Algorithm

The purpose of self-assembly, within the context of the DFA, is to have the modules autonomously assemble in a decentralized or distributed way, i.e. without a supervisor or a global coordinate frame. In this scenario, there is no need for modules to assemble in a particular configuration. It is sufficient if the modules simply assemble at random. However, in order to do so they must move towards one another, or converge on a particular location. As such, the strategy is to bring modules within a small enough region that would ensure collision with one another and therefore achieve self-assembly behaviour.

The scenario that we describe falls into a class of problems known in literature as multi-agent rendezvous problems [1]–[5]. More generally, we can state our particular problem as follows: Consider a set of mobile autonomous agents labelled 1 through N, which can all move in a single plane. Each agent is capable of sensing other agents in its vicinity within a radius of $\mathrm{R}_{\mathrm{max}}$, centered at the agent's current position, see Figure B.2.1. Sensing another agent, in this case, refers to an agent's ability to infer the *direction* to another agent, but not distance (see previous section).

The ability to sense an agent is assumed to be mutual, i.e. if agent i is visible to agent j, then j is also visible to i. Our system can therefore be modelled as an undirected and connected *mutual visibility* graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is a nonempty, finite set, and $\mathcal{E}$ is a subset of $\mathcal{N} \times \mathcal{N}$ satisfying $(k, k) \notin \mathcal{E}$ for all $k \in \mathcal{N}$. The elements of $\mathcal{N}$ are called nodes (or agents), and an element $\{i, j\}$ of $\mathcal{E}$ is referred to as an edge, which represents the ability of an agent i to detect agent j. The set of agents that are visible to an agent i is denoted by $\mathcal{N}_i = \{j | \{i, j\} \in \mathcal{E}\}$. To simplify notation, we may use i and j to respectively denote agent i and agent j in the remainder of this work.

The assumptions that we will make are the following:

1. Agent motion is restricted to movement on a single plane.
2. Agents are modelled as points; occlusion during its sensing period and collision between agents during its maneuvering period are ignored.
3. Practical issues such as measurement noise, process noise, and time delays are ignored.

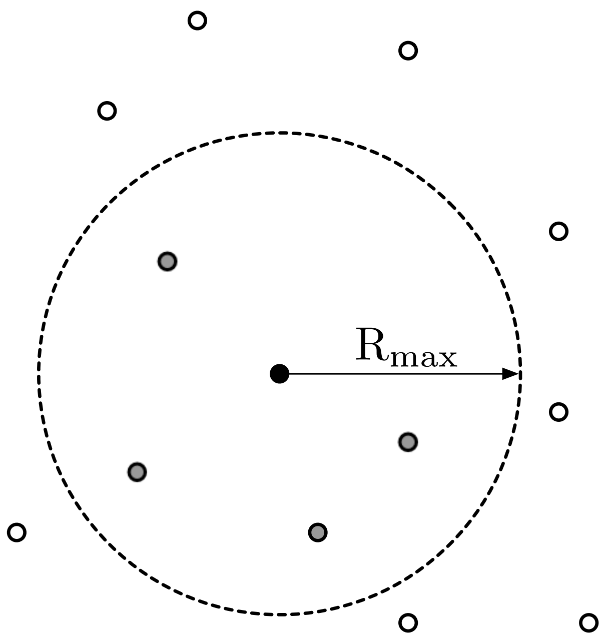


Figure B.2.1 Agents are modeled as points. Each agent is capable of sensing other agents within a radius of R_{max} (*grey*), centered at its current position. By sensing, we mean an agent is able to infer the *direction* to another agent.

4. The mutual visibility graph is initially connected; agents that are not part of this connected graph are ignored.

We propose an iterative algorithm consisting of the following two phases:

1. **Sense**. Agents hold position and survey its environment for other agents within its vicinity.
2. **Maneuver**. Agents move from its current position to a waypoint along a computed trajectory and hold position.

In this stop-and-go maneuver, agents are not able to continuously sense its environment. We will also assume that all agents complete a phase prior to any one agent starting on the next. This inherently requires some form of synchronization among the agents, however other similar stop-and-go rendezvous methods have been shown to work asynchronously provided some constraints [4].

The key to our algorithm is the waypoint selection strategy. We are interested in a strategy that guarantees the connectivity of the mutual visibility graph, i.e. once a graph $\mathcal{G}$ is connected, this graph should remain connected for all time. How this is achieved will be described in the following. Implementation details, such as how the maneuvers are carried out and what happens when agents collide is left for future work.

The selection of a waypoint is divided into two parts: (1) waypoint direction and (2) waypoint distance. The algorithm is summarized in Algorithm 1.

Algorithm 1 Computes the direction and distance to the next waypoint with respect to agent i.

```
// Global Variables
$d$: Degree of $i$
$\mathcal{N}_i$: Set of agents visible to $i$
$\ell_j$: Maximum permissible travel distance to the waypoint with respect to $j$
$\mathrm{R_{max}}$: Visibility radius

// Input
$\hat{\mathcal{X}}$: Set of direction vectors to all visible agents $j \in \mathcal{N}_i$

function ComputeWaypoint($\hat{\mathcal{X}}$)
    $\hat{\mathbf{p}} \leftarrow (1/d) \sum_{j \in \mathcal{N}_i} \hat{\mathbf{x}}_j$

    for $j \in \mathcal{N}_i$ do
        $\theta_j \leftarrow$ Angle made by $\hat{\mathbf{p}}$ and the line joining $i$ and $j$
        if $|\theta_j| \geq \pi/2$ then
            BREAK
        end if
    end for

    for $j \in \mathcal{N}_i$ do
        if $\theta_j \leq \pi/3$ then
            $\ell_j \leftarrow \mathrm{R_{max}}/2$
        else
            $\ell_j \leftarrow (\mathrm{R_{max}}) \cos \theta_j$
        end if
    end for

    $|\mathbf{p}_i| \leftarrow \min_{j \in \mathcal{N}_i} \ell_j$
    return p
end function
```

Waypoint Direction

Let $\hat{\mathbf{x}}_i$ denote the direction (obtained through heading measurements) to agent i. A waypoint's heading $\hat{\mathbf{p}}_i$ is defined as the mean (or maximum likelihood) heading to all visible agents $j \in \mathcal{N}_i$,

$$\hat{\mathbf{p}}_i = \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \hat{\mathbf{x}}_j, \tag{B.2.1}$$

and d_i is the degree of node i (or the size of the set $\mathcal{N}_i$). Intuitively, an agent should move in the direction that it thinks is most likely to harbour the majority of agents. However, we impose a constraint stating that agents at the periphery of the visibility graph have priority of moving first. This is to ensure that agents closer to the centre of the visibility graph do not move out of range. The graph would otherwise become disconnected. This can be stated mathematically as follows:

$$|\theta_j| < \frac{\pi}{2} \quad \forall j \in \mathcal{N}_i, \tag{B.2.2}$$

where θ_j is the angle made by the direction vector $\hat{\mathbf{p}}_i$, and the line joining i and j, see Figure B.2.2. Agents which satisfy this equation are allowed to move, while the others remain stationary for the current iteration of the algorithm.

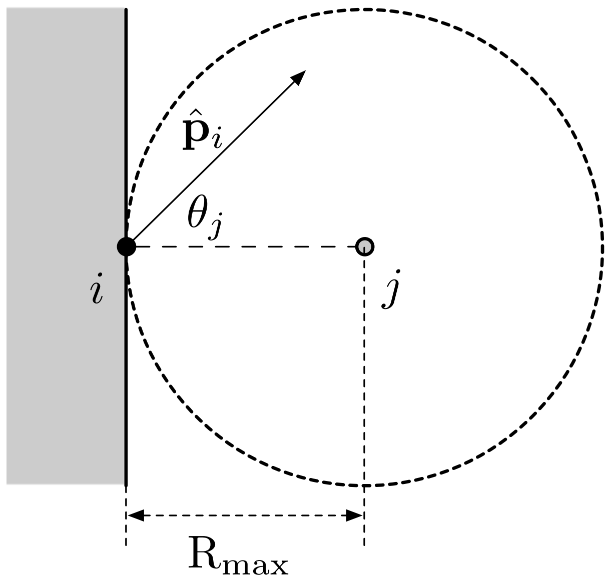


Figure B.2.2 Assume that agent i has detected j, which is positioned at a distance $\mathrm{R}_{\max}$ from i. In this scenario, if i moves in any direction away from j then the connectivity between i and j is lost. The allowed direction of travel for agent i with respect to j is bounded by a line passing through i that is tangent to a disk of radius $\mathrm{R}_{\max}$ centred at j. The allowed direction of travel in this case is to the right of the grey area in the figure.

Waypoint Distance

For agents to remain visible to one another during their maneuvering period, they should restrict their motion by only moving within a closed disk of diameter $\mathrm{R}_{\max}$ centered at a point that is equidistant from their initial positions. However, an agent's position is not known and cannot be estimated. What is known is that if an agent senses another agent, that other agent must be within its sensing radius of $\mathrm{R}_{\max}$. There exists two extreme cases. On one hand, a pair of agents may be infinitely close together, their motion is thus confined to a circle of diameter $\mathrm{R}_{\max}$ centered at the two agents. On the other hand, a pair of agents may be at a maximum distance of $\mathrm{R_{max}}$ apart, confining their motion to a circle of diameter $\mathrm{R_{max}}$ centred at a point equidistant from the two agents. The maximum distance in which an agent can travel is thus confined to the intersection of these two circles, see Figure B.2.3. The maximum distance ℓ_j that i can move towards its waypoint without leaving the vicinity of $j \in \mathcal{N}_i$ is given by

$$\ell_j = \begin{cases} \dfrac{\mathrm{R_{max}}}{2} & \text{if } \theta_j \leq \dfrac{\pi}{3}, \\ \mathrm{R_{max}} \cdot \cos\theta_j & \text{otherwise.} \end{cases} \tag{B.2.3}$$

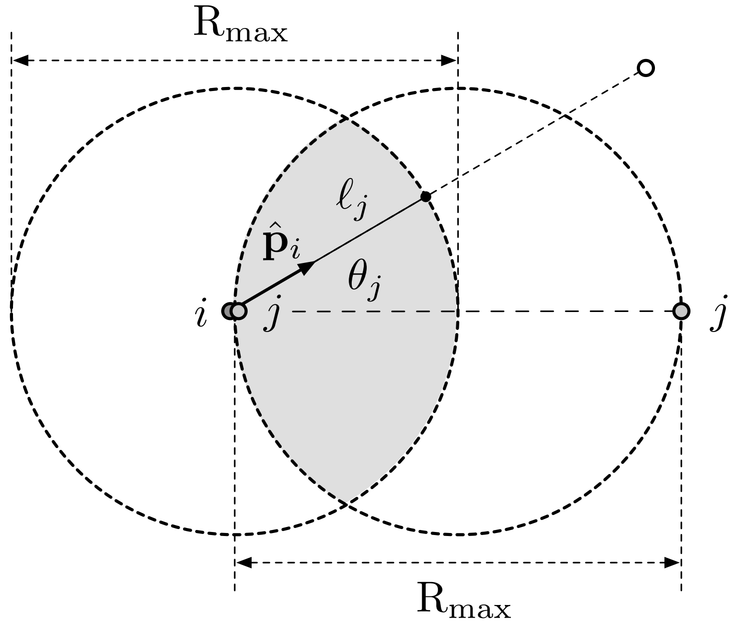


Figure B.2.3 The maximum distance that i can move in the direction $\hat{\mathbf{p}}_i$ without leaving the vicinity of $j \in \mathcal{N}_i$ is given by ℓ_j, which is the distance a line, starting at i, makes with the intersection of the two circles shown in the figure. Each circle represents the area of two extreme cases in which a pair of agents i and j may move in one iteration without leaving each other's sight: the circle on the left respresents the case where the agents are infinitely close together, and the circle on the right respresents the case where the agents are at a maximum distance of $\mathrm{R_{max}}$ apart.

The maximum distance ℓ_j can be computed for all visible agent $j \in \mathcal{N}_i$, and the minimum is chosen as the waypoint distance for agent i,

$$|\mathbf{p}_i| = \min_{j \in \mathcal{N}_i} \ell_j. \tag{B.2.4}$$

References

[1] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, "Distributed memoryless point convergence algorithm for mobile robots with limited visibility," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 818–828, 1999.

[2] Z. Lin, B. Francis, and M. Maggiore, "Getting mobile autonomous robots to rendezvous," *Control of Uncertain Systems: Modelling, Approximation, and Design*, pp. 119–137, 2006.

[3] J. Lin, A. S. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem. Part 1: The synchronous case," *SIAM Journal on Control and Optimization*, vol. 46, no. 6, pp. 2096–2119, 2007.

[4] ——, "The multi-agent rendezvous problem. Part 2: The asynchronous case," *SIAM Journal on Control and Optimization*, vol. 46, no. 6, pp. 2120–2147, 2007.

[5] S. Martınez, J. Cortes, and F. Bullo, "On robust rendezvous for mobile autonomous agents," in *International Federation of Automatic Control World Congress*, 2005.

List of Outreach Activities

The Distributed Flight Array and its research results have been disseminated to the general public through live lab demonstrations, presentations, and high-profile media outlets.

Lab Demonstrations

From 2009 to 2013, the Distributed Flight Array was shown in more than 100 lab demonstrations to guests from academia, schools, media, government, and industry.

Talks

- Festival Robotique, Lausanne, Switzerland, Apr. 2013
- Technikwoche Innerschweiz, Schwyz, Switzerland, Sept. 2012
- Technikwoche St. Gallen, Switzerland, Mar. 2012
- Zürcher Sport – Ferienlager Fiesch, Fiesch, Switzerland, Oct. 2011
- ETH Zurich Maturandentage, Zurich, Switzerland, Sept. 2011.
- Swiss Science Center Technorama, General Assembly of Schweizerische Gesellschaft Pro Technorama, Winterthur, Switzerland, May 2011

Selected Media Coverage

- "Robot Self-Assembles, Then Flies: Self-sufficient wheeled robots lock together to make a larger, flying robot", MIT Technology Review, Oct. 2010
- "Robots Podcast: Distributed Flight Array", IEEE Spectrum, Aug. 2010
- "The mini flying robot drones that join forces before takeoff - all without human help", Associated Newspapers Ltd., June 2010
- "Flying robots self-assemble into midair swarm", CNET News, June 2010
- "Video: Tiny Autonomous 'Copters Combine Voltron-Style To Create a Larger, Stronger Aircraft", Popular Science Online, June 2010

- “Robots: Distributed Flight Array", Robots.net, June 2010
- “The Self-Assembling UAV", Aviation Week, June 2010
- “The Distributed Flight Array," Robots Podcast, Aug. 2010
- “Daily Planet," Discovery Channel, Canada, Jan. 2011
- “Technical Progress and Future Technologies," Russian Television Company TV 3, July 2010
- “Die Intelligenz der Roboter," NZZ Format (SF1), Apr. 2010

List of Supervised Student Projects

Below is a complete list of student projects that were supervised at the Institute for Dynamic Systems and Control at ETH Zurich as part of the author's doctoral studies.

Master Thesis
(*Six months, full-time research project*)

- Miguel Picallo Cruz, *Parameterized Control Methodology for a Modular Flying Robot*, 2012
- Jürg Weber, *Towards a Modular Battery Charging Station*, 2011
- Maximilian Kriegleder, *State Estimation of the Distributed Flight Array*, 2010 – 2011
- Alireza Ramezani, *The Distributed Flight Array*, 2009

Semester Project
(*Semester-long, part-time research project*)

- Andreas Waeber, *Driving Kinematics for the Distributed Flight Array*, 2013
- Thomas Lindeberg, *Measuring the Distributed Flight Array Modules with a New Testing Platform*, 2013
- Clemens Fischer, *Outlier Detection*, 2012
- Corsin Gwerder, *Reconfiguring Distributed Flight Array Topology*, 2011
- Simon Dössegger, *A Model-Based Procedure to Determine the Flight Control Parameters of the Distributed Flight Array*, 2011
- Andreas Lauber, *Flight Motor Test-Bench*, 2010
- Daniel Eberli, *Drive Simulator for the Distributed Flight Array*, 2010
- Luzius Brodbeck, *Drive Simulator for the Distributed Flight Array*, 2009

Bachelor Thesis
(*Semester-long, full-time research project*)

- Christof Dubs, *Design and Implementation of a Flipping Mechanism for a DFA Module*, 2011
- Aaron Coulin, *Implementation of a Digital 3-Axis Compass for Absolute Orientation*, 2011
- Raphael Wüest, *Solar Cell Power and Sensing*, 2009

Studies on Mechatronics
(*Semester-long, part-time literature study*)

- Andreas Waeber, *A Literature Review on Rendezvous Algorithms*, 2013
- Aaron Coulin, *Literature Review of Altitude and Attitude Sensors*, 2011
- Christoph Kammer, *Wireless Power Supply for the Distributed Flight Array*, 2010
- Raphael Shottenhaml, *Inter-module Communication for the Distributed Flight Array*, 2010

!And Yet it Moves
(*Two semester-long classes*)
Francesco Crivelli, Edward Ho, Claudio Marforio, Caterina Vitadello, Ville Widgrén, Martin Widmer, Filip Wieladek, and Daniel Wolfertshofer, *The Distributed Flight Array*, 2008 – 2009

Hilfsassistent

Mahdi Asadpour, *Software Development*, 2010 – 2011

Summer Internship

Sneh Vaswani, *Hardware Development*, 2010