Single Chip Solution for Stabilization Control & Monocular Visual Servoing of Small-Scale Quadrotor Helicopter

by

Mohammed Raju Hossain

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Faculty of Engineering & Applied Science

Memorial University of Newfoundland

May 2017

St. John's Newfoundland

Abstract

This thesis documents the research undertaken to develop a high-performing design of a small-scale quadrotor (four-rotor) helicopter capable of delivering the speed and robustness required for agile motion while also featuring an autonomous visual servoing capability within the size, weight, and power (SWaP) constraint package. The state of the art research was reviewed, and the areas in the existing design methodologies that can potentially be improved were identified, which included development of a comprehensive dynamics model of quadrotor, design and construction of a performance optimized prototype vehicle, high-performance actuator design, design of a robust attitude stabilization controller, and a single chip solution for autonomous vision based position control. The gaps in the current art of designing each component were addressed individually. The outcomes of the corresponding development activities include a high-fidelity dynamics and control model of the vehicle. The model was developed using multi-body bond graph modeling approach to incorporate the dynamic interactions between the frame body and propulsion system. Using an algorithmic size, payload capacity, and flight endurance optimization approach, a quadrotor prototype was designed and constructed. In order to conform to the optimized geometric and performance parameters, the frame of the prototype was constructed using printed circuit board (PCB) technology and processing power was integrated using a single chip field programmable gate array (FPGA) technology. Furthermore, to actuate the quadrotor at a high update rate while also improving the power efficiency of the actuation system, a ground up FPGA based brushless direct current (BLDC) motor driver was designed using a low-loss commutation scheme and hall effect sensors. A proportional-integral-derivative (PID) technology based closed loop motor speed controller was also implemented in the same FPGA hardware for precise speed control of the motors. In addition, a novel control law was formulated for robust attitude stabilization by adopting a cascaded architecture of active disturbance rejection control (ADRC) technology and PID control technology. Using the same single FPGA chip to drive an on-board downward looking camera, a monocular visual servoing solution was developed to integrate an autonomous position control feature with the quadrotor. Accordingly, a numerically simple relative position estimation technique was implemented in FPGA hardware that relies on a passive landmark/target for 3-D position estimation.

The functionality and effectiveness of the synthesized design were evaluated by performance benchmarking experiments conducted on each individual component as well as on the complete system constructed from these components. It was observed that the proposed small-scale quadrotor, even though just 43 cm in diameter, can lift 434 gm of payload while operating for 18 min. Among the ground up designed components, the FPGA based motor driver demonstrated a maximum of 4% improvement in the power consumption and at the same time can handle a command update at a rate of 16 kHz. The cascaded attitude stabilization controller can asymptotically stabilize the vehicle within 426 ms of the command update. Robust control performance under stochastic wind gusts is also observed from the stabilization controller. Finally, the single chip FPGA based monocular visual servoing solution can estimate pose information at the camera rate of 37 fps and accordingly the quadrotor can autonomously climb/descend and/or hover over a passive target.

Acknowledgments

I would like to express my deepest gratitude to my supervising professor Dr. Nicholas Krouglicof who throughout this work offered his guidance and conveyed a spirit of adventure in regard to this research. Without his persistent help, relentless encouragement and versatile expertise this work would not have been possible. It was an absolute privilege to have him as my mentor. I am extremely thankful for all he has done for me throughout my graduate studies. To him, I wish to say “You are a wonderful supervisor, professor, researcher, and a great man!”

I would like to extend my thanks to the co-supervisor and doctoral supervisory committee member, Dr. Ralf Bachmayer and Dr. Geoff Rideout respectively for their valuable feedback and expert opinions in developing the key ideas of this thesis. I also thank Dr. Cheng Li and Dr. Syed Imtiaz for their service in the committee for doctoral comprehensive examinations.

Special thanks to the brilliant team of technologists at the Faculty of Engineering & Applied Science, Memorial University of Newfoundland (MUN) for the skillful services, and in particular Messrs. Brian Pretty, Tom Pike, and Steve Steele for their help in implementing the design ideas of this thesis.

I am forever grateful to Dr. Kaaren May and Mr. Stephen Reddin, past project managers of the Research Grant and Contract Services (RCGS) at MUN. They have borne all the logistical requirements associated with my research work, which has enabled me to concentrate on my research. Additional thanks to them for helping me with writing technical reports as well as proofreading them for me.

I gratefully acknowledge the financial support contributed by the Research and Development Corporation of Newfoundland (RDC), Atlantic Canada Opportunities Agency (ACOA), Memorial University of Newfoundland, and the Boeing Company. I am particularly grateful to RDC for awarding me with scholarship funding through the Ocean Industries Student Research Award (OISRA) program.

Thanks to my peers Messrs. Taufiqur Rahman, Michael Morgan, Dion Hicks, Dennis Fifield, Migara Lianage, and Teng Wang for their thoughtful words, friendship, and for making my life fun while working.

I would like to extend my sincerest appreciation to my mother, Mrs. Rashida Akther, and to my father, Mr. Nurul Anwar for their inspiration and motivation to always do my best in everything I do, for their faith in me, and for allowing me to be as ambitious as I wanted. Without their support I would never have been able to undertake this doctoral thesis.

Most importantly, I would like to thank my wife Mrs. Fathima Naheed. Her support, encouragement, quiet patience, and unwavering love were undeniably the bedrock upon which the past several years of my life have been built. Her simple words 'you can do it!' really mean a lot to me.

Lastly, I dedicate this thesis to my lovely child Rafaz Anwar who is the pride and joy of my life. I love you more than anything and I appreciate your patience for not having sufficient attention from me during this thesis. And thanks for reminding me every morning with your growing speaking skills: "Papa, go to office and write your thesis".

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ABS	Acrylonitrile Butadiene Styrene
ADRC	Active Disturbance Rejection Control
AHRS	Attitude & Heading Reference Sensor
BLDC	Brushless Direct Current
DAQ	Data Acquisition
DC	Direct Current
DoF	Degree of Freedom
EJS	Eulerian Junction Structure
EMF	Electromotive Force
ESC	Electronic Speed Controller
ESO	Extended State Observer
FoM	Figure of Merit
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GCS	Ground Control Station
GPS	Global Positioning System
I/O	Input-Output
IC	Integrated Circuit
IMU	Inertial Measurement Unit

IR	Infra-red
IR-LED	Infra-red Light Emitting Diode
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
LUT	Look-up Table
MEMS	Micro-Electro-Mechanical System
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
PCB	Printed Circuit Board
PLL	Phase Locked Loop
PWM	Pulse Width Modulation
RC	Radio Controlled
RF	Radio Frequency
RTL	Register Transfer Level
SISO	Single Input Single Output
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SWaP	Size, Weight, and Power
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
VGA	Video Graphics Adapter
VHDL	VHSIC Hardware Description Language
VTOL	Vertical Take-off and Landing
ZOH	Zero Order Hold

Chapter 1

Introduction

In recent years, Unmanned Aerial Vehicles (UAVs) have become an important tool in military and civilian applications for their potential uses in both indoor and outdoor environments. This rapid adoption of UAVs can be attributed to a variety of factors. The idea of a flight without an onboard human pilot is indeed an attractive one, which greatly reduces size, cost, and, in some sense, complexity to support essential functions to accommodate people. Equally important is the ability of UAVs to prevent human beings from flying into hazardous or hostile environments [1]. Lastly, UAVs are capable of performing various complicated maneuvers which elude even the most talented human pilots. Therefore, UAVs are considered as the best tool for *dull*, *dirty* and *dangerous* surveillance and inspection missions, when compared to manned aircrafts.

Among the two main types of UAVs, fixed-wing and rotor-crafts, rotor-crafts have the important capability of hovering. Subtypes are helicopters and multi-rotors; multi-rotors are preferred for the robustness and modularity of the fuselage, being less subject to damage and easier to repair. Furthermore, quadrotors are the most widespread and least costly multi-rotors. Quadrotors have two pairs of counter-rotating propellers located at the vertices of a square. The rotational speeds of the motors are varied to obtain the desired flight pattern.

Within the last few decades, the revolutionary advancement of semiconductor and computing technologies evolved the quadrotors into smaller and more efficient aircraft. In addition to that, the systems have become more autonomous and more aware of the environment than ever. Recently quadrotors have been envisioned as the *delivery man* of the future [2]. Of course, their potential uses are virtually limitless and extend far beyond the simple application of delivering packages. A few possible uses include autonomous border surveillance [3], aerial photography & videography, bridge & structure inspection [4], and swarm mobile sensor networks [5].

However, to this date, development of a fully autonomous quadrotor capable of performing the activities in an outdoor environment is still an open problem for researchers. The major functions to design such a quadrotor include reliable sensing, fast actuation, robust stabilization, and autonomous position control. The development process also includes number of engineering challenges such as construction of a platform that can hold all the avionics sensors while demonstrating agile maneuvers, development of a high performance actuation system capable of generating agile motion, formulation of a low level stabilization control system, and a high level autonomous position control feature. Performing the aforementioned tasks encompasses both analysis and synthesis of UAV motion control laws and the associated hardware implementation. This requires expert engineering knowledge from several disciplines including electronics, system integration, control engineering, and software development. Correspondingly, this thesis focuses on the theoretical and practical aspects of designing and implementing a prototype of an agile quadrotor platform based on field programmable gate array (FPGA) hardware that can be employed in autonomous surveillance activities in a relatively harsh outdoor condition.

1.1 Motivation

General requirements to achieve agile motion of a quadrotor include: a vehicle frame equipped with all the necessary avionics yet capable of demonstrating highly dynamics maneuvers, a fast and energy efficient actuation module to exploit the agility of the vehicle at low power cost, a robust control module that can demonstrate reliable performance even under external disturbance, and finally an appropriate position control system to realize autonomous localization. Typically it is difficult to adhere to these specifications when a quadrotor is constructed using off-the-shelf components, because off-the-shelf vehicle frames, actuators, sensors and control hardware that are designed for general purpose use cannot address application-specific requirements of speed, size, and compatibility. Thus this thesis is principally motivated by the need for a design approach that addresses speed, reliability, and SWaP (size, weight, and power) requirements of an agile autonomous quadrotor from the ground up.

Many researchers have studied the structural synthesis of an agile quadrotor [6, 7] and have concluded that size and weight reduction can result in increased agility of the vehicle. However, reducing the size shrinks the available real-estate for integration of multiple sensors and powerful processing units leading to a challenge of practical implementation. Correspondingly, this thesis employs published architectures for designing small-scale quadrotors that suggest a printed circuit board (PCB) as the vehicle frame (see Section 2.2). However, the existing art involves multiple PCBs vertically stacked to accommodate various processing units, which increases the size and hinders the agility. As a solution to this vicious cycle, this thesis proposes a design methodology to construct a small-scale quadrotor vehicle frame that is optimized for payload capacity and flight endurance. Accordingly, PCB technology is proposed to construct the small-scale vehicle frame, and single chip processing based on FPGA technology is chosen to be integrated with the PCB.

Along with the vehicle frame design, a high performance motor-propeller assembly is needed to add agile actuation capacity to the platform. Many actuation (i.e., motor) control technologies for the quadrotor have been proposed in literature (see Section 2.4.1). Most of these controllers are based on an off-the-shelf electronic speed controller (ESC) that has a limited command update rate (typically < 400 Hz) making it less attractive for a high performance quadrotor design. Moreover, the off-the-shelf ESCs do not allow access to their motor commutation algorithms and do not offer closed loop speed control. In recognition of these limitations while appreciating the capacity of FPGA hardware, which can offer parallel computation, superior speed, greater reliability, and power efficiency [8], this thesis proposes a ground up development and implementation of a brushless direct current (BLDC) motor control algorithm implementable on FPGA hardware.

As an under-actuated aerial platform, a quadrotor is only capable of a stable hover at its equilibrium. For the same reason, its position and attitude cannot be controlled simultaneously [9]. Moreover, when the platform is agile, developing a stabilization control algorithm for the vehicle becomes a challenging task. Many stabilization control algorithms for the quadrotor have been reported in the literature (see Section 2.4). However, a stabilization control algorithm that is robust in both a nominal hover and a hover in the presence of an external disturbance is rare. Motivated by this gap in the existing studies, this thesis proposes a robust stabilization control algorithm based on active disturbance rejection control (ADRC) [10] technology and a cascaded control architecture [11]. Unlike many modern controllers, the ADRC can deliver robust control performance without the requirement of a model of the system, which allows this control technique to be particularly suitable for embedded applications.

A quadrotor is considered autonomous when it can feature position control along with its stabilization control. In general, the orientation measurement feedback for stabilization is acquired from inertial sensors (composed of gyros, accelerometers and magnetometers) while outdoor position and velocity measurements are obtained from a global positioning system (GPS). The main drawback of the common GPS receivers used in UAVs is the fact that position measurements may be inaccurate by up to a few meters. Therefore, the GPS solution for navigation is not applicable in the case where precision position control is required. On the other hand, using vision as a measuring system has multiple advantages over employing other sensors. For example, cameras are passive and in general are also light and cheap. Furthermore, they can be used for both indoor and outdoor applications. Given the size and weight restrictions faced by a small-scale agile quadrotor, this thesis proposes a ground up development of a camera sensor populated on the same PCB that acts as the vehicle frame. For the development of the camera driver and to implement the vision based position control (visual servoing), the same FPGA hardware is used that holds the rest of the digital circuits discussed above. This results in a single chip visual servoing solution suitable for integration in a small-scale quadrotor. Different visual servoing techniques have been reported in literature (see Section 2.5) ranging from a monocular approach to a stereoscopic approach using both a passive target and active landmarks. However, most of the visual servoing algorithms are computationally expensive and are not suitable for single FPGA implementation. Moreover, implementing a control application on FPGA hardware in an exercise in designing digital circuits has yet to be adopted widely by the practicing control engineers, who are accustomed to developing control applications for conventional control platforms in the form of software codes. Correspondingly, this thesis proposes a numerically simple monocular visual servoing technique complemented with a discrete Kalman Filter based motion sensing algorithm to develop a feedback controller for position control of the platform.

1.2 Statement of Co-Authorship

The thesis has been undertaken under the framework of the Intelligent Sensor Platform for Remotely Piloted Vehicles (INSPIRUS) project at Memorial University of Newfoundland. The focus of the project is to develop functions for autonomous operation of UAVs, and includes both ground and aerial robots. The author has collaborated with other researchers of the project to accomplish the goals of this thesis. Research outcomes have been published in peer-reviewed full length conference papers. Among the publications, this thesis includes content from [12, 13, 14, 15], which were principally written by the thesis author. In addition, permission for reusing each of these articles in an academic dissertation was obtained from the appropriate copyright owners.

The contributions of the thesis in these publications include a literature survey, development of the central concept, development of a mathematical framework, designing of experiments, data analysis, and interpretation of results. The co-authors have contributed by providing expert critiques on the research approach and the theoretical foundation. Additionally, they have assisted in construction of test-bench, executing experiments, and have implemented designs devised by the thesis author. With the exception of Section 3.3.5, Section 3.3.4.1 , Section 3.3.6, Section 3.3.10, and Section 7.4.1, the author claims total intellectual ownership of the engineering designs, the formulations, and the analyses presented in this thesis. The design ideas of motor driver circuitry presented in Section 3.3.4.1 were primarily conceived by Dr. Nicholas Krouglicof, the principal investigator of the INSPIRUS project. The iPad interface of the ground control station (GCS) presented in Section 3.3.10, the implementation of contour tracking in FPGA hardware for visual servoing presented in Section 7.4.1, and the development of the FPGA hardware itself were jointly developed by various project members of the INSPIRUS project. Therefore, the thesis author, along with other project personnel contributed towards implementation and revision of the aforementioned design ideas.

1.3 Contributions of the Thesis

The contributions of the thesis can be summarized for both theoretical and practical aspects of quadrotor aerial robots, which are elaborated in the following list:

Design and prototyping of a performance optimized small-scale quadrotor: In order to develop a prototype quadrotor that can offer a good balance between size, payload capacity, and flight endurance an optimized design and construction methodology is presented in this study. The novelty of the approach lies in the monolithic integration of vehicle frame and processing power that can conform to the optimized size and performance parameters. Accordingly, the frame of the vehicle is constructed based on PCB technology which can keep the size and weight of the vehicle within the design specifications and also offer increased payload capacity while retaining agility of the vehicle. To meet the SWaP restriction of the small-scale quadrotor, single chip FPGA based processing hardware is chosen to integrate on the same PCB frame. The result is a size, weight and power efficient small-scale agile quadrotor capable of carrying a high payload while also offering longer flight time. The performance improvement is demonstrated through comparative study conducted with respect to commercial platforms of similar form-factor.

Development of bond-graph based multibody dynamics modeling and control law for aerial vehicles with a case study focused on quadrotor UAV: Typical modeling approaches formulate time derivative equations of platform dynamics based on the assumption that the platform is a single rigid body with its body axis aligned to the center of gravity of the platform. However, the formulation obtained in that approach has fewer provisions for modification and model expansion. The novelty of the proposed modeling technique originates from the considerations of the dynamics of each component of the vehicle that has its own dynamics, and the inclusion of interaction between components using mechanical joints. Unlike a single rigid body model, which ignores the effect of multi-body interactions that occur in a real system, the proposed comprehensive multi-body model can reproduce actual dynamic behaviour of the respective system. Moreover, the proposed graphical modular modeling approach mitigates obsolescence of the model. The performance and the accuracy of the model are validated via simulation study conducted by taking a quadrotor as a case example.

Design, development, and implementation of FPGA based power efficient BLDC motor controller for aerial vehicles: Because of limitations in command update rate, power efficiency, and the absence of closed loop speed control in the off-the-shelf motor controllers, they are not suitable as the actuation systems of agile quadrotors. Alternatively, utilizing the parallelism of the FPGA technology, a ground up BLDC motor driver and speed controller are developed in this thesis by employing a trapezoidal back-EMF (electromotive force) profile and a power efficient commutation control algorithm. The performance of both commutation and closed loop PID velocity control was experimentally validated using a corresponding implementation in FPGA hardware.

Formulation of a cascaded robust stabilization control law for quadrotor and its implementation in FPGA hardware: This thesis proposes a novel cascaded attitude stabilization control law using PID as the inner loop high dynamics angular rate controller and ADRC as the outer loop disturbance compensated orientation controller. This is in contrast with the conventional stabilization controller that heavily relies on a system model making it less suitable for systems that typically encounter perturbation of dynamic parameters due to external disturbances (i.e., wind gusts). In addition, the model based controller involves complex numerical computation, making it impractical for realization in real-time parallel processing hardware, which is a key computational component in the space constrained agile platform, as discussed above. A series of performance characteristics experiments were performed to evaluate the stabilization performance both in nominal hover and hover in gusty wind conditions.

Development and implementation of a single chip visual servoing solution for a small-scale quadrotor capable of measuring 3-D position relative to a static ground landmark during flight using a single downward looking camera and a passive landmark: Employing the projection from an infra-red (IR) landmark and processing vision using a pinhole camera model, a computationally efficient position estimation algorithm is proposed and deployed in the FPGA hardware that can compensate for vehicle angular motion for accurate localization in real-time. Furthermore, vision processing data is fused with an inertial sensor for velocity estimation of the vehicle. Accordingly, a feedback position control system is developed for autonomous visual servoing. The uniqueness of the proposed technique lies in the monolithic integration of the image processing, vision processing, motion processing, and control law in a single chip FPGA processor that uses a monocular perspective view from a quadrotor to perform autonomous maneuvers.

1.4 Organization of the Thesis

The following is a brief outline of the remainder of the thesis.

Chapter 2 briefly reviews the existing literature relevant to the engineering challenges associated with each contribution discussed in the previous section.

Chapter 3 details the design and construction of the performance optimized agile quadrotor that will be employed in the rest of the thesis as the test-bench. Comparative analysis of the optimized performance between the proposed prototype and off-the-shelf quadrotors are also presented in this chapter.

Chapter 4 constructs the appropriate dynamic models of the quadrotor and its actuation system to conduct simulation studies.

Chapter 5 details the development and implementation procedure of the FPGA based BLDC motor driver/controller. Experimental results are presented along with the performance evaluation of the driver and the controller.

Chapter 6 formulates the ADRC-PID cascade attitude stabilization control law. In addition, the FPGA implementation of the stabilization system is described here. Experimental evaluation of the controller is presented with performance analysis and the interpretation of the controller response.

Chapter 7 documents the procedure of the monocular visual servoing technique for a SWaP constraint quadrotor and achieves the complete system in a single FPGA chip through development of the appropriate digital circuits. It also reports the validation of the system's functionality and performance through experimental results.

Chapter 8 offers the concluding remarks along with a discussion on limitations of the study and possible scope for future research.

Chapter 2

Literature Review

2.1 Introduction

Unmanned vehicles, including aerial vehicles, offer new perspectives for transportation and services. Although the legal requirements are still quite restrictive [16], UAV applications, and in particular quadrotor usage, are becoming widespread, from military usage to civil applications, such as inspection tasks [4] and aerial imaging [17]. Since the interest of this thesis is to automate broadly-used UAVs, the quadrotor platform is chosen as the preferred vehicle for this study.

Recent technological advances in sensors, actuators, batteries, and processing modules that allow installing on quadrotors all components necessary for autonomous flights at a reasonable cost also constitutes a favourable factor. However, the development of these applications raises several challenges. First, from a mechanical point of view, the vehicle must have good flying capabilities (i.e., agility and maneuverability) and provide enough payload for embarked sensors with a long flight time to reliably conduct outdoor missions. While the specific limits on size, weight and cost are, of course, arbitrary to an extent, the motivation is to develop a low cost platform that needs to be agile in performance and has an optimum capacity of payload loading and flight endurance. A survey of agile quadrotor design techniques is presented in Section 2.2.

A good understanding of quadrotor maneuvers, associated aerodynamic effects, and fine modeling of these effects is crucial to design control laws for the vehicle to optimize the maneuverability and energetic efficiency of the system. Accordingly, a survey of existing accurate numerical modeling techniques is conducted in Section 2.3. The simulation model is also useful to evaluate the performance, robustness, and limitations of the controller.

Since the FPGA hardware is chosen as the real-time processing unit for the platform, it is more efficient to use the same FPGA hardware as the motor driver and motor speed controller instead of using a separate module available for a general purpose. The approach not only saves real-estate and the cost of adding new components but also reduces the weight required by the off-the-shelf module. Moreover, the custom actuator design approach increases the flexibility of implementing a power efficient motor commutation and control technique, which is required to achieve long flight endurance, as the motors are the primary power consumer of a quadrotor. A review of the state-of-the-art BLDC motor driving techniques using FPGA technology is reported in Section 2.4.1.

Relying on information provided by the sensing unit, effective control can help in overcoming the limitations of inexpensive sensors used to design a cost-effective platform. In general, the quadrotor control task requires two levels: low level flight stabilization control (i.e., attitude control) and high level flight translation control. For low level flight stabilization control of the platform with reasonable robustness to aerodynamic perturbations (such as wind gusts), a feedback control law must posses fast response and the capability of disturbance compensation. Accordingly, Section

2.4.2 provides a survey of existing control techniques for low-level stabilization control of a quadrotor, ranging from simple linear control to complex nonlinear controllers, essentially to identify the strengths and weaknesses of the control techniques, keeping in mind that the prototyping platform used in this thesis is FPGA based hardware.

Finally to integrate a real-time high level control feature such as vision based position control in a small-scale quadrotor, appropriate sensor technology and a vision processing algorithm need to be developed. Besides the size, weight, and power restriction of the platform, an appropriate computational hardware has to be accommodated for the development of the visual servoing solution. Since in a non-instrumented environment no single sensor can produce direct measurement of pose, a combination of sensors may have to be used to accomplish this task. Accordingly, Section 2.5 presents an extensive vision based position estimation technique and development of high level control and planning methods for addressing sensing, vision and control challenges associated with a small-scale quadrotor.

This chapter provides both the comprehensive background research and the review of related works targeting each of the challenges that have been addressed in this study. The objective is to bestow the reader with the needed information to analyze the rest of the document.

2.2 Design of Performance Optimized Quadrotor

A conventional quadrotor design has a structure which includes a body that is formed with a simple composite material or wood so that the body can be independently built from electric and electron structures that control the electronic devices for a flight. The weight increase causes the main board, the battery and the propeller to be enlarged, and accordingly the structure of the flying vehicle becomes complicated, difficult to assemble, and increase the production cost of the flying vehicle. Subsequently, the increase of the weight of the flying vehicle decreases the efficiency of flight [18]. In order to address the challenges associated with size, weight, and agility of the quadrotor, existing research has reported various conclusive remarks. It was experimentally shown by [7] that agile performance of a quadrotor can be achieved by reducing the size of the vehicle platform to attain greater acceleration that also enables rapid response to disturbances resulting in greater platform stability. There are two commonly accepted approaches that study scaling in aerial vehicles [19]. Froude scaling suggests that the linear acceleration of the quadrotor is independent of the characteristic length of the vehicle while the angular acceleration $\alpha \sim L^{-1}$. On the other hand, Mach scaling leads to the conclusion that linear acceleration $a \sim L$ while angular acceleration of the quadrotor $\alpha \sim L^{-2}$. In [6], authors have experimentally demonstrated the result of the scaling criterion and proved that smaller quadrotors are more agile. However, a small platform size to allow agile maneuver can result in limited payload capacity to carry useful onboard sensing modules and does not offer high flight endurance. These three features are in contradiction with each other, making platform selection difficult.

Table 2.1 shows a list of the current, most commonly used hovering platforms below 80 cm diameter that are commercially available [20]. Most of these platforms have a payload capacity of 50 gm to 250 gm, with the exception of the Pelican which has almost 500 gm payload capacity. However, the platform size is too large to demonstrate agile performance. It is evident from the chart that as the size decreases, the payload capacity decreases with it. This is largely due to its own structural weight that compromises the payload capability. On the other hand, most of the platforms have around 12 to 20 min maximum flight endurance. However, there is no visible trend of flight time observed that varies with the size.

Table 2.1: Review of commercially available multi-rotor UAVs less than 80 cm, sorted by platform size (in diameter)

Platform Name	Diameter	Flight Time Max.	Take-off Weight Max.	Payload Max.	Payload Loading	Battery Loading
Hornett	29 cm	15 min	350 gm	50 gm	14%	38%
Coax	34 cm	20 min	320 gm	70 gm	22%	27%
Hummingbird	53 cm	20 min	750 gm	200 gm	27%	18%
A.R.Drone	56 cm 64 cm	15 min 12 min	550 gm 550 gm	150 gm 100 gm	27%	12%
Pelican	75 cm 80 cm	20 min 18 min	1250 gm 1250 gm	500 gm 370 gm	40%	30%
DF-X4	79 cm	20 min	980 gm	250 gm	26%	16%

Analyzing the capabilities of these platforms, it is safe to say that if the structure weight of a small platform can be developed, the payload capacity will improve significantly. At the same time, if the propulsion system for the platform is developed carefully, the flight endurance of a small-scale vehicle can be sustained. Accordingly, a dimension estimation technique [20] is adopted in this study to optimize platform size, flight endurance, and payload capacity of the quadrotor. The optimization algorithm uses the geometric constraint of the vehicle and the well established Momentum theory [21].

To improve the flexibility of sensor integration in a small-scale vehicle, [18] reported a construction approach where the electronics are stacked on the vehicle platform. The mainframe was designed around a PCB, which also acts as the medium for circuit construction. Since the PCB is composed primarily of fiberglass, a PCB frame provides sufficient rigidity without adding to the weight of the system [22]. This thesis accordingly adopts the idea through a monolithic integration of the frame and the actuators (i.e., motor-propeller assembly) of a quadrotor helicopter on a single

PCB, which can result in an exceptionally lightweight and low-cost UAV system that is capable of agile performance.

In addition to size, weight, and power restrictions of a small-scale platform, the vehicle demands an efficient control processor that can generate maneuver control signals quickly enough to demonstrate agile maneuver. To this end, micro-processor technology has been used in both small-scale and large scale quadrotors. However, when multiple sensing modules are installed along with vision sensors, the performance of the microprocessor degrades due to the sequential operating mechanism of the processor. In the past researchers have used FPGA hardware to carry the parts of the sensing and control activity that are time critical while leaving the other processes for the microprocessor [23, 24]. However, having multiple control boards increases the weight and power requirement of the quadrotor leading to compromised flight endurance of the vehicle. Because of the limitations of multiple processor integration, a single FPGA based quadrotor construction is proposed in this study. Moreover, an FPGA chip can be populated directly on the PCB frame, which is conducive to the light-weight design requirements of a performance optimized platform [25].

Chapter 3 presents a detailed design of a performance optimized quadrotor through development of a dimension versus performance optimization method. Further, a prototype quadrotor is developed with the help of PCB technology for frame construction and FPGA technology for processor integration.

2.3 Accurate Dynamics Modeling of Quadrotor

A model of the quadrotor UAV is necessary to facilitate controller design tasks and allow the validation of the design concepts through simulations. The topic of quadrotor helicopter flight dynamics has been extensively investigated in [26]. Additionally, a comprehensive overview of helicopter aerodynamics has been presented in [27], which provides an explanation of both the Momentum Theory and Blade Element Theory.

In more recent years, further approaches to modeling quadrotor platforms have been investigated and validated with real flight test data. In [28], the derivation of aerodynamic forces used in the equations of motion for stability and control analysis of a quadrotor is presented. In [29], authors gave a theoretical analysis of the dynamics of a quadrotor in order to develop a model of it. An interesting aspect of the work was a method to determine the moments of inertia in which the authors assumed masses with known geometric shapes attached to a center of rotation by thin rods. Authors in [30] developed a quadrotor robot using a custom-built chassis and avionics with off-the-shelf motors and batteries. Additionally, the authors provided modeling techniques for controller design of the vehicle. The work presented in [31] investigated three separate aerodynamic effects; namely, total thrust produced, blade flapping, and airflow disruption. They showed that these effects caused moments that affected attitude control, and thrust variation that affected altitude control.

Some of the most crucial effects to be modeled entail the dynamics of the motors [32]. In [33] a model of Direct Current (DC) motors is used as the actuator model for the quadrotor. From this model two main aspects are highlighted: first, the model is nonlinear, since it includes a quadratic term; second, it includes a time constant parameter, meaning that there will be a delay between input requested and actual rotor speed.

The dynamics models developed in previous literature are based on a single rigid body model. For deriving the system model, essentially two common approaches have been adopted; the Lagrangian approach [34, 35] or the Newton Euler formulation [29, 30]. However, due to the natural existence of many dependent inertial elements resulting from kinematic constraints involved in a quadrotor system composed of multiple bodies, a single rigid body can result in compromised accuracy [36].

In contrast, this work presents a multi-body approach for dynamics and control modeling of the quadrotor vehicle using bond graphs, wherein the mainframe of the vehicle is assumed to be one body, and each of the motor-propeller assemblies are considered as separate bodies connected to the mainframe via four revolute joints. Newton-Euler formalism with body fixed coordinates is used to model the dynamics of each rigid body. A DC motor model is used to incorporate the actuator dynamics. Unlike a single rigid body model, which ignores the effect of multi-body interactions that occur in a real system, the proposed comprehensive multi-body model can reproduce actual dynamic behaviour of the respective system [37, 38, 39]. The graphical nature and explicit power flow paths inherent in the bond graph formalism facilitate a graphical approach for multi-body model construction.

Using the numerical model developed by Bouabdallah in [35] a bond-graph based quadrotor dynamics model has been developed by the author in his past research work [12]. A similar approach is taken to develop a bond-graph model of a quadrotor in [40]. In this study, multi-body interaction dynamics is incorporated to the models to improve the accuracy of the model. In order to develop a comprehensive model, motor dynamics are also incorporated along with the multi-body interaction as presented in Chapter 4.

2.4 FPGA based flight Controller for Quadrotor

Because agility is a prerequisite for a UAV to react swiftly to external disturbances, the vehicle must have the dynamic capability to adapt to external perturbations without compromising vehicle stability. It is obvious that the dynamic response of the overall system can be improved by improving the dynamics of each component that builds up the quadrotor. The primary components that contribute to the dynamics are the four BLDC motors that drive the four propellers. It is possible to improve the dynamics of the quadrotor by enhancing the performance of the BLDC motor controller. Along with the BLDC motor controller, a robust stabilization control algorithm is mandatory to utilize the full potential (i.e., agility) of the vehicle. Accordingly, existing literature focused on the BLDC motor driver, motor speed controller, and stabilization controller is reviewed in the following sections.

2.4.1 High Performance BLDC Motor Driver/Controller

BLDC motors are widely used actuators for UAV systems. According to authors in [41], BLDC motors are also suitable for applications where the volume available for the drive installation is restricted. Therefore use of BLDC motors as UAV actuators has been a suitable solution until now.

Recently, several attempts have been made to drive BLDC motors using a six-transistor H-Bridge circuit to drive the three phases of the motor [42] and in some cases the hardware chosen for implementing the controller was FPGA [43, 8, 44]. In [8], authors have attempted electronic commutation of a BLDC motor. To spin the BLDC motor, the stator windings were energized in a sequence and the rotor position information has been used to define the proper energizing sequence. Typically, in sensor based control the rotor position is sensed using Hall effect sensors embedded in the stator. By reading the Hall effect sensors, a 3-b code can be obtained, with values ranging from one to six. Each code value represents a sector in which the rotor is presently located. Each code value therefore provides information on the windings that need to be excited to turn the rotor [45]. Since inverter switches are used to commutate the motor winding, there is switching loss associated with the driving of the motor. However, proper implementation of a low-loss commutation algorithm can minimize the switching loss and accordingly consume less power during motor operation, which can result in a power efficient actuation system for the quadrotor.

In order to vary the speed of the motor, authors in [46] proposed a speed control technique for BLDC motors that involves changing the applied voltage across the motor phases. The controller used a sensor-ed method based on the concept of pulse width modulation, PWM. In the last two decades, many sensor-less drive solutions [47] have also been offered to eliminate the position sensor for BLDC motors with trapezoidal back-EMFs. In [48], researchers have developed commutation to vary speed based on hysteresis control. Both hysteresis control and trapezoidal back-EMF base commutation are analogous to the hall sensor waveforms. However, sensor-less commutation is inefficient for low speed operation as the magnitude of back-emf generated in low speed is not enough to estimate the position information. Therefore a versatile motor driver requires sensor based commutation to derive the position of the rotor during commutation.

On the other hand, to implement closed loop speed control of the motor the simplest and most effective control scheme was found to be a proportional-integral (PI) controller; however, the choice of a proper structure to provide an anti-windup strategy is still an open question to the researcher [49]. To support the feedback control loop various speed measurement techniques are developed using hall sensor signals as the basis of estimation [50, 51]. Though the time based hall sensors are easy to implement in discrete hardware, they are susceptible to noise [52] due to the misalignment associated with the installation of the hall sensors. To overcome the noise in the time based speed estimation, speed observers have also been reported in recent studies [53, 54]. However, observer based estimation requires more hardware resources compared with the time based estimation. Accordingly, both estimation techniques are studied in this thesis with the aim to choose the appropriate technology suitable for FPGA implementation.

Although off-the-shelf BLDC motor controllers are readily available, they do not provide access to the commutation control algorithm and they are limited to low update rate operation. Using an off the shelf controller can have a compromised outcome in an agile quadrotor. In addition, off-the-shelf controllers do not feature closed loop control. Cognizant of the limitations in the off the shelf controller and adopting the technologies and ideas from the aforementioned research studies, a single FPGA based BLDC motor driver is developed and a closed loop speed control law is implemented in Chapter 5. Since a custom commutation technique is adopted in this ground up development of motor driver, it is possible to implement a low-loss commutation algorithm that can result in a power efficient motor driver when compared to the off-the-shelf speed driver that uses classic commutation technique for simplicity of implementation.

2.4.2 Robust Attitude Stabilization of Quadrotor

Control of a quadrotor is a challenging task due to the fact that the vehicle is under-actuated (i.e., the DoF of the system are greater than the number of actuators) and this presents strong nonlinear dynamics. Besides the non-linear behaviour, an added challenge for the controller is to stabilize the platform in the presence of unpredictable external disturbances like wind gusts or sensor noise.

Many flight control algorithms ranging from linear conventional PD control [55, 56] and PID control [57, 58] to more advanced non-linear control techniques such as backstepping [59, 60] and feedback linearization [61, 62] have been proposed in the literature. Authors in [63], present a comprehensive overview of such methods in the context of linear as well as nonlinear control systems. In terms of linear systems, multiple references are given to Single Input Single Output (SISO) PID control as well as optimal control strategies such as the Linear Quadratic Regulator (LQR) [64], Linear Quadratic Gaussian (LQG), and Gain Scheduling [65]. Nevertheless, only a few studies have been done in the case when the quadrotor is affected by external disturbances. For example, in [66], an integral predictive and robust nonlinear H_∞ control strategy for a quadrotor was introduced. In [67], novel nonlinear feedback control laws are proposed to compensate for modeling errors and perform robustly against external perturbations such as wind gusts. Table 2.2 shows a summary of the control laws with with their advantages, disadvantages and applications. In [68], the authors proposed the unconventional approach of a classic control law (cascaded control architecture) for stabilizing a platform with high dynamics, wherein the authors augmented a classical PID controller with another term for aggressive maneuvers. This controller consists of four terms. The first three terms are the same as like the ideal PID controller. The fourth term is trying to stabilize the rate of the Euler angle rate (acceleration) and to overcome the high acceleration effect.

Unlike a classical model independent PID controller, some of the non-linear controllers can efficiently handle external disturbance and model uncertainties. However, the prior art generally attempts to implement controllers on off-the-shelf vehicle platforms that are equipped with a controller capable of handling complex mathematical formulations. In addition to that, implementing a control algorithm and image processing algorithm in the same processing device can lead to latency in computation and accordingly poor control performance. In order to address this, the FPGA is used in this study as the sole processing hardware that will hold all the digital circuits of the auto-flight system. Since the FPGA is not suitable for deploying complex mathematical formulations, a numerically efficient disturbance rejection control technology is adopted here, widely known as the Active Disturbance Rejection Control (ADRC) law. The ADRC is a novel control technology proposed by Han [10] that borrows

Table 2.2: Review of existing control law implemented in quadrotor.

Control Law	Advantages	Disadvantages	Associated Research
PD and PID	Easy implantation, model independent, very common control scheme design in real life applications	Poor robustness when the system encounters multiple challenges, not optimal solution	[55, 56, 57, 58]
LQR	Handles complex dynamic systems, robust performance, asymptotically stable for controllable systems, large stability margin	Requires access to the full state which is not always possible, Difficult to implement in discrete hardware	[64]
Backstepping	Robust to external disturbances, handles all the states of the system and accounts for the nonlinearities	Difficult to implement in discrete hardware, Time inefficient	[59, 60]
Gain Scheduling	Simple implementation of the control laws over the full flight envelope	Resource inefficient, Time inefficient	[65]
Cascaded Control	Efficient in handling fast dynamics, Easy to implement	Not robust in handling external disturbances	[68, 11]

the idea of error driven control from classic control theory, rather than model-based control. It employs an extended state observer (ESO) [69] to estimate the external disturbance in real time and adjust the error driven control effort [70] resulting in a robust control performance with a low computational cost. Although the ADRC has been successfully implemented in many control problems, including for an aerial vehicle [71, 72], a survey of related literature suggests that its application in quadrotor stabilization has been extremely limited. However, only a simulation experiment has been conducted where an attitude control law was developed for quadrotor stabilization.

Utilizing the cascaded control architecture for handling aggressive quadrotor dynamics and ADRC control to compensate for external disturbances in real time, a novel ADRC and PID based cascaded control architecture is proposed in Chapter 6 of this thesis, that demonstrates the robust stabilization performance both in nominal flight and under wind gusts.

2.5 Monocular Visual Servoing of Quadrotor

The design of UAVs capable of performing precision navigation usually involves a trade-off between performance of the navigation sensors and the weight of the sensors. Due to the limitations on the payload of small quadrotors, research on vision based control that can perform meaningful tasks is still in its infancy. Recent research that applied a vision based control approach [73, 74, 75, 76, 77] used bigger quadrotors equipped with sophisticated sensors, as a larger vehicle offers high payload capabilities to integrate a large number of components. But larger quadrotors are neither agile nor cost effective when it comes to performing missions in a space constrained environment or dynamically changing weather. In addition, the orientation measurement of the UAVs is achieved by using an inertial sensor (i.e., IMU or AHRS) while position and velocity measurements are obtained from a GPS. The main drawback of the common GPS receivers used in aerial vehicles is the fact that GPS receivers do not work in indoor operations and the position measurements are inaccurate by up to a few meters when operating outdoors. Therefore, the GPS solution for precision navigation is not applicable in the case where precision hovering, tracking or position hold is required. In contrast, use of vision as a measuring system has multiple advantages over employing other sensors. For example, cameras are passive and in general are also light and cheap. They can be used for both indoor and outdoor applications as well. It is therefore necessary to analyze the implications of utilizing a vision sensor (i.e., a camera) that will appear in the control loop. Vision based position estimation and position control are referred to as visual servoing. The approach has been adopted in many applications for developing navigation control algorithms of aerial vehicles. The author in [78] presents an overview of different visual servoing techniques, differentiating between two main categories: direct visual servoing, in which the visual control is directly responsible for giving actuator commands to the robot, and indirect visual servoing, in which the visual control gives a control input to a lower level control loop that computes actuator commands of the robot. Since the quadrotor developed in this study contains an inner loop attitude controller, an indirect visual servoing approach is surveyed for the development of a visual servoing system.

Different contributions for solving this problem have been reported in literature. The key to the visual servoing mechanism is the relative position estimation algorithm based on analysis of feature information in the image. Two fundamental techniques have been reported in literature, one using a landing pattern or landing target and the other without any reference target. For example, among reported research that did not use a reference target, authors in [79] proposed a recursive method for estimating range using a Kalman filter with a monocular sequence of images. In [80], authors have developed a UAV system using an inertial measurement unit (IMU) and four onboard cameras for the purposes of localization and pattern recognition. In [81], authors have used optical flow for hovering flight and vertical landing control. In the former projects, calculations are done by the ground station, which reduces the requirement of on-board real-estate and allows for great processing power at the ground station, but leads to restrictions for achieving complete autonomy.

On the other hand, in [82], the author proposed a monocular iterative pose estimation technique to estimate all six degrees of freedom (DoFs) using a single camera and a passive target with at least five points with known geometry. The author developed a system of non-linear equations based on Euler-Rodrigus parameters and the quaternion representation of finite rotation formula. The solution is evaluated using iterative least square analysis of the equations. In [83], the author presented an on board vision system to detect the landing pad, where the 5 DoFs pose of the UAV is estimated via an onboard downward looking camera. In [84], the authors proposed an efficient design of the landing marker as well as a vision-based landing marker detection algorithm. In [85], a dual camera is adopted to estimate the 6-DoF pose of the UAV. Location is achieved via an onboard camera and a camera on the ground. In [86], a recursive least squares approach is developed to estimate range, assuming that height remains approximately constant above a certain target. Authors in [87] used a downward-looking camera to track a target with known characteristics. The motion of the target is also known. A template matching algorithm is used for acquiring the target in the images and is integrated with a trajectory controller for landing the helicopter. The position of the target in the image is used as the input to a robust Kalman filter. A linear controller based on a kinematic model of the helicopter is used to perform trajectory following and landing. A system is presented in [88] that would allow a UAV to land autonomously on a carrier moving on the ground (in a horizontal plane only). The authors used a Wii remote IR camera for tracking a known pattern of IR lights installed in the moving carrier. PID control techniques were used to perform the landing mission, augmented by second order derivative terms (acceleration of the vehicle). Visual servoing has also been demonstrated by combining vision with inertial measurements in a filter as shown in [89, 90]. Other approaches using different combinations of sensors can also be found: authors in [91] combine vision with GPS, in [92] a vision sensor is combined with IMU and GPS; in [93] a camera, laser range, and IMU as sensors were used for full navigation. Most of the reported algorithms are computationally expensive and demand a processor with high computational power, which in turn demands larger real-estate for implementation. Since small aerial platforms are SWaP constrained, it is important to stress the computing platform for integration of the visual servoing hardware and realization of its associated algorithms. Accordingly, this study proposes FPGA hardware as the solution for a computing platform for small-scale UAVs and adopts the idea of a monocular approach for visual servoing using a single landmark.

Since the early days of reconfigurable computing, FPGAs have been an attractive platform for implementing image processing [94] and real time computing. The inherent parallelism in typical image processing algorithms and the capability of FPGAs to harness that parallelism through custom hardware make FPGAs an excellent computational match. A tremendous amount of research has explored the use of FPGAs for various vision systems, ranging from simple filtering, to stereo vision systems, to multi-target tracking (see for example [95, 96]). Largely due to Moore's law, FPGA technology has made significant advances to the point where sophisticated imaging systems can be implemented on a single FPGA [97, 98]. This approach is unique, as it not only meets the SWaP requirement of the UAV, but also makes the image acquisition and image processing at high speed through the use of parallelism of FPGA hardware, and two SRAMs (Static Random Access Memory) that are integrated with the FPGA chip.

One of the other features of successful vision-based helicopter control is the frequency at which camera images are sampled and processed. Helicopters can move quickly and it is not guaranteed that the update rate of a vision system is high enough for control purposes. According to [99], on-board image processing must be performed at a frame rate of 30 Hz or higher for effective vision-based object tracking. In the case of FPGA hardware implementation, where parallel computation of camera control, image processing, and control law evaluation is possible, the update rate of the vision sensor is limited by the image capture rate of the camera. Accordingly, a camera is developed from the ground up to utilize the full potential of the image sensor.

In addition, this study borrows the concept of state estimation using Kalman filter to estimate relative velocity of the quadrotor using position information from the camera sensor. Subsequently, a cascaded position PID controller is implemented wherein the inner-loop controls the velocity estimated by the Kalman Filter and the outer-loop controls the position measured using image sensor data. As a reference for the position estimation a passive target containing a circle with known diameter is used. The complete FPGA hardware implementation of the proposed visual servoing technique is presented in Chapter 7.

2.6 Conclusion

The existing literature on agile quadrotor platform development techniques is reviewed in this chapter. In addition, the prior art of accurate dynamics modeling is also discussed. By assessing the state-of-the-art FPGA based BLDC motor driving technology in a UAV application domain, a power efficient quadrotor motor driver technology is identified for hardware implementation. After discussing the reported low level flight stabilization controllers for agile quadrotors, a novel cascaded PID and ADRC based stabilization controller is proposed for further analysis and prototyping in FPGA hardware. Finally, a high level single position control architecture is adopted through assessment of existing vision sensors and visual servoing techniques available in previous research studies.

Chapter 3

Design & Construction of a Performance Optimized Quadrotor

3.1 Introduction

As UAV research progresses, emphasis is given on platforms that is smaller, agile, and has high degree of autonomy. These platforms hold great promise to assist with both research tasks and civilian applications, but the difficulties in designing and building such robots often are an inhibitive barrier preventing their development. Along with the design challenges, the coupled nature of hardware and software subsystems in a vehicle often demand complex engineering skills to develop an appropriate device. Therefore, it is necessary first to understand the key factors involved in construction of an efficient quadrotor.

In the literature review, Table 2.1 has indicated that there are a limited number of commercially available quadrotors that are small, lightweight, agile and yet can provide effective flight endurance and payload capacity. Some of them are small ($\approx$ 35 cm) and cannot demonstrate reasonable payload capacity ($<$ 100 gm), while others are too big with superior flight endurance (≈18 - 20 min) and payload capacity (≈350 - 400 gm) but at the same time the large size (≈ 75 - 80 cm) of the vehicles results in compromised agility. In this chapter of the thesis, a generalized design strategy is introduced to construct an efficient quadrotor tailored to increase flight endurance and payload in a small form-factor quadrotor platform.

According to Kumar [7], scaling down a quadrotor platform can lead to a significant increase in agility of the quadrotor and its ability to react quickly to command maneuvers. Scaling has a tremendous effect on the dynamics and physical properties of the quadrotor as well. These properties include mass, inertia, linear velocities and angular velocities, flight-time etc. If the characteristic length of the robot is L, by scaling down the size of the robot, it can infer linear and angular accelerations at a scale of L^{-1} and L^{-2} [100]. Based on the data reported for the commercially available platform (see Table 2.1), a quadrotor size < 50 cm (vehicle diameter) is considered as one of the design requirements for this development process.

A dimensioning method is developed in Section 3.2 to optimize size, payload capacity, and the flight endurance of the vehicle. In Section 3.3, a ground up design of a small-scale quadrotor is proposed. The goal is to design a vehicle platform using PCB technology to reduce size and weight of the platform while maintaining rigidity of the structure. A single chip FPGA hardware based processing technology is adopted to meet SWaP constraints and add parallel computing to the platform. Design considerations also include making it cost effective, easy to manufacture, and easy to integrate multiple sensors, including a camera. Such a design is instrumental to achieve a fully autonomous agile aerial platform that can demonstrate high payload capacity and long flight endurance. Finally the performance of the constructed platform is evaluated in Section 3.4 and the concluding remarks are presented in Section 3.5.

3.2 Materials & Methods

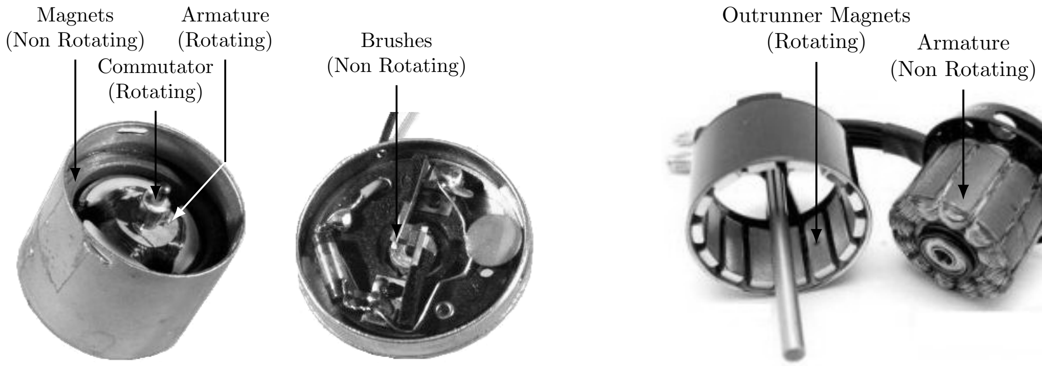
In this section the materials required for designing and constructing a high performance quadrotor platform are carefully identified for optimum design of size, weight, and efficiency of the platform. Next, a numerical framework is developed for estimating optimum platform dimensions based on design specified vehicle size, payload, and flight endurance.

3.2.1 Selection of Components & Materials for Construction

One of the primary components of the quadrotor is the electric motor used to construct the motor-propeller assembly (i.e., propulsion system). The choice of electric motor technology is very limited when it comes to construction of UAV platforms, namely brushed Direct Current (DC) electric motor technology and brushless DC motor (BLDC) technology.

The first brushed DC motor was invented in 1832 [101] by William Sturgeon. A pictorial representation of such motor is shown in Figure 3.1(a). Typically the electrical current is carried to the motor's rotating armature by the brushes (i.e., the commutator). These commutators mechanically define the exact timing required to produce a consistent torque on the motor shaft. Unfortunately, the mechanical arrangement is susceptible to wear and tear, which subsequently introduces efficiency losses to the motor. However, due to the simple commutation mechanism, the speed of the motor can be controlled with simple PWM signal prescribed using transistors only. Due to this simple control electronics many UAVs still employ the brushed motor as part of its propulsion system.

In 1962, the mechanical wear issues along with the efficiency losses were overcome with the invention of the BLDC motor [102]. A strip down BLDC motor image is



(a) Brushed DC Motor (b) Brushless DC Motor

Figure 3.1: Internal workings of a brushed and brushless DC motor

presented in Figure 3.1(b). As the name implies, the BLDC motor has no mechanical brushes and can demonstrate efficiencies around 90%. However, the electronics associated to the commutation control of the motor are complex and expensive. Typically a three phase commutation controller is used to operate the motor. Therefore, if the motor is accurately controlled with an appropriate controller, a higher efficiency can be obtained from the system. Accordingly, in this thesis, BLDC motor technology is chosen as the driver for the propulsion unit of the proposed quadrotor platform.

The next component in the propulsion system is the propellers. The performance of a propeller is dictated by its aerodynamic profile and rigidity. Different types of propellers are commercially available as shown in Figure 3.2. The design of the propeller vary based on the applications. The commonly used propellers include, the sport, thin-electric, slow-fly type, flexible type, and carbon fiber propellers. Sport propellers are designed for a high rpm operation (190K rpm/diameter in inches [103]) and are used for fast flight speeds. For medium to slow flight speeds thin-electric propellers are used that is designed to operate at medium rpm (145K rpm/diameter in inches). For slow flight and hovering machines slow-fly propellers are used, which are optimized for low rpm (65K rpm/diameter in inches). On the other hand, flexible

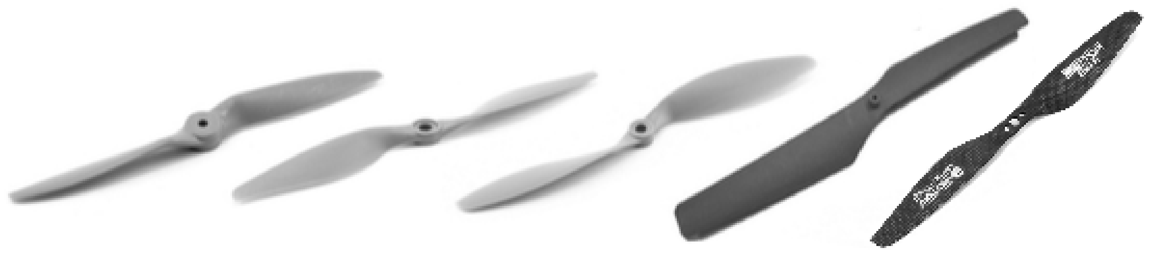


Figure 3.2: Different types of propellers used in UAV design. From left to right: sport, thin-electric, slow-fly, flexible, and carbon fiber propeller.

static-thrust propellers are designed keeping in mind the safety of the user and to operate at low rpm; whereas, carbon fiber (CF) propellers are stiff and optimized for a long rpm range, making them versatile for both sport and slow flying. However, CF propellers are expensive, due to the cost of the construction material. In general, the width of the blade chord is inverse proportional to the maximum operating speed of the propeller, in order to generate thrust at low speed operation. Furthermore, as the maximum operating speed reduces, a thinner width of the blades is chosen to make the propeller, which in turn reduces the rigidity and weight of the propeller. However, a propeller have a tendency to bend in the direction of produced force resulting in coning effect [104]. This phenomenon is prominent in flexible propeller, which makes it inefficient for high operating condition. Considering the limitations of the flexible propeller and versatility of the carbon fiber propellers, the propulsion unit for the proposed quadrotor design is constructed with a carbon fiber propeller. However, due to the rigid structure of carbon fiber propellers, vibration isolation needs to be considered during mechanical design of the platform, essentially to reduce noise in attitude estimation of the inertial sensor while the platform is flying.

To this end, the components of the propulsion system is in place. The next consideration is the structural design. Since every gram of extra weight added to the platform can contribute to the loss of endurance and loss of payload capacity, a light weight platform structure needs to be constructed. However, the structure must also posses rigidity to hold propulsion system and strong enough to withstand minor impacts during flight.

One of the most common material used in UAV construction is carbon fiber. Carbon fiber, primarily made of carbon atoms, is not only a light weight material but also posses high strength characteristics. The high strength of carbon fiber is due to the microscopic crystal structure formed by carbon bonded atoms, which are parallel to the fiber length. When a large number of fiber is combined together to form a yarn, and then to a fabric, it results in carbon fiber with high strength to weight ratio (1400 MPa and density = 1.55 g/cm^3 [105]). A composite is formed from the carbon fabric by combining it with plastic resin. However, carbon fiber can easily crack when it is subjected to a high impact force and usually it is expensive.

On the other hand, there is fiber-glass material, which can withstand impact force and also less expensive compared to carbon fiber. Unlike carbon fiber material, it has bi-directional fibers. However, its strength to weight (900 MPa and density = 1.85 g/cm^3) ratio is not as good as that of carbon fiber. As fiber-glass (FR4) is used to fabricate PCB, the material can be easily cut to any desired 2-D shape. Moreover, if a quadrotor mainframe is designed from a PCB, then the avionics, electronics, and autonomy sensors can be populated on the same PCB, as opposed to the carbon fiber frame body, where the components are mounted on the frame with fasteners and wirings that can add weight to the system. Use of PCB technology to form vehicle structure would not only reduce weight of the platform but also shrink the form-factor of the platform. Additionally, carbon fiber construction is expensive when mass production of a quadrotor is required. Since PCB based frame design offers a cleaner and monolithic platform construction is possible while also maintaining a good balance between size, weight, and cost of construction, it is chosen as the material for quadrotor frame construction in Section 3.3.4.

Since one of the design goals for the proposed quadrotor is to make it small and agile, consideration must be given to the selection of a processing computer. Typically, if the vehicle is small and lightweight, it cannot carry enough computational power to do some mission specific tasks as well as implement parallel processing to effectively control the vehicle while performing other time critical tasks, which would otherwise be easily possible with the greater payload allowances of larger vehicles through integration of multiple processors or a large computing platform. In the past, quadrotor vehicles under 500 grams have encountered SWaP constraints while integrating on-board vision-based autonomous navigation. Research has also been conducted on small agile quadrotors [6] that demonstrates the full on-board avionics based autonomy.

Given the fact that a small size platform inherits high dynamics and its control processor would demand real time performance while supervising all control modules including sensor data acquisition, data processing, and control law evaluation, the selection of a processing module becomes a challenging task. In addition to the real time computation requirement, design considerations like payload limitations, low power consumption, cost-effectiveness, and ability to fit within on-board available 'real estate' needs to be satisfied. Generally, micro-processor based computing units are employed for quadrotor controller implementation. However, when processing with multiple sensing units including processing of image data (which is a design requirement), performance of micro-processor based computation may introduce latency to the time critical control architectures due to the sequential nature of operation. As a solution to this, multiple processors are integrated into one platform affecting payload capacity, flight endurance and the agility of the platform. To address the constraints involved in integration of high computational capacity required by a small-scale agile platform, an FPGA technology based single chip parallel processing unit is chosen in this design. The FPGA based processor not only offers required computation power but also meets the SWaP requirements that are unique to small UAV platform designs.

3.2.2 Platform Dimension & Load Design Method

In order to establish the relationship between the size, flight endurance and propulsion efficiency of the quadrotor, a mathematical framework developed in [20] is in order. The framework is based on geometric constraints and propulsion efficiency of the vehicle. Accordingly, the propulsion efficiency is calculated from Momentum theory and propeller Figure of Merit (FoM). The FoM relates directly to the type and size of propeller used. By varying the battery weight and payload criteria, the size of the platform can be optimized for a specified payload and flight endurance. In addition, the formulation is also free from complex propeller aerodynamics as the Momentum theory incorporates the propeller thrust estimation model within its calculation.

First the weight of the platform is calculated based on a specified payload and flight endurance. As seen in Table 2.1, typical payload loading for small-scale quadrotor varies between 14% to 40%. Here, the payload loading, L_p is defined as the ratio between the maximum payload and take-off weight. Given a desired payload requirement m_p, and payload loading as a design specification, the take-off weight m_t can be determined using:

$$m_t = \frac{m_p}{L_p}. \tag{3.1}$$

The next step is to calculate the vehicle size based on a predefined ratio between the propeller and vehicle size. Assuming the diameter of the propeller, d_2 is 37% of the vehicle diameter, d_1 while the avionics consumes the center space of the quadrotor as denoted by d_3 in Figure 3.3, the geometry of the vehicle can be estimated as:

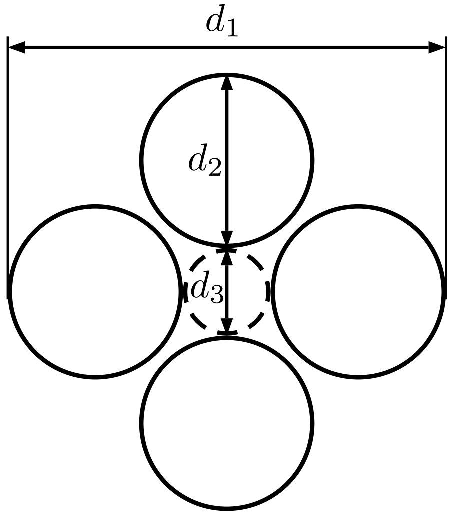


Figure 3.3: Geometric parameters of the quadrotor. Vehicle diameter (d_1), propeller diameter (d_2), and avionics real estate diameter (d_3)

.

$$d_2 = \frac{d_1}{2.7}, \qquad d_3 = \frac{d_1}{3.9}. \tag{3.2}$$

The propeller disk area, A is then calculated using the total number of propellers and diameter of each propeller.

$$A = 4\pi\left(\frac{d_2}{2}\right)^2, \tag{3.3}$$

where 4 is the total number of propeller in a quadrotor. It can be observed that as the radius of propeller increases, the area used to generate thrust increases quadratically. The required thrust, T generated by the propeller can then be estimated based on the ideal power, P_i required during hover, which is defined by Momentum Theory in [21]:

$$T = 2\rho A\left(\frac{P_i}{T}\right)^2 \tag{3.4}$$

where ρ is the density of air at a given altitude. Assuming hovering condition, when thrust is equal to the total weight of the platform (i.e., take-off weight), the ideal power required to hover can be estimated by rearranging equation (3.4) and replacing

T by $m_t \cdot g$:

$$P_i = m_t g \sqrt{\frac{m_t g}{2\rho A}}, \tag{3.5}$$

where g is the gravitational acceleration. The actual power, P_a required during hover can then be formulated based on the FoM of the propeller:

$$P_a = \frac{P_i}{\text{FoM}}. \tag{3.6}$$

Note that, the actual power estimated using equation (3.6) reflects total power required by four propellers. However, to estimate power required by one propeller, equation (3.3) needs to be reformulated for a single propeller. Since FoM is required to evaluate equation (3.6), Section 3.3.1 will identify FoMs for commonly used propellers including CF propeller.

Next, the propulsion efficiency η is determined using the actual power required to hover and the platform takeoff weight:

$$\eta = \frac{m_t}{P_a}. \tag{3.7}$$

As the flight endurance t_f is a function of the propulsion efficiency and battery capacity c_b, the battery capacity can be evaluated using the actual power required to hover and the desired flight endurance (in hours):

$$c_b = t_f \, P_a. \tag{3.8}$$

The weight of the battery can then be estimated based on the specific energy density, e_d of the chemical properties of the battery:

$$m_b = \frac{c_b}{e_d}. \tag{3.9}$$

Since Lithium-Polymer based battery is used in this study, the value of e_d is equal to 184 W-h/kg. Both η and c_b change with respect to the platform size. Therefore, it is necessary to estimate the battery loading, L_b to determine the platform size. L_b is the ratio between the battery weight and take-off weight of the platform:

$$L_b = \frac{m_b}{m_t}. \tag{3.10}$$

Once the battery weight is calculated, the structure weight m_s is evaluated from the the payload weight and battery weight m_b:

$$m_s = m_t - m_p - m_b. \tag{3.11}$$

The aforementioned mathematical framework presenting the relationship between the size of the platform, flight endurance, and specified payload is further used to determine the optimum size of the vehicle based on the specified design criteria.

3.3 Design & Construction of the Quadrotor

The design of the quadrotor begins with the implementation of the dimensioning method catered to the required flight endurance and payload criteria. Once the dimension along with other design requirements are estimated, actual construction of the vehicle would take place, which includes frame design, avionics system design, selection of the battery, and the design of a GCS. However, before this is possible a realistic propeller FoM needs to be determined to estimate an optimized dimension of the quadrotor.

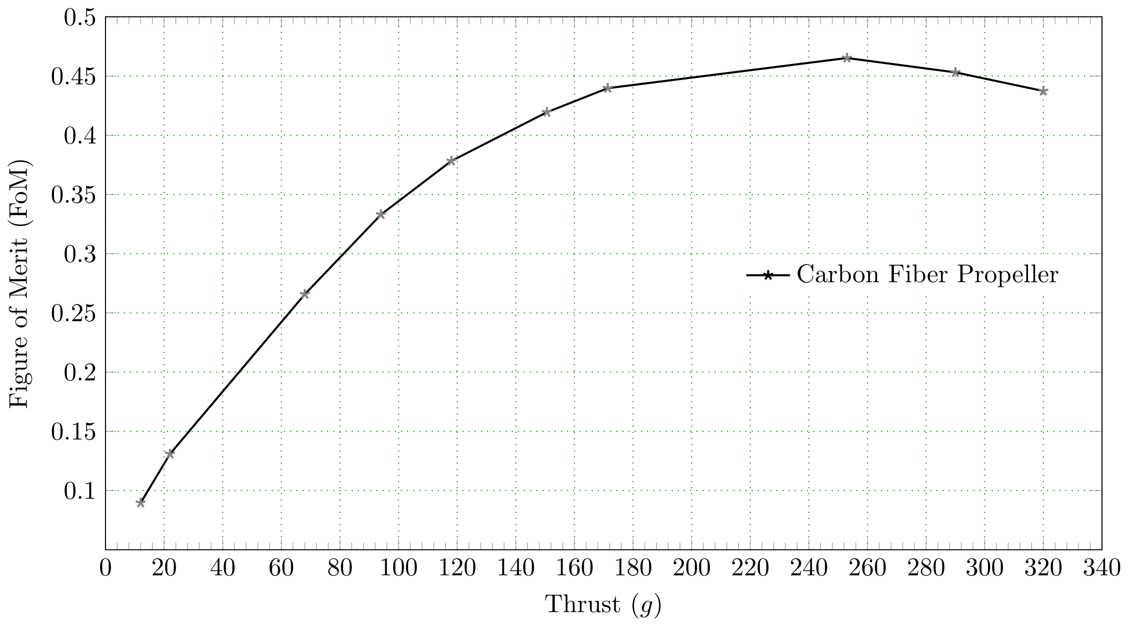


Figure 3.4: Measured FoM profile of carbon fiber propeller used in small-scale UAVs.

3.3.1 Propeller FoM Measurement

The FoM of a propeller, is defined by the ratio of the ideal power and the actual power required to hover [21]. Accordingly, the FoM is measured for a various propellers [21, 20]. However, information on carbon fiber propeller is not available in the reported literature. Static thrust tests have been performed here to identify the FoM of the propeller. In order to obtain an accurate estimation of the FoM, the efficiency loss of the motor needs to be minimized. Accordingly, a good set of motor-propeller combination is important that can efficiently generate propulsion. In this study, an off-the-shelf high efficiency BLDC motor (MN1806) was used to measure the FoM.

The FoM for the carbon fiber propeller is evaluated using measured actual power and the calculated ideal power from equation (3.5). Figure 3.4 presents the FoM profile of a carbon fiber propeller (CF 6×2, from T-Motor). Generally, the maximum value attained by the FoM profile is considered as the FoM value of the propeller. However, as the FoM of the carbon fiber propeller rises to a maximum value and then follows a downward trend, the FoM for the propeller is estimated as approximately 0.44. Whereas, FoM values for thin-electric, slow-fly, and flexible propeller are 0.37, 0.36, and 0.23 respectively [20], which are lower than the FoM of carbon fiber propeller. This concludes that, the carbon fiber propeller is a better candidate due to its high FoM, light weight and high stiff structure.

3.3.2 Estimation of Performance Optimized Dimension

Naturally, every platform design involves some predefined design goals, and the design parameters are estimated to achieve those goals. In this study, the design goals are to achieve a flight endurance of 20 min, a minimum payload capacity of 400 gm at 40% payload loading, and to harness the agility the platform needs to be less than 50 cm in diameter.

Using the dimensioning method, presented in Section 3.2.2, the recommended size of the quadrotor can now be determined. The platform weight can then be estimated based on the 400 gm payload requirement and the 40% payload loading, which in this case becomes 1000 gm. This means that, each motor needs to lift at least 250 gm. As the platform will be optimized for agility and endurance, the structure will need to be very light weight with a large battery. Based on the battery loading information presented in Table 2.1 a 27% battery loading is chosen here. This defines that the structure along with avionics can be as much as 33% of the total weight provided that the total weight is found to be 1000 gm for a 400 gm payload at 40% payload loading.

As shown Figure 3.5 left, for the specified payload requirement and flight endurance, a range of propulsion efficiencies can be determined using equation (3.2) through equation (3.7). This is based on the specified payload requirement of 400 gm at 40% loading and desired flight endurance of 20 min, estimated using approximate

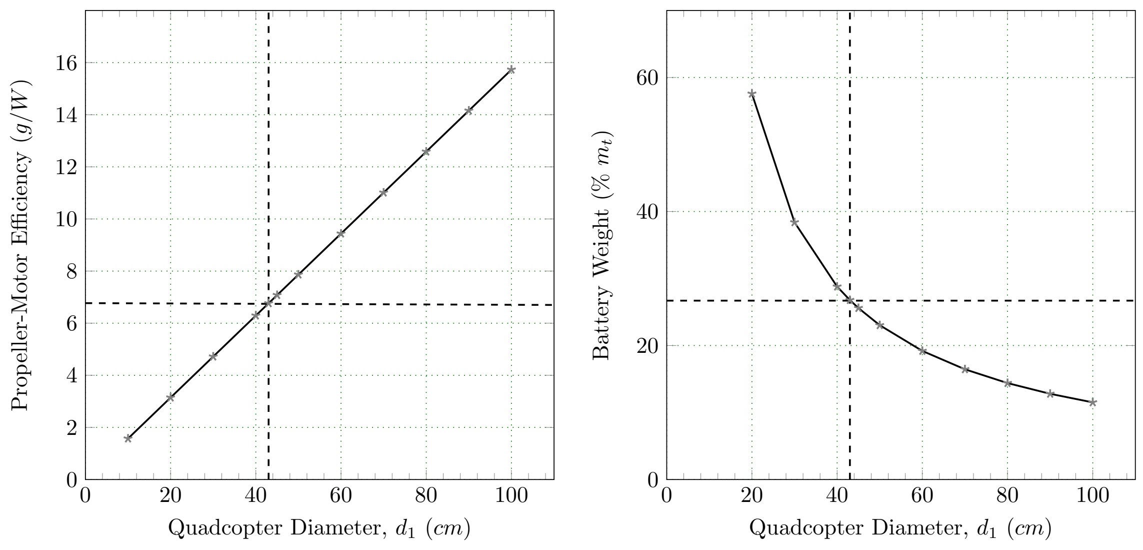


Figure 3.5: Estimation curve associated with design goal. Left: Estimated propeller-motor efficiency with respect to the platform diameter. Right: Estimated battery weight with respect to the platform diameter. The dotted lines on the right figure indicates the recommended platform size, which is selected using the desired battery loading ($m_p = 400$ gm, FoM = 0.44 (carbon fiber propellers at $T = 250$ gm), $L_p = 40\%$, $L_b = 27\%$)

maximum FoM of 0.44. Additionally, a range of battery weights, shown in Figure 3.5 right, corresponding to various platform sizes, can be determined using equation (3.8) through equation (3.10).

Based on the desired 27% battery loading, an estimated platform size of 43 cm ($\sim$ 17.0 inches) in diameter with a propeller diameter of 15.9 cm ($\sim$ 6.2 inches), and avionics real-estate of 11.02 cm (4.24 inches) in diameter are recommended for an agile 43.0 cm, high-endurance hovering platform capable of lifting 400 gm. In order to implement this in reality, a motor that is capable of producing at least 250 g of thrust with a propeller-motor efficiency equal to $\geq$6.7 g/W would be required. The dimensioning method also indicates that the goal structure weight should be approximately 330 gm.

3.3.3 Selection of Motor-Propeller Assembly

According to the design requirements, the chosen motor and propeller for the quadrotor construction were the T-Motor MN1806 and CF6×2 inches respectively. The motor-propeller combination was chosen, as it meets the required weight to thrust ratio and is very light weight (21.6 gm) when compared to its equivalents. In addition, the propeller is designed to fit with the motor without the need of additional components (made by the same manufacturer), which reduces the weight even further. The thrust curve for this propulsion module is shown in Figure 3.6. At the hover point thrust (with full payload), which is defined as 1/4 of the platform weight (250 gm), the motor-propeller efficiency can be determined. Notice that the hover point is neither too high nor too low in the thrust range as seen in Figure 3.6 left, which is the optimal point when trying to achieve high-endurance while carrying a high payload. This is only possible due to the reduction in structure weight. In addition the propulsion efficiency matches the required efficiency at hover point as seen in Figure 3.6 right.

3.3.4 Design & Construction of the Vehicle Mainframe

In order to accommodate the avionics, computing processor, and the motor-propeller assembly within the estimated structure weight of 330 gm while also maintaining a low profile construction within a 43 cm platform diameter (when viewed from the top), the PCB fabrication technique is chosen in this development. The PCB rapid prototyping technique allows for a tight integration between the structure, electronics and sensors, in addition to minimized wiring, reduced human labor and reduced complexity associated with the mechanical assembly.

A fiber-glass panel laminated with thin copper layers (i.e., PCB) can act as a

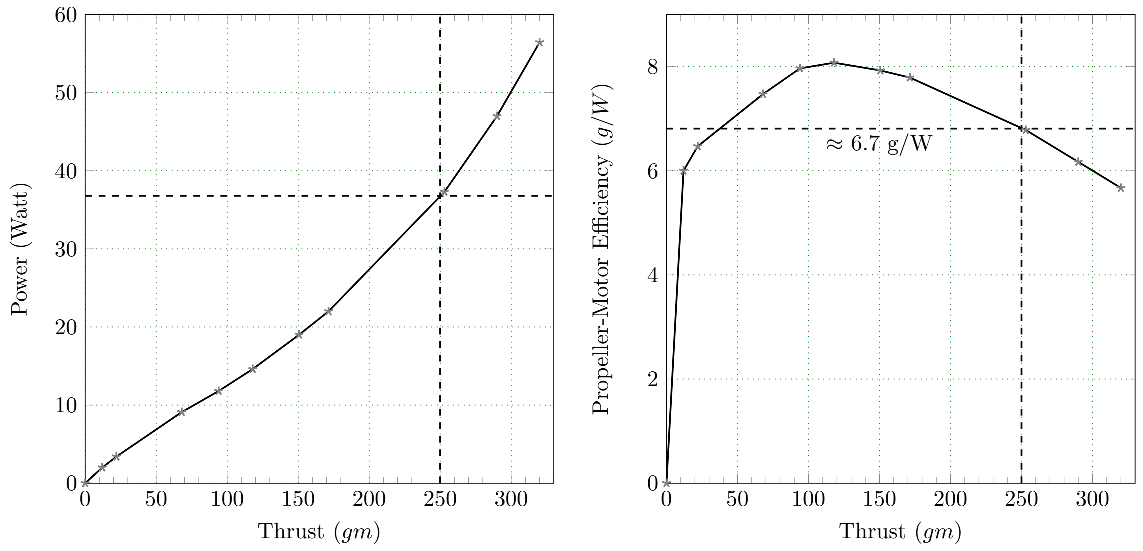


Figure 3.6: Thrust curve of the MN1806 motor with a CF6×2 inch propeller. The dotted lines in left figure indicate thrust and power at the hover point

low-weight and low-profile wiring medium for routing electrical connections traced on the flat surface. In addition, wirings can be routed in multiple layers. With the help of modern Computerized Numerically Controlled (CNC) machines, the boards can be cut into any 2-D shapes with a greater precision. This allows the shaping of a PCB frame as the quadrotor main body and allow sensors and electronics to be directly embedded into the structure of the flying robot, with minimal effort. This idea has been used to rapidly construct the platform structure as shown in Figure 3.7.

The entire body is fabricated from a single PCB with the motor connections directly routed into the four arms. The holes for fixing the motors and other parts are pre-drilled during the PCB fabrication process, so that all the pieces can be assembled easily. The structure already includes all the electrical connections, thus minimizing the wiring. The presented version of the airframe PCB holds the driver circuitry for four BLDC motor controllers, an attitude & heading reference sensor (AHRS), a Wi-Fi based communication module, and the power distribution module, all implemented in one 4-layer board. Increased copper thickness is chosen to handle

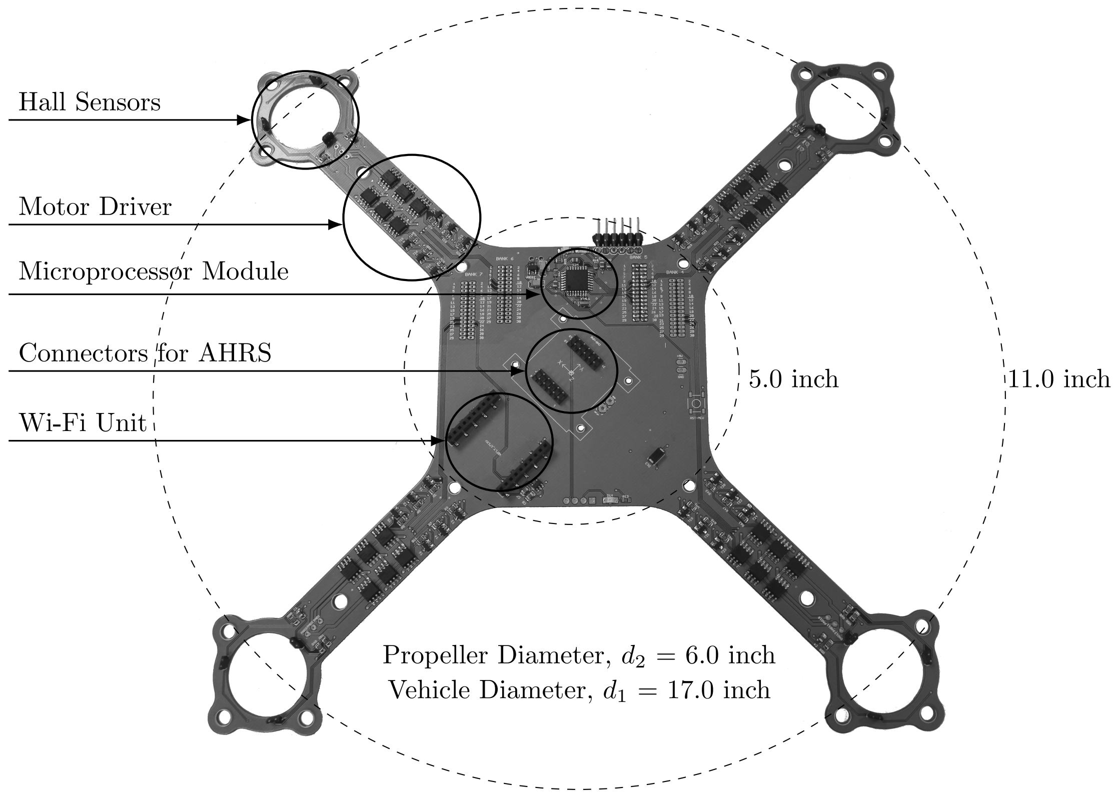


Figure 3.7: Image of a populated airframe PCB showing the location of the primary components and the associated geometric dimension.

the high current that will be generated by the BLDC motors. The characteristic length L (distance between center of one motor to the center of the opposite motor) of the PCB board is 11 inches ($\approx$ 27.9 cm). The platform diameter and the propeller diameter suggest that the real-estate for avionics installation needs to be limited to an area of less than 5 inches (12.7 cm). Though the subsystems collectively comprise more then 100 components, the components are densely populated to increase the real estate for future sensor integrations. For example, the component packages used in designing the PCB are 0402 packages for both resistors and capacitors measuring only 1.0 $\times$ 0.5 mm, while the larger integrated circuits (ICs) have a terminal pitch of only 0.5 mm.

The orientation sensor (i.e., AHRS) is placed at the center of the PCB and the orientation measurement axis is aligned with the control axis of the quadrotor (i.e, the axis drawn from one motor to the opposite motor). Similarly, the Wi-Fi module is mounted on top of the mainframe PCB. Motor driver circuitries are placed on their respective arms that hold the motors. The placement of the motor driver circuitry is chosen carefully to harness the air-stream from the propeller leading to a thermally efficient motor driver. This helps to cool the drivers, which is instrumental for longer flight time and longevity of the discrete ICs used to design the driver.

In order to assist the debugging process during development, the current version of the quadrotor employs three separate PCBs; one FPGA hardware board, one camera board, and the mainframe PCB itself. The platform is designed so that the FPGA hardware board and the camera board can be stacked below the mainframe PCB with minimal effort. However, in the future iterations the components of all boards will be populated in the vehicle frame PCB to reduce the weight even further. It needs to be noted that the gap between the stacked boards is kept as low as possible to assume a profile equivalent to a monolithic (all components populated on a single board) integration of the mainframe PCB, FPGA board and camera board.

3.3.4.1 Embedded Electronics of the Mainframe PCB

The main components embedded on the mainframe PCB include custom motor driver modules, hall sensors for rotor position estimation, an AHRS module for attitude sensing, a Wi-Fi module for wireless communication, and a microprocessor module for high level test bench design. The motor driver circuitry is designed to commutate a 3-phase BLDC motor, wherein the windings of the motor are wound in wye (Y-shaped) configuration. The main components of the driver circuit are three P-type MOSFETs (IRF9392) and three N-type MOSFETs (IRF7413Z) arranged in H-configuration to

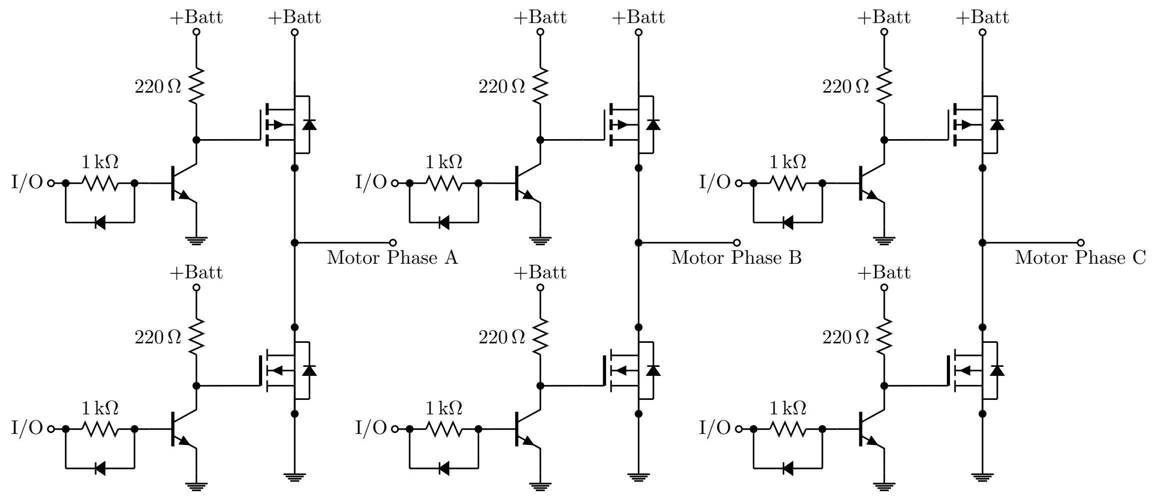


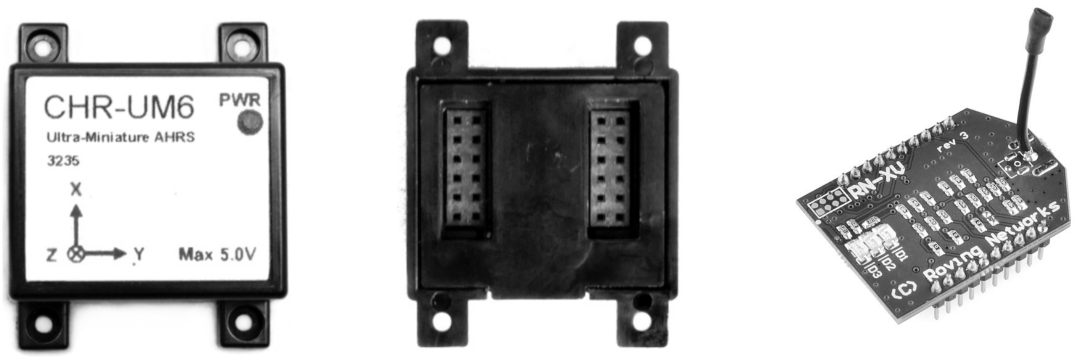
Figure 3.8: Driver circuitry for BLDC motor.

serve as a one-half H-bridge circuit. To drive the MOSFET switches from the FGPA I/O pins, NPN transistors (MMBT100) are used as seen in Figure 3.8. Detailed design analysis and design synthesis of the BLDC driver are presented in Chapter 5. The MOSFET switches are chosen based on low drain to source resistance $R_{DS(ON)}$ (i.e., low conduction loss), low switching loss, maximum supply voltage, and maximum current consumption. Basically, the switches are chosen to match the specifications of the motors at a full load condition.

In order to derive a motor commutation algorithm, three hall effect sensors are employed that provide the position feedback of the rotor using the magnets mounted on the rotor of the motor. For precision placement of the hall sensors, the components are mounted directly on the mainframe PCB (see Figure 3.7), encircling the holes for the motors. A1250 hall sensors from Allegro microelectronics were chosen to serve this purpose. The hall sensor switches its output at ±5 Gauss magnet field. Using a Gauss Meter, this sensitivity information was empirically identified, measuring magnetic field strength around the outrunner BLDC motor (the stator is at the center of the motor). Since hall sensors are sensitive to noise, a simple R-C noise filter circuit is populated as per the data-sheet provided by the hall sensor manufacturer [108]. The circuit also includes a pull-up resistor to amplify the sensor signal.

For the attitude sensing of this small-scale quadrotor design, a commercial AHRS module is preferred over developing an attitude sensor from the ground up. Due to the small form-factor of the proposed UAV, the dynamic property of the vehicle is expected to be faster; accordingly, a high data update rate AHRS system is desired. To strike a balance between the performance, weight, and the real-estate constraint, an off-the-shelf AHRS (CHR-UM6 by CHRobotics Inc.) is acquired. The module updates filtered attitude information and raw sensor data at 500Hz and 800Hz respectively. The AHRS incorporates the latest solid-state MEMS technology resulting in a low power avionics module. It is designed to be mounted on the frame PCB using 2×12-pin connectors as seen in Figure 3.9(b).

Typically, in commercial radio controlled (RC) aircrafts, a Radio Frequency (RF) receiver is used to receive and decode RF signals sent from a transmitter operated by a remote pilot. A receiver is not a requisite in a fully autonomous flight control system as no remote pilot is required. However, most of the UAV designs today retain the receiver component for failsafe purposes, where the human pilot has higher authority to remotely control the UAV during emergencies. In the proposed UAV, the receiver as well as the transmitter capability are implemented using an off-the-shelf 802.11b/g Wi-Fi protocol based communication module (RN-XV Wifly). Similar to the AHRS module, the Wi-Fi module is mounted directly on the PCB frame via 2×10 pin connectors as seen in Figure 3.9(c). Essentially, it is employed to receive control signals from a remote pilot or GCS and at the same time send back the telemetry information. The module communicates with the control processor using the serial universal asynchronous receiver/transmitter UART (RS-232) protocol. Once telemetry data is pushed into the Wi-Fi module using serial UART protocol, it is



(a) AHRS-Top View. (b) AHRS-Bottom View. (c) Wi-Fi Module.

Figure 3.9: Components housed on the mainframe PCB.

converted into UDP (User Datagram Protocol) protocol to broadcast over a Wi-Fi signal, which is then accepted by the GCS equipped with Wi-Fi trans-receiver and vice versa.

The mainframe PCB also houses an open source micro-controller [109] circuitry designed to populate on the main PCB that adds a data logging feature to the quadrotor. This data logging feature can support the debugging process during the development cycle by logging important flight telemetry data for post flight analysis. The analysis is critical for design modification or design improvement of the platform. The processor used in this design is an Atmel 8-bit AVR RISC-based microcontroller (ATmega-328) containing 32KB of flash memory and 2KB of Static Random Access Memory (SRAM). The processor is operated at 16MHz. The processor can also be used to perform a preliminary test of control algorithms and image processing algorithms before the actual hardware coding (i.e., VHDL) into the FPGA chip takes place. The adopted design philosophy is effective for time efficient development.

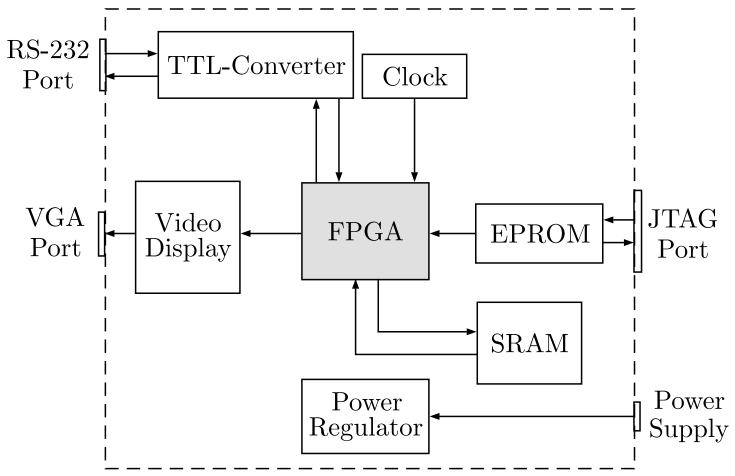


Figure 3.10: Functional block diagram of the FPGA Hardware Module.

3.3.5 Design & Construction of the FPGA Hardware PCB

Now that the mainframe PCB is constructed, the flight control hardware or the FPGA hardware PCB is designed and constructed to supervise tasks associated with avionics data acquisition, generating switching signals for motor drivers based on hall sensor readings, and to execute control laws using the sensory information. This means that the selection criterion was to find a cost efficient FPGA chip that can offer required logic elements for firmware implementation and have a sufficient number of I/O (input-output) peripherals to connect with all avionic sensors and motor drivers.

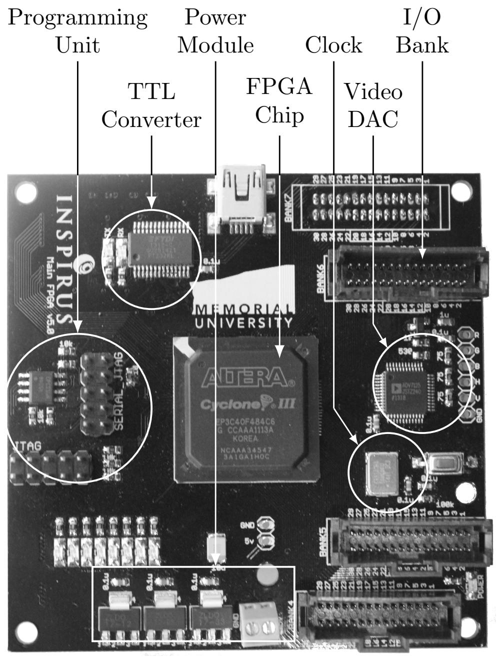
Accordingly, from a large selection of FPGA chips, the Altera Cyclone III - EP3F484C6 was chosen for this design. The FPGA chip consists of 39,600 logic elements and 128 9-bit multipliers. Note that the chip was selected keeping in mind the future extension of the sensor and avionics modules; therefore, it is not optimum hardware for the aforementioned avionics module. In this current iteration of the design, the FPGA hardware board houses only one FPGA chip; however, multiple chips can be installed depending on the design requirements. A functional block diagram of the FPGA hardware PCB is presented in Figure 3.10.

A JTAG programming unit is integrated into the FPGA board to store and run the program in real-time. A 16MB flash memory is used as the storage module. A 50MHz oscillator is integrated in the circuit to drive the FPGA chip; however, the coded firmware in the FPGA chip can be processed at much higher clock speed if a Phase Locked Loop (PLL) clock is implemented during firmware development. A serial to the USB chip (FT-232) is integrated to interface with a personal computer. Three 30-pin I/O banks (connectors) are used to interface with the quadrotor mainframe PCB. These pins run hardwired to the FPGA chip as well. Since the UAV is designed to operate with full autonomy based on a camera sensor, two SRAMs are hardwired to the FPGA chip to achieve faster image processing. A video DAC (Digital to Analog Converter) module is incorporated into the FPGA hardware board to broadcast images via the VGA (Video Graphics Adapter) port. Notes that the video DAC module and the VGA port are used for development purposes only.

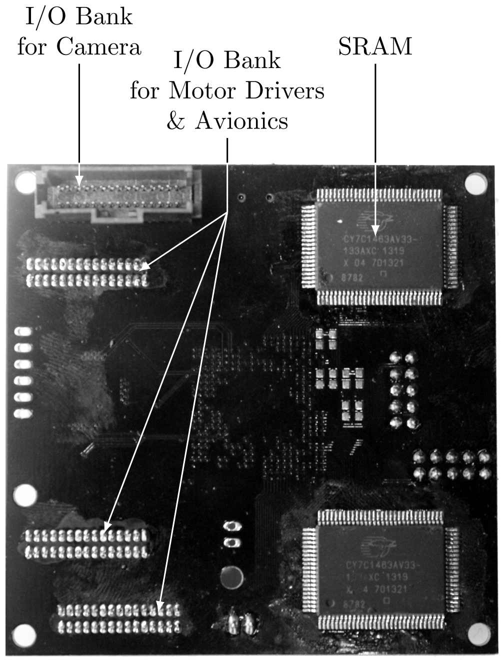
In order to match the dimension of the custom FPGA board with the estimated real-estate present for avionics integration with the quadrotor, the form factor of the board is chosen as 3×3 inches that fits within a 4.25 inches diameter circle, which is compliant with the design recommendations. A pictorial presentation of the board is presented in Figure 3.11. The compact form factor increased the challenge of laying out board traces, as the FPGA chip contains more than 120 I/O (input-output) pins as well as other peripherals. As a solution to this problem, the board was designed using 8-layer PCB, which increases the real-estate for laying out wire traces.

3.3.6 Design & Construction of the Camera Subsystem

In order to achieve vision-based localization, a stand-alone camera subsystem is designed to work with the FPGA hardware. Though the AHRS can also estimate the 3D position of the aircraft relative to its starting location by integrating the measured



(a) Top View

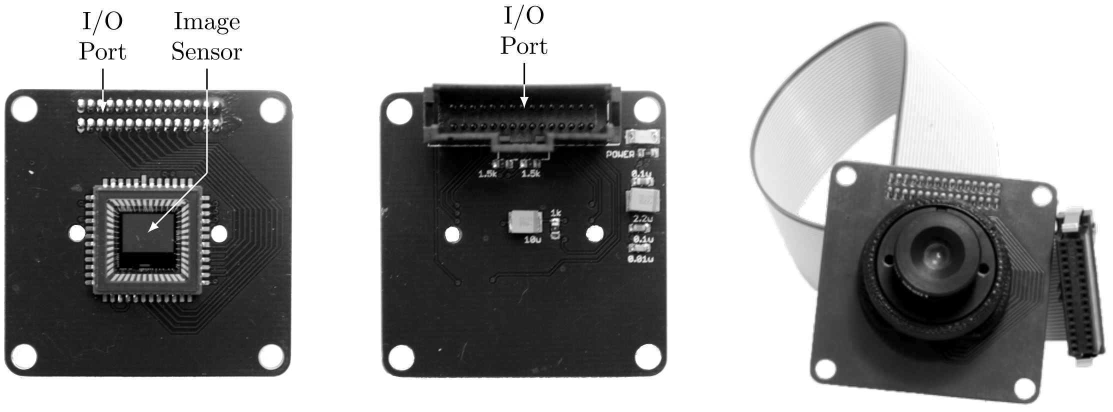


(b) Bottom View

Figure 3.11: PCB Layout of the FPGA hardware board.

linear acceleration information twice, the naive approach of integrating the acceleration introduces drifting that would make position estimation unreliable. Besides, the acceleration measurements from the accelerometer are based on the gravitational acceleration of earth on the accelerometer, which can be affected if the platform is tilted without moving in any lateral direction. These conditions suggest the use of external sensors for position estimation such as the Vicon motion tracking system [110], or a vision sensor.

In practice sensors like Vicon systems are not realistic for outdoor operation or cost effective for indoor tests. In addition, off-the-shelf cameras do not provide sufficient features to exploit the full capacity of the camera, in particular the frame rate of the camera. Since FPGA hardware is used in this thesis, which has the capacity to drive the image sensor while performing other tasks without introducing latency,



(a) Camera Top View. (b) Camera Bottom View. (c) Camera with cable.

Figure 3.12: Custom camera module to support quadrotor visual servoing application.

it is natural to develop a camera system ground-up.

Accordingly, a lens-camera assembly is designed and constructed using a three Mega pixel quarter inch digital CMOS image sensor (MT9T031C12STC) equipped with a 3 mm lens. Figure 3.12 presents an illustration of the custom camera module and its associated PCB layout. The camera PCB connects to one of the peripherals (I/O bank) of the FPGA board via a ribbon cable. All the components including the power module for the camera are integrated onto a single PCB.

3.3.7 Selection of the Battery

Technological advancement to improve the specific energy density of a battery is a slow moving process [111]. Since the first rechargeable Lead-Acid battery was invented in 1859, it has taken almost a century to get to the latest Lithium Ion Polymer (Li-Po/PL-iON) battery, which was first commercialized in 1996. There are three important aspects that make a battery useful for aerial vehicles. First, the battery must have a high specific energy density, which is a key factor to minimize the battery

Table 3.1: Power consumption by the components used in the quadrotor

Components	Supply (Volt)	Current (Amp)	Power (Watt)
4-Motors (at full payload)	8.2	18.40	150.88
FPGA module	3.3	0.270	0.90
Camera module	3.3	0.045	0.15
Wi-Fi module	3.3	0.036	0.12
Inertial sensor module	3.3	0.030	0.10
Others (estimated)	3.3	0.050	0.17
	Estimated total power consumption		152.32

weight and increase flight duration. Second, the battery must be rechargeable to enable renewable operation. Third, the battery must be relatively safe to prevent danger to the user or to the environment.

The Li-Po battery is most commonly used in UAV platforms. This is mainly because of its highest specific energy density when compared to other batteries like Lead-acid, Ni-Cd, NiMH and Li-Ion. Li-Po technology has a specific energy density ranging from 100 to 200 W.h/kg [112]. However, there is a new battery technology, named Lithium Sulfur (Li-S), currently being developed that is expected to be available soon in the commercial market. Li-S technology has a specific energy density of 350W·h/kg which is nearly twice that of the Li-Po technology. By simply replacing the Li-Po batteries with Li-S batteries, the flight endurance could be doubled.

Since Li-S is not available for use, a 2-cell Li-Po battery (typically 7.4 - 8.2V, maximum 8.4V) is chosen for the proposed design. In order to efficiently power the system for 20 minutes of flight time, the amount of current consumed by each component is measured. Table 3.1 presents the power consumption of the quadrotor components during a typical operation. As usual, among all the components the main power consumer are the motors. Based on the total power consumption (150.32 Watt), to achieve 20 min of flight operation at full payload, a battery with $\frac{152.32Watt*1hr*20min}{8.2V*60min}$

= 6.19 Ah capacity is required. Accordingly, a custom size (2.8×2.8×0.7 inch) Li-Po battery was ordered for manufacture. The dimension of the battery was defined based on the available real-estate underneath the avionics section of the quadrotor mainframe PCB. The battery weighs 266 gm and can discharge at a rate of 6.0Ah (maximum 30 Ah). The battery balances between both weight and the available energy, having an energy density of approximately $\frac{6000mA*8.2V}{0.266kg}$ = 184 Watt-h/kg. This is the same parameter that has been used in Section 3.3.2 for designing the optimum dimension of the proposed quadrotor.

3.3.8 Design of Inter-component Communication

After all the PCBs are made and the components are selected, the connection between the components is designed to make a prototype of the final assembly. A block diagram is shown in Figure 3.13 that depicts the connection diagram used for the quadrotor. Since the FPGA is the computing processor of the platform, it is at the center of the connection diagram.

In the proposed design, the motor drivers are powered directly from the battery and the low power rated components are channeled through the appropriate voltage regulator as shown with bold lines in Figure 3.13. Since the avionics along with the driver switches are controlled by the FPGA hardware, they are either hardwired to the FPGA chip or connected via I/O banks as discussed in Section 3.3.5. The FPGA chip collects motor hall sensor position information and triggers the right driving sequence as shown with the directional arrows in Figure 3.13. The FPGA chip also gather, orientation data from the AHRS, image data from the image sensor, and command data from the Wi-Fi module. The data is then processed to execute either manual or autonomous flight operation. In addition, the FPGA can configure an image sensor and upload data to the data logger module, to support post processing and real-time

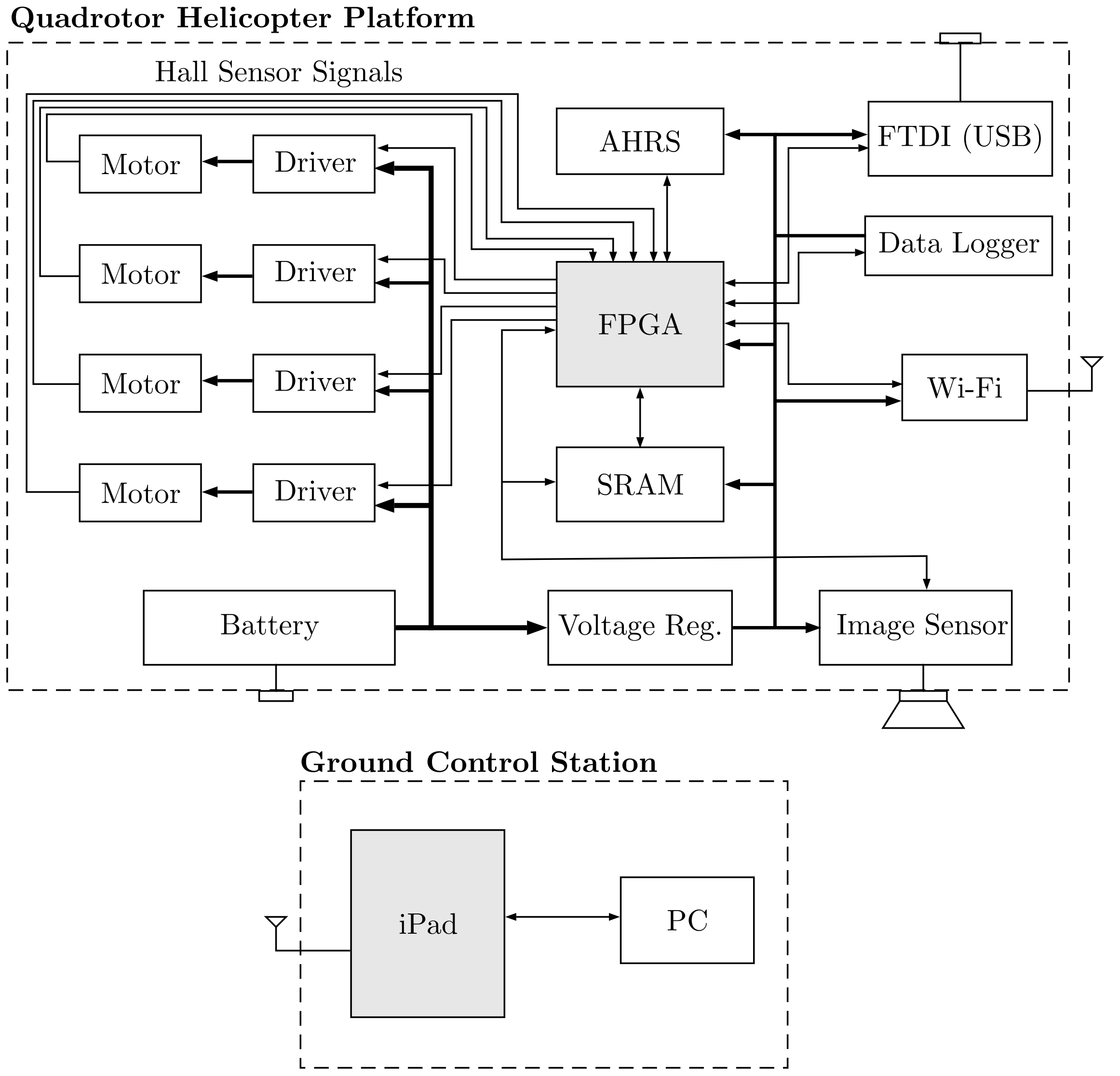


Figure 3.13: Block diagram illustrating the components of the system with their respective interconnection between each subsystem. The central FPGA unit acts as the hub between all peripheral sensors and devices while also acting as the low level controller for achieving stabilized autonomous flight.

visualization.

3.3.9 Complete Assembly & Weight Breakdown

The proposed quadrotor is an example of a modern mechatronics system, wherein adoption of both mechanical and electronics engineering design tasks have played a strong role during the prototype development. The assembly of the quadrotor takes place in a stacked fashion as seen in Figure 3.14. The FPGA hardware is mounted right underneath the mainframe PCB and connected via 4×30-pin connectors. The AHRS and the Wi-Fi module are mounted on top of the mainframe PCB using their respective connectors.

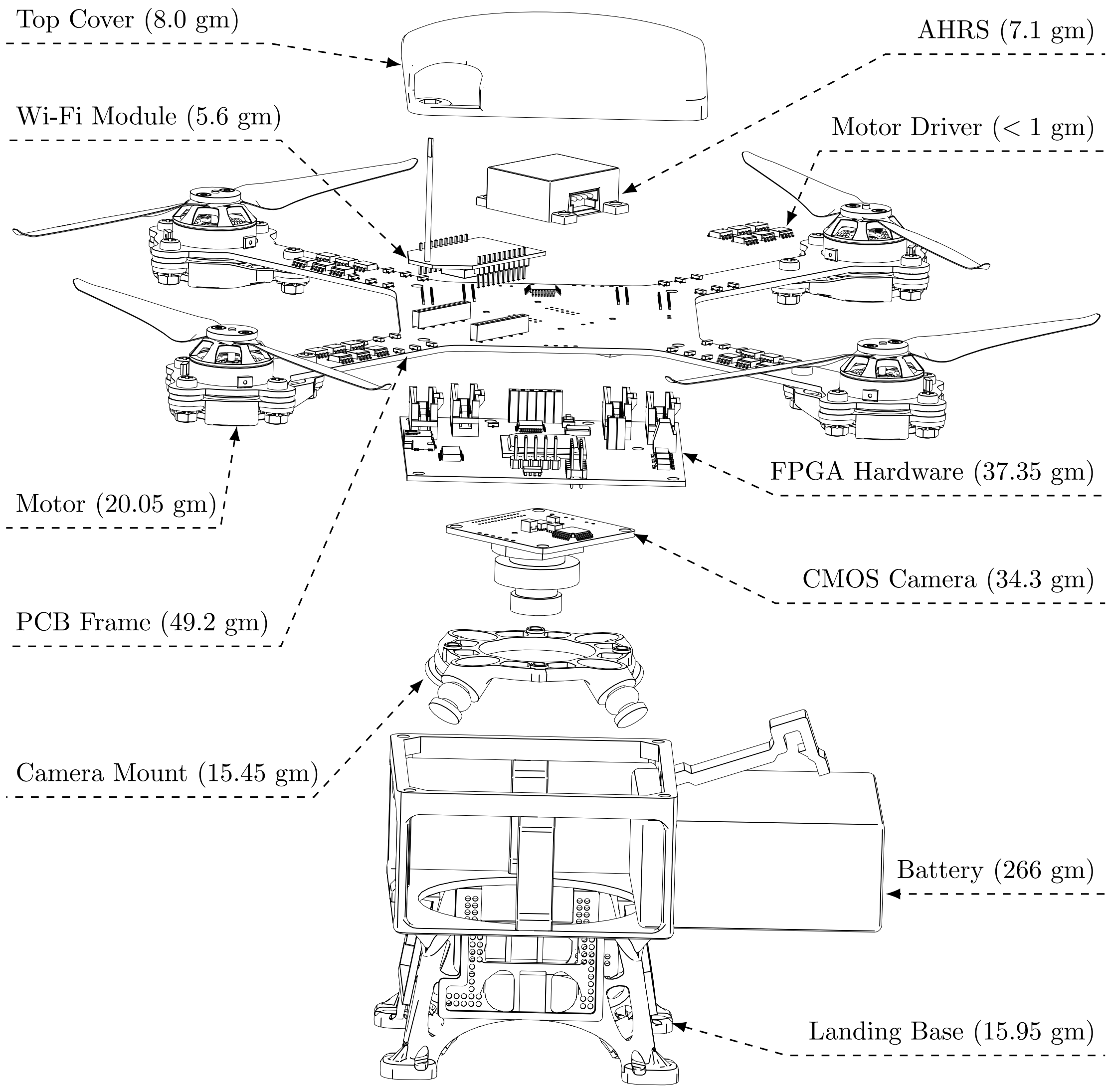


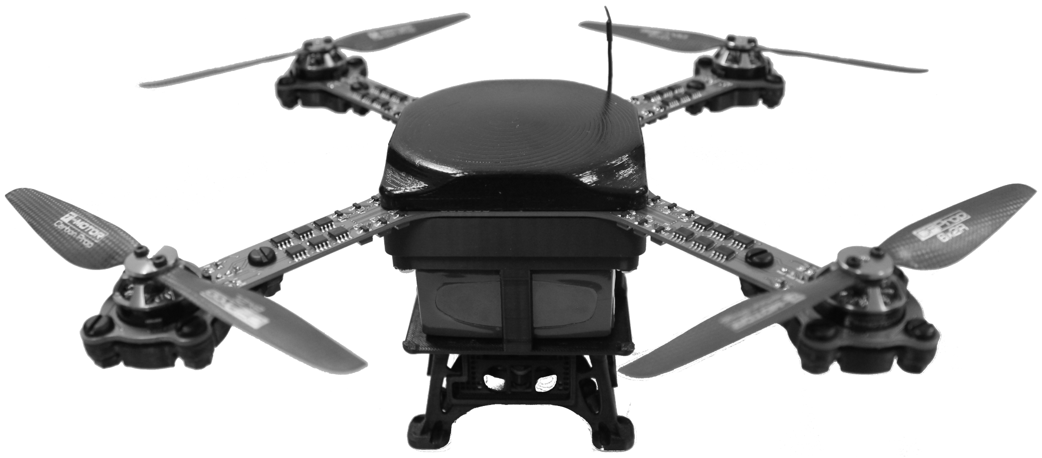
Figure 3.14: Exploded view of the proposed quadrotor platform.

The BLDC motors are recessed into the mainframe PCB board through their respective holes, to align properly with the hall sensors that are mounted on the PCB board and placed circumferentially around the holes. The motors are then mounted to the PCB board using a custom designed motor holder. The camera is housed inside the landing base facing down while it is mounted to the inner edges of the landing base. A cover is attached to the top of the platform to protect the components from dust and water. Finally, the battery is inserted horizontally into the landing base to complete the assembly process. Note that in a fully assembled prototype the camera sits below the battery as seen in Figure 3.15. However, to reflect the appropriate assembly procedure the camera is shown above the landing base in Figure 3.14.

The mechanical parts, namely the landing base/battery holder, motor holders and the top cover, are 3-D printed using ABS (Acrylonitrile Butadiene Styrene) plastic. 3-D printing technology is chosen in this development process due to its capacity of prototyping 3-D models with complex geometry. Since ABS is a thermoplastic material with high yield strength, it helps to reduce the weight of the component without compromising its rigidity. Detailed weight breakdown of the components is demonstrated in a pie chart as seen in Figure 3.16.

3.3.9.1 Vibration Suppression Mechanism

Since the quadrotor employs high speed BLDC motors that spin four carbon fiber propellers at high speed, the propulsion system introduces high frequency mechanical vibration into the vehicle frame. If the vibration is transmitted to the AHRS module that holds noise sensitive sensors, the sensor fails to estimate reliable attitude and motion information. Therefore, suppression of the high frequency vibration is critical for proper data estimation and improved stabilization control of the system. In order to prevent mechanical vibration from translating to the mainframe PCB, and especially to prevent it from affecting operation of the inertial unit and camera sensor,


(a) Isometric View of the vehicle.

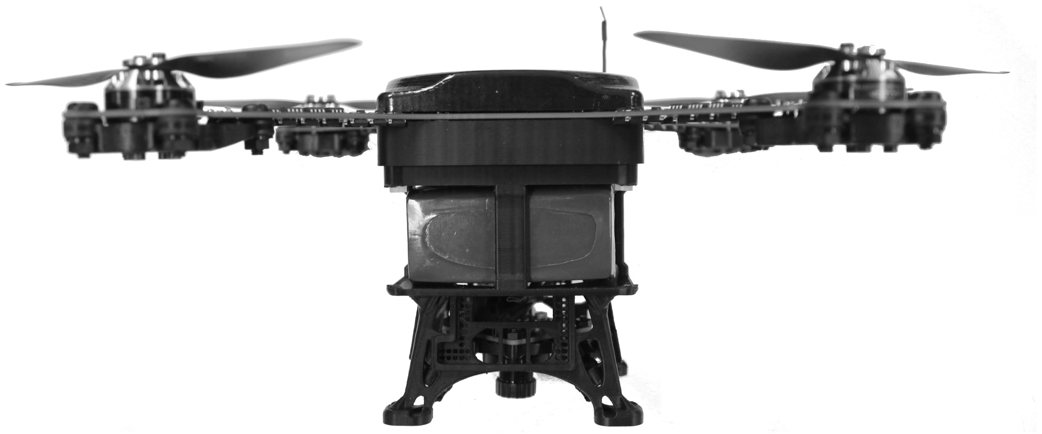

(b) Front View of the vehicle.

Figure 3.15: Images of the fully constructed quadrotor. Two different views are presented to show the propeller arrangement and the downward looking camera.

the propulsion unit is isolated from the airframe structure using four rubber vibration dampers as shown in Figure 3.17. The rubber grommets offer a complete isolation of the motor holder from the mainframe PCB, dampening the high frequency vibrations generated by the motor-propeller assembly.

A second stage vibration isolation mechanism is adopted to prevent low frequency vibration from propagating to the camera module. Four soft rubber ball dampers are installed at the joint of the camera mount and the landing base, creating flexible isolation points as shown in Figure 3.18 (camera is attached to the mount). Note that

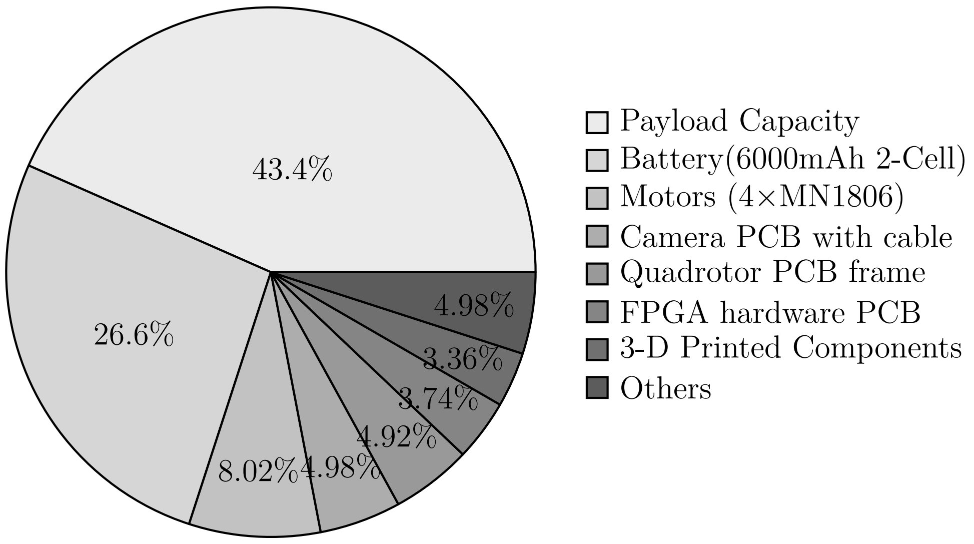


Figure 3.16: Weight distribution of the proposed quadrotor UAV.

the vibration dampers are empirically chosen in this study and a detailed vibration analysis is deferred for future research.

3.3.10 Ground Control Station (GCS)

In this development, the GCS is designed based on mobile applications for a commercial handheld tablet (i.e., Apple iPad). A pilot command interface (i.e., iOS Application) is developed for generating and transmitting command signals. The platform allows the user (i.e., pilot) to monitor the position of the vehicle in real time and to command the vehicle in manual flight mode. The GCS application also features options for emergency control command, switching between manual and autonomous flight and visualizing states of the passive target/landmark during visual servoing.

In order to reduce the learning curve for the pilot and to improve pilot handling qualities during manual flight, an intuitive flight control approach is adopted utilizing the built-in inertial sensor of the tablet platform. The attitude of the iPad is mapped to the attitude command prescribed for the aircraft, resulting in superior pilot handling quality. For example, to achieve motion towards the left, the pilot is required

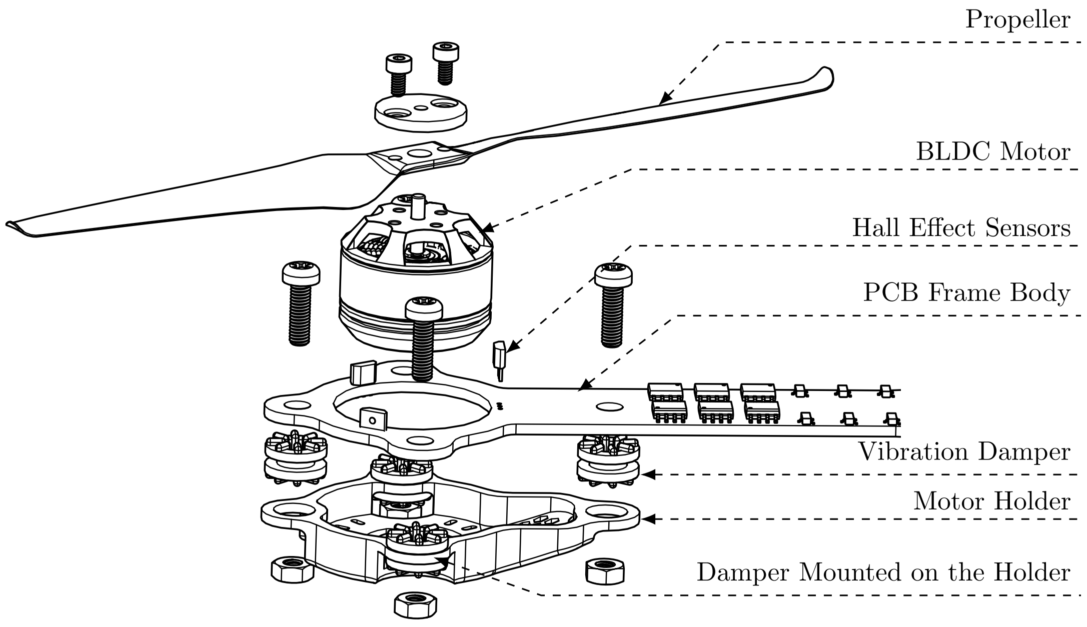


Figure 3.17: Vibration suppression mechanism of the BLDC motors.

to tilt the iPad left,and similarly to steer right, the pilot needs to roll the iPad to the right. The throttle is controlled using a slide bar located at the right of the application interface and is designed to be controlled with the right thumb as shown in Figure 3.19. An emergency STOP button is located at the left hand corner of the interface that is pressed to prevent an unexpected maneuver during development and debugging. The HOLD button on the left bottom is used to switch between manual and auto flight mode. Basically, pilot needs to hold the button to engage the manual flight mode and takeover from the auto-flight system. Once the button is released the vehicle maintains its attitudes at 0° on all three axes and can also hold its position in the air, given that the position information is available from the camera and passive target. On the other hand, if the position information is unavailable, the vehicle only maintains it stabilization, but the thrust command control still remains available to the pilot for altitude control.

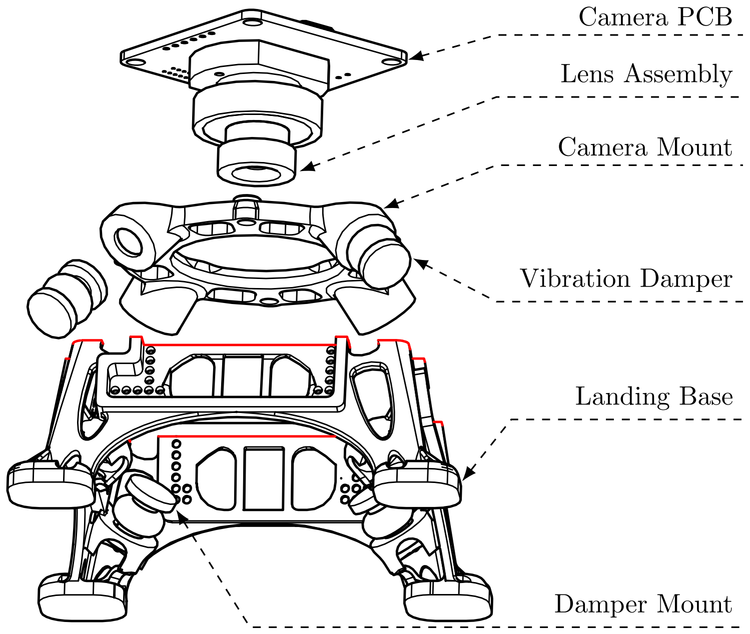
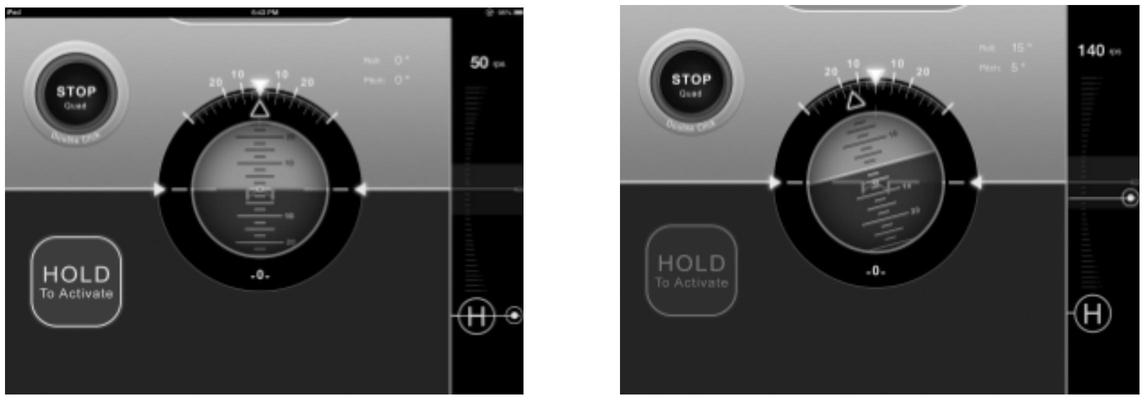


Figure 3.18: Vibration suppression mechanism of the camera subsystem.

3.4 Results

Once the quadrotor platform is constructed and tested for the functionality of the components the design performance indices are measured for verification. Table 3.2 shows a comparison between designed parameters estimated using the dimensioning method and the measured parameters obtained from the real platform. The designed parameters are very similar to the real platform in terms of size, propulsion efficiency, and battery loading. However, the structure weight and the payload have improved in the real platform. The ≈3.0% increase in payload loading is due to ≈3.0% decrease in structure loading. Despite having a good match between the required propulsion efficiency and the estimated propulsion efficiency, a 2 min loss of flight endurance is observed in the real platform. This could be attributed to the dynamics and efficiency of the battery when subjected to a long endurance operation, which is not studied as part of this design and construction study. In addition, the design strategy did not



(a) GCS interface during an auto flight mode

(b) GCS Interface : Sending manual flight command of 15° roll, 5° pitch, and throttle as 150 propeller rps

Figure 3.19: Interface of the Ground Control Station (GCS).

Table 3.2: Performance comparison between the optimized dimension design and the real quadrotor platform

	Size	Flight Time	Payload	Propulsion Efficiency	Structure Weight	Battery Weight
Design:	43.00 cm	20.0 min	400 gm (40.0%)	6.70 g/W	330 gm (33.0%)	270 gm (27.0%)
Real:	43.18 cm	18.0 min	434 gm (43.4%)	6.70 g/W	301 gm (30.1%)	266 gm (26.6%)

take into account the power consumption of the avionics, which would contribute to a slight reduction in the flight endurance.

In order to compare the standing of the proposed platform among the off-the-shelf platforms that have been presented in Section 2.2, a graphical comparison is presented in Figure 3.20 showing the maximum payload versus the platform size. Similarly, a graphical comparison is presented in Figure 3.21 showing the standing of the proposed platform in terms of endurance of flight. Since the performance data of off-the-shelf quadrotor only specify maximum value of the flight endurance and payload loading separately, a direct comparison is not possible as the flight endurance is a function of

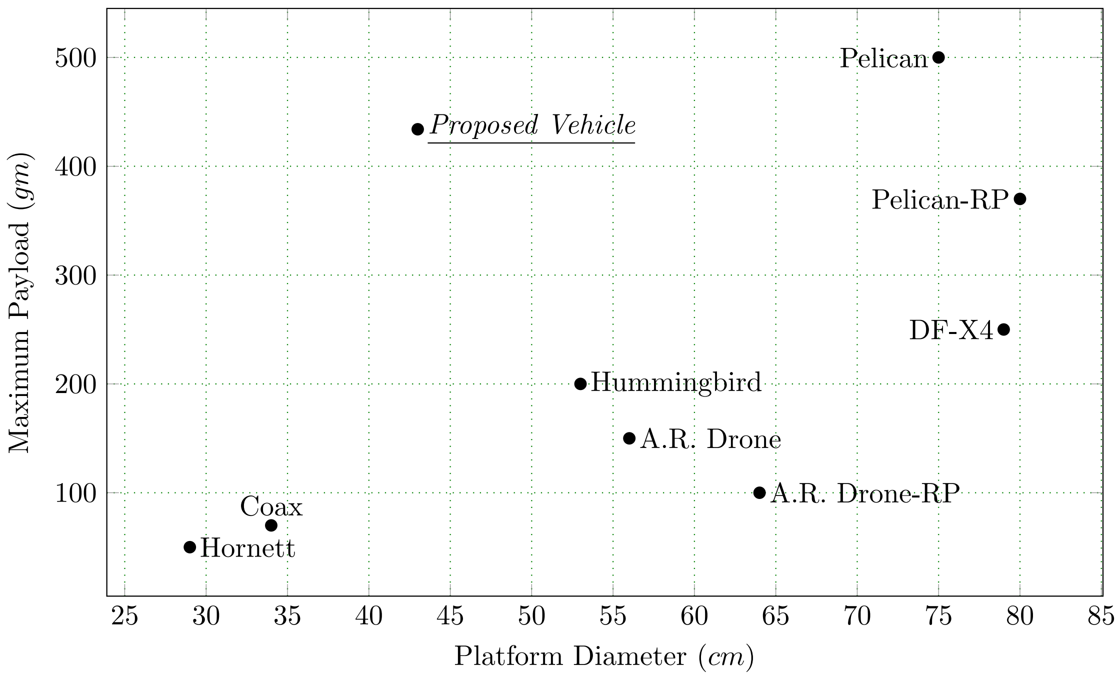


Figure 3.20: Payload capacity versus platform size comparison of the proposed quadrotor with representative commercially available platforms.

payload added to the vehicle. Therefore, the comparison is only presented based on the reported information.

When comparing the payload capacity among the representative platforms, it is evident that the proposed system is capable of carrying almost twice as much payload as any other platform of the same size and is almost equal to Pelican-RP, one of the high payload loading platforms. However, the Pelican-RP is very large and so agility is compromised for the platform due to its size, which is not the case for the proposed platform. Additionally, the maximum flight endurance is very close to that of the Coax platform, one of the high endurance platforms as seen in Figure 3.21. But the payload loading for the Coax is almost 1/4 of the proposed platform. Hence it is safe to claim that the proposed vehicle is well optimized for flight endurance and payload loading while being a small size platform, which is required to harness agility of the vehicle.

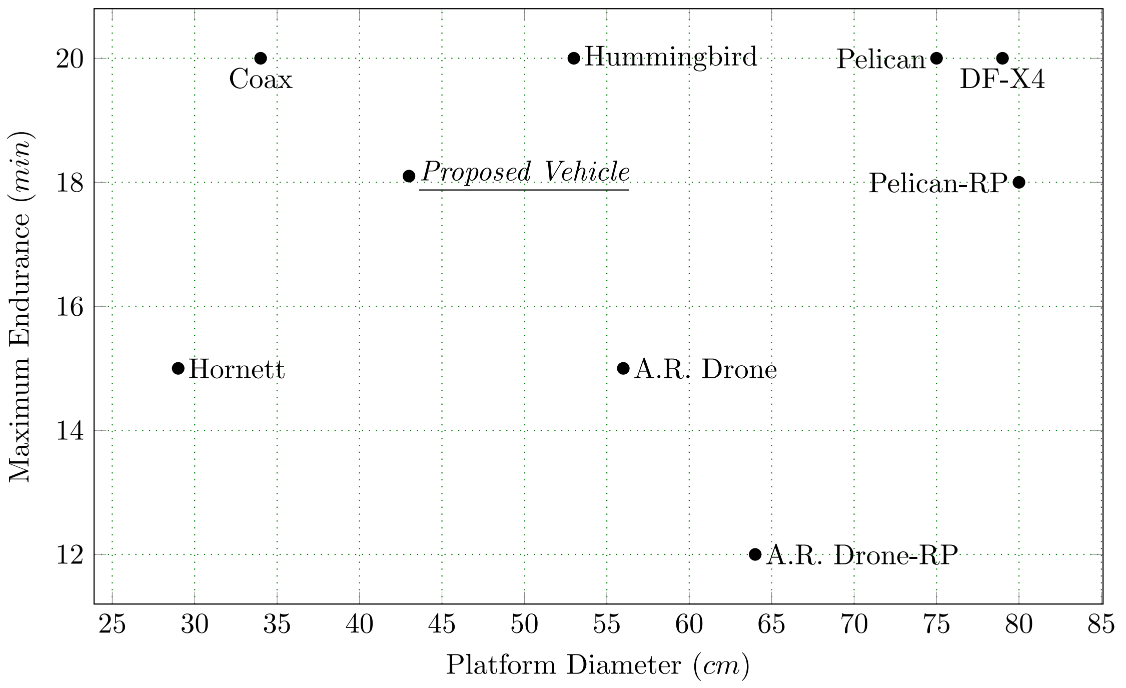


Figure 3.21: Flight endurance versus platform size comparison of the proposed quadrotor with representative commercially available platforms.

3.5 Conclusion

The existing quadrotor platforms are either too large to be able to perform agile maneuvers or too small to offer balance between payload loading and long endurance operation. In order to participate in lengthy missions and carry mission specific sensors while also being small and able to perform agile motions, it was necessary to design and construct a high performance quadrotor that has a balance between size, payload loading, and endurance of flight. To accomplish this engineering challenge, a generalized design strategy has been employed to estimate the dimensions of a hovering platform for a specific flight endurance and payload loading criteria.

Based on the high FoM value of carbon fiber propeller, it has been concluded that carbon fiber propeller in combination with MN1806 motor would be suitable to construct the propulsion system for the endurance and payload balanced platform within a reasonably small form-factor (platform diameter <50 cm).

Accordingly, a quadrotor is designed and constructed using PCB technology to reduce structure weight while maintaining rigidity of the platform. In addition, PCB technology reduces the need for additional wiring, lowers cost when mass produced, improves the manufacturability, and makes the assembly process time efficient.

Finally, to accommodate a powerful processing unit within a size constraint platform, single chip FPGA technology based hardware is chosen for the proposed platform. The FPGA hardware not only meets SWaP requirements but also allows real-time parallel processing of the data acquisition & control tasks, which is critical for an agile platform to conduct time critical maneuvers through execution of firmware modules in parallel.

The optimized dimensioning strategy, together with the PCB technology for platform structure development and the FPGA technology as platform processing hardware, has produced an efficient quadrotor platform that can carry a large payload, conduct long endurance missions, and still be agile due to the small form-factor of the vehicle.

Chapter 4

Multibody Dynamics Modeling and Control of Quadrotor using Bond Graph

4.1 Introduction

The study of kinematics and dynamics help to understand the physics of the quadrotor and its behaviour [26]. Together with modeling, the determination of the control algorithm structure is very important for improving stabilization. Thereof, design of a controller involves the development of an accurate system dynamics model. Researchers have modeled and simulated a quadrotor in different ways. Authors in [28], derived the differential equations and formulated the dynamic model, and developed a dynamic feedback controller. P. McKerrow obtained the dynamic model for theoretical analysis of a quadrotor [29]. Researchers also obtained a nonlinear dynamic model of a quadrotor for state variable control based on the Euler angle and an open loop position state observer, and emphasized attitude control rather than the translational motion of the UAV [30]. Authors in [34] performed autonomous take-off, hovering and landing control of a quadrotor by synthesizing a controller using the Lagrangien model based on the Lyapunov analysis. Authors in [35] developed a Lagrangian dynamic model, and designed sensing & control of an indoor micro quadrotor.

In previous research, dynamics equations of the quadrotor are developed assuming a single rigid body, which compromises the effect of multi-body interaction that occurs in the real system. This is mainly because of the fact that as the number of bodies in a system increases, the detailed dynamic interaction structure of the system becomes quite complicated due to the natural existence of many dependent inertial elements resulting from kinematic constraints involved in the system.

To supplement this, a multi-body dynamics modeling approach for a quadrotor using Bond Graph representation is proposed in this chapter. A multi-body approach typically writes 'exact' algebraic constraint equations [37]. Unlike Newton-Euler or Lagrangian formulation, constructing a 3-D multi-body with multiple kinematic loops in bond graph does not require extensive analytic derivation of the kinematic constraints. This is because the multi-body bond graph modeling approach treats the system as an interconnection of a number of parts that interact dynamically. A bond graph describes the interaction at the graphic level using the exchange of energy as its basis, and is independent of the physical domain of the model. It represents the elementary energy related phenomena among the components (generation, storage, dissipation, power exchange) using a small set of ideal elements that can be coupled together through external ports representing power flow. The constraints between bodies are enforced approximately by elastic and dissipation elements which, it can be argued, model actual effects in real systems. This unique feature of bond graph formalism can be attributed to the significant contributions of the work in [113, 38].

For the bond graph development, commercially available bond graph simulation software named '20-SIM' has been chosen for this study. This is a very apt modeling tool, combining powerful model calculation features with useful visualization features.

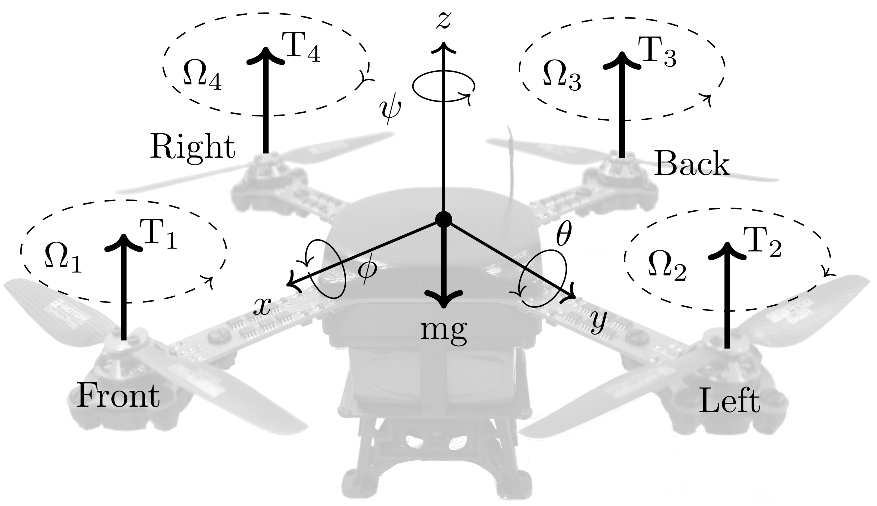


Figure 4.1: Schematic diagram of a quadrotor.

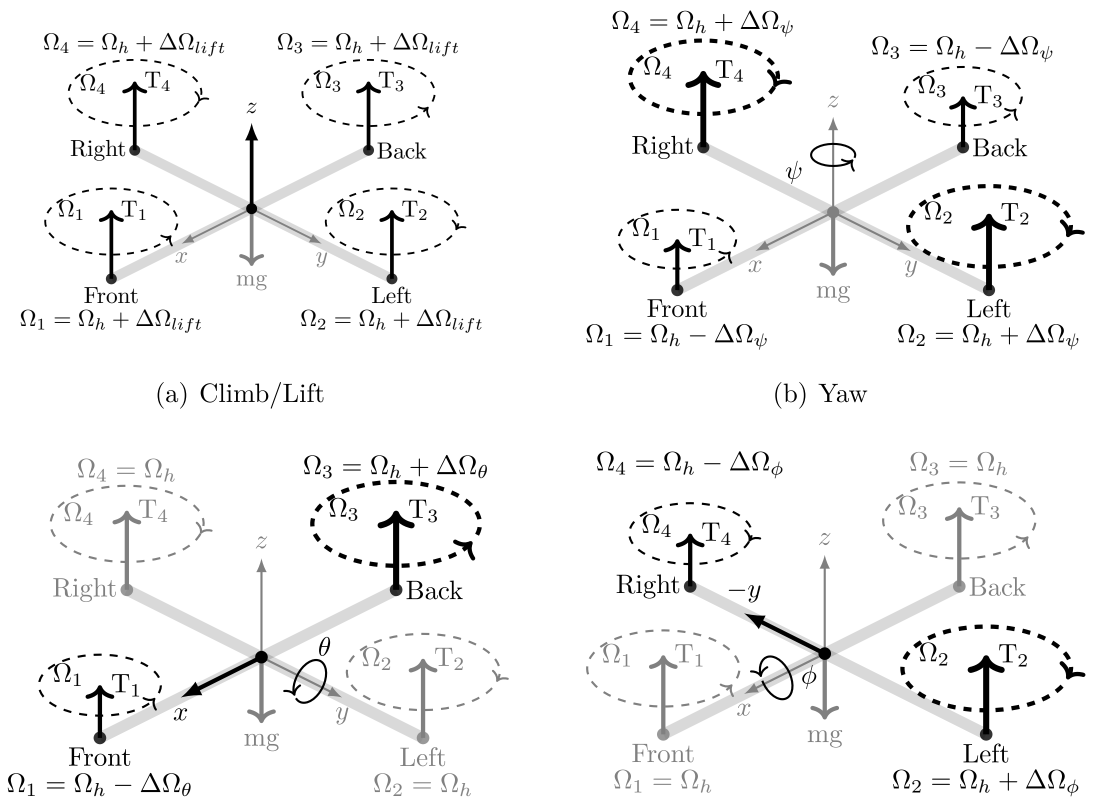
Most admirable is its DirectX 3D environment, allowing an instant 3D display of the model.

4.2 The Quadrotor

In order to understand the kinematics, physics, and control maneuvers of the quadrotor a brief introduction to the system is presented in this section.

A quadrotor is a non-coaxial multi-rotor aircraft vehicle which can achieve vertical take-off and landing (VTOL). The movement of the quadrotor is generated by varying the speeds of the four rotors relative to each other, by generating lift and torque of the four butterfly-distribution propulsion units. Unlike typical helicopter models, which have variable pitch angles, a quadrotor has fixed pitch angle propellers. A pictorial representation of the quadrotor vehicle with four fixed pitch propellers is shown in Figure 4.1.

The front and rear rotors rotate counter-clockwise while the other two rotate clockwise, so that the gyroscopic effects and aerodynamic torque are canceled in


$\Omega_4 = \Omega_h + \Delta\Omega_{lift}$
$\Omega_3 = \Omega_h + \Delta\Omega_{lift}$
Ω_4
T_4
Ω_3
T_3
z
Right
Back
Ω_1
T_1
Ω_2
T_2
x
mg
y
Front
Left
$\Omega_1 = \Omega_h + \Delta\Omega_{lift}$
$\Omega_2 = \Omega_h + \Delta\Omega_{lift}$
$\Omega_4 = \Omega_h + \Delta\Omega_\psi$
$\Omega_3 = \Omega_h - \Delta\Omega_\psi$
ψ
$\Omega_1 = \Omega_h - \Delta\Omega_\psi$
$\Omega_2 = \Omega_h + \Delta\Omega_\psi$
$\Omega_4 = \Omega_h$
$\Omega_3 = \Omega_h + \Delta\Omega_\theta$
θ
$\Omega_1 = \Omega_h - \Delta\Omega_\theta$
$\Omega_2 = \Omega_h$
$\Omega_4 = \Omega_h - \Delta\Omega_\phi$
$\Omega_3 = \Omega_h$
$-y$
ϕ
$\Omega_1 = \Omega_h$
$\Omega_2 = \Omega_h + \Delta\Omega_\phi$

(a) Climb/Lift

(b) Yaw

(c) Pitch & x Translation

(d) Roll & $-y$ Translation

Figure 4.2: Motion of the quadrotor; (a) Climb/Lift, (b) Yaw, (c) Pitch, (d) Roll.

trimmed flight. The basic motions of the vehicle can be described using the models presented in Figure 4.2. The rotors generate thrust forces (T) perpendicular to the plane of the rotors and moments (M) about the x and y axes; the moment about the z-axis is obtained using counter torques of the rotor. Increasing or decreasing the speed of the four propellers simultaneously permits climbing and descending (Figure 4.2(a)). Vertical rotation (yaw) is achieved by creating an angular speed difference between the two pairs of rotors which in turn creates reactive torques (Figure 4.2(b)). Rotation about the longitudinal axis (pitch) and lateral axis (roll), and consequently their coupled horizontal motions, are achieved by tilting the vehicle. This is possible by changing the relative propeller speed of one pair of rotors (Figure 4.2(c), 4.2(d)).

4.3 Multi-body Model Construction

Combining the ideas from [113, 38], the first step in the multi-body modeling approach is to develop a systemic model construction procedure comprised of the following steps:

1. Identify the privileged frame, a concept developed by Favra and Scavarda in [38] to systematize the construction of multi-body models with kinematic loops at a graphic level. In the case of the quadrotor helicopter, a body fixed frame is chosen as the privileged frame. This is partly because of the fact that sensor information and actuator forces are exerted on a body-fixed frame, so it is more natural to write up the dynamics using a body-fixed frame.

2. Define all independent bodies that will construct the system. In this study, the quadrotor is considered to be comprised of five different bodies: there is a mainframe that can move in all six degrees of freedom, and four propulsion units (motor-propeller assemblies) that can spin independently on one axis while attached to the mainframe at some defined points. It is assumed that individual components considered here can be adequately modeled as rigid bodies.

3. Construct the dynamic model of the center of mass of each rigid body. In a pioneering work [113], Karnopp and Rosenberg developed the Eulerian Junction Structure (EJS) using bond graph formalism that represents the dynamics of a rigid body in the body fixed frame. [39, p. 352].

3. Determine the *articulation points* on each body at which different interconnections can be modeled using joint structure that also needs to satisfy the kinematic structure. In this case, propulsion systems when considered independently have no translational degrees of freedom and only one rotational degree of freedom, leaving the other two rotational degrees of freedom constrained;

hence, it can be attached to the mainframe using a revolute joint model. The flow and effort vectors associated with the joint need to be derived and they need to be expressed in the privileged frame through appropriate coordinate transformation.

4. Construct the bond graph models of all joint types present in the quadrotor. Connect the transformed effort and flow vectors of each articulation point to the appropriate joint model through *multibonds* [114].

5. The final step is to incorporate additional dynamic systems to obtain a complete model of the quadrotor. Since the propellers are driven with a motor, a DC motor model is required to generate motion of the propeller.

In order to explain these steps in detail, we assume that the mainframe structure is symmetric and the static stator of the motor is part of the vehicle frame body. For simplicity of multi-body modeling the propellers are considered as flat bars and the out-runner (rotor of the motor) is a part of the propeller and attached underneath it. Figure 4.3 shows the coordinate system of each body and the articulation points (P_1 to P_4) in the mainframe. The body fixed frame in each body is assumed to be at the center of gravity of the respective bodies and aligned with the principal axes of the respective body. The z-axis is considered to be positive upward. Due to the double symmetry of the mainframe, it is logical to set the origin O at the intersection of the axes of the mainframe. The direction of the axes is chosen with the commonly used right hand convention. The body fixed frame $\mathbb{B} = (x,\ y,\ z)$ is considered to be moving with respect to the earth frame/inertial frame $\mathbb{I} = (X,\ Y,\ Z)$ and the body fixed frame $\mathbb{P} = (x_p,\ y_p,\ z_p)$ of the propeller is moving with respect to the vehicle frame body.

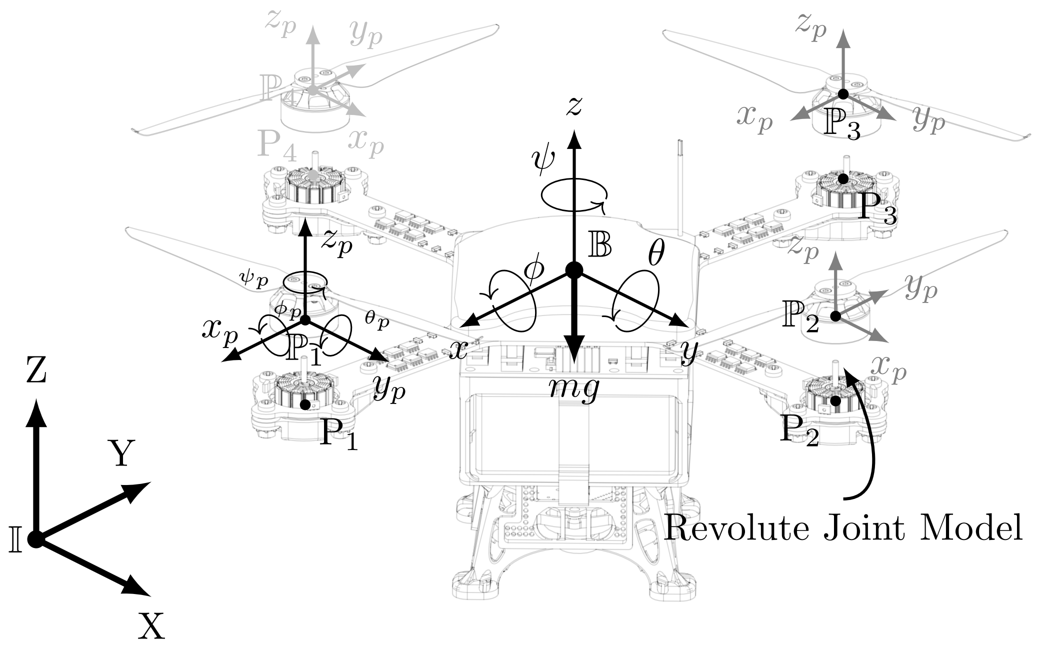


Figure 4.3: Coordinate system for multibody modeling of quadrotor.

4.3.1 Modeling a Rigid Body

The equation of motion of a rigid body with frame $\mathbb{B}$, under external forces and moments applied at the center of mass and expressed in the body frame, is developed in the Newton-Euler formalism [39, p. 352], which appears in equation (4.1) and (4.2) and in both their intrinsic way of representation and their tensorial counterparts. With respect to these body fixed coordinates, the rotational inertia properties remain invariant and the products of inertia are all zero. The first one represents the conservation of linear momentum, written as:

$$\sum \boldsymbol{F} = \frac{d\boldsymbol{p}}{dt} = \left.\frac{d\boldsymbol{p}}{dt}\right|_{rel} + \boldsymbol{\omega} \times \boldsymbol{p} = \left.\frac{d\boldsymbol{p}}{dt}\right|_{rel} + \epsilon_{ijk}\boldsymbol{\omega_j p_k}, \tag{4.1}$$

where $\boldsymbol{F}$, $\boldsymbol{\omega}$, and $\boldsymbol{p}$ represent external forces, angular velocity vectors and linear momentum vectors respectively; d/dt, $d/dt|_{rel}$, ϵ_{ijk} represent the derivative with respect

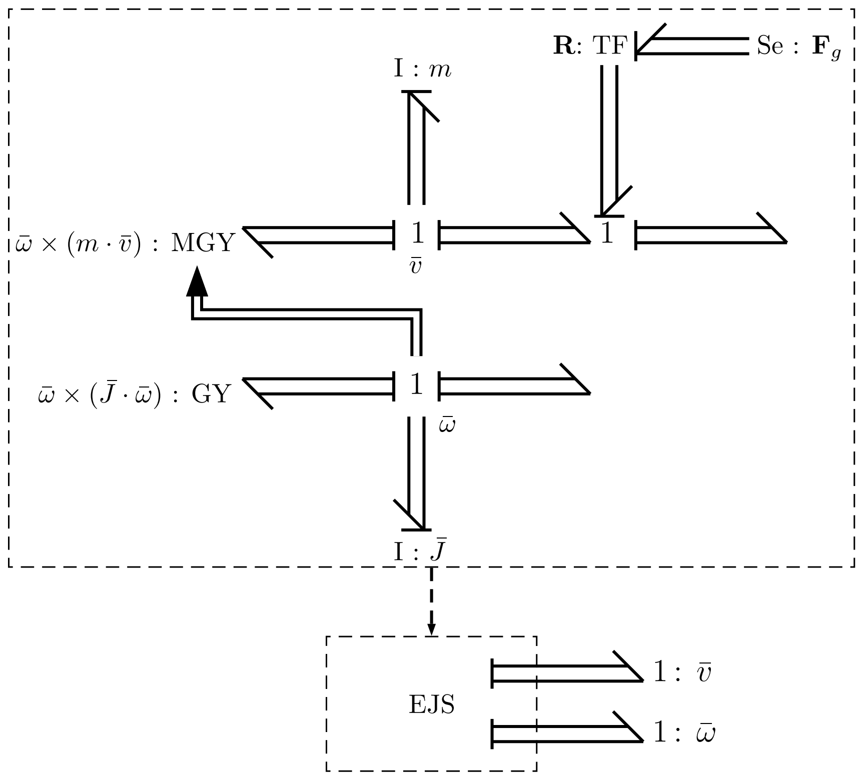


Figure 4.4: Bond Graph model of the Eulerian Junction Structure to represent rigid body dynamics in 3-D space.

to the inertial frame, the derivative with respect to the body attached frame. The cross product is expressed using Levi-Civita tensor notation. The second of the Euler equations sets up the conservation of angular momentum:

$$\sum \boldsymbol{M} = \frac{d\boldsymbol{h}}{dt} = \left.\frac{d\boldsymbol{h}}{dt}\right|_{rel} + \boldsymbol{\omega} \times \boldsymbol{h} = \left.\frac{d\boldsymbol{h}}{dt}\right|_{rel} + \epsilon_{ijk}\boldsymbol{\omega_j h_k}, \tag{4.2}$$

where $\boldsymbol{M}$ and $\boldsymbol{h}$ represent the external torque and the angular momentum vector. Complete bond graph representation of the 3-dimensional motion of a rigid body based on the Euler equations is shown with an EJS in Figure 4.4. Since three-dimensional dynamics of rigid bodies are of interest, vector bond graphs are preferred for their compactness and ease of implementation of vector and matrix relationships as presented by Bos in his Ph.D. thesis [115]. Three power multibonds are attached to the upper left 1-junction representing the center-of-mass speed-vector $\boldsymbol{v}$. The effort variable associated with the multibond pointing into the inertia $I : m$ is the first term in the right-hand side of equation (4.1); the effort of the multibond on the left is the second term of the right-hand side, while that of the multibond on the right is the term of the left-hand side. The latter represents the external forces acting on the center-of-mass point of the body and, thus, it is an input or external connection port of this module. An analog description can be given for the lower 1-junction concerning the rotational speed vector $\boldsymbol{\omega}$ and the torques in equation (4.2). Note the multi-signal link from the lower 1-junction to the MGY-element (Modulated Gyrator), necessary to implement the modulation of the second term on the right-hand side of equation (4.2) by the speed $\boldsymbol{\omega}$. The components of the inertia matrix $\boldsymbol{J}$ are the principal moments of inertia with respect to the principal axes of the rigid body. The force due to gravitation $\boldsymbol{F}_g$ is appropriately augmented using an effort source and a transformer (for coordinate transformation) as seen in the figure. The lower part of Figure 4.4 is the mask which will be used as a compact representation when connecting this model to others while modeling the complete system.

4.3.2 Modeling Transformation of Rotation

Since a body fixed frame is the privileged frame for this multi-body model development it is necessary to transform the gravitational force (prescribed in the fixed inertial frame) to the body fixed frame. Accordingly, Euler angles with the ZYX convention of Euler rotation are chosen to parameterize the rotation of the body fixed frame. Equation (4.3) presents the rotational transformations.

$$\begin{aligned} {}^{b}\boldsymbol{\omega} &= \boldsymbol{R}\,{}^{I}\boldsymbol{\omega}, \\ {}^{b}\boldsymbol{v} &= \boldsymbol{R}\,{}^{I}\boldsymbol{v}, \end{aligned} \tag{4.3}$$

where:

$$\boldsymbol{R}(\phi,x)=\begin{bmatrix}1&0&0\\0&\cos\phi&-\sin\phi\\0&\sin\phi&\cos\phi\end{bmatrix},\boldsymbol{R}(\theta,y)=\begin{bmatrix}\cos\theta&0&\sin\theta\\0&1&0\\-\sin\theta&0&\cos\theta\end{bmatrix},\boldsymbol{R}(\psi,z)=\begin{bmatrix}\cos\psi&-\sin\psi&0\\\sin\psi&\cos\psi&0\\0&0&1\end{bmatrix}$$

$$\boldsymbol{R}=\boldsymbol{R}(\psi,z)\ \boldsymbol{R}(\theta,y)\ R(\phi,x).$$

Here, $R = SO(3)$ is the rotation matrix representing ZYX Euler angles of yaw(ψ), pitch(θ), and roll(ϕ). The Euler angles $E = [\psi\ \theta\ \phi]^T$ are calculated from the angular velocity vector $\boldsymbol{\omega}$ of the body. The analytic expression between the angular rates $[\omega_x\ \omega_y\ \omega_z]^T$ and the corresponding time variant of angles presented in equation (4.4) can be derived from the definition of the ZYX convention [39].

$$\begin{bmatrix}\dot{\psi}\\\dot{\theta}\\\dot{\phi}\end{bmatrix}=\begin{bmatrix}0&\frac{\sin\phi}{\cos\theta}&\frac{\cos\phi}{\cos\theta}\\0&\cos\phi&-\sin\phi\\1&\sin\phi\tan\theta&\cos\phi\tan\theta\end{bmatrix}\begin{bmatrix}\omega_x\\\omega_y\\\omega_z\end{bmatrix}. \tag{4.4}$$

Although these transformation angles suffer from singularity at $\theta = \pm\pi/2$ known as 'Gimbal Lock', this does not affect UAVs in regular flight maneuvers. It may be noted that, in bond graph modeling when the effort variables are rotated from inertial frame to body frame, the flow variables are transformed back according to the law of conservation of energy. To estimate the effort, transposition of the rotation matrix is calculated within the transformer (TF).

4.3.3 Modeling Articulation Points

The port variables $P_{i,i=1\ldots4}$ in the proposed multi-body model (see Figure 4.3) are defined with respect to the center of mass of the mainframe body. Referring these port variables to the coordinates of articulation points with a four rotor assembly allows the application of forces and moments to the system. Let the position vector of articulation point i be denoted as $\boldsymbol{r}_i$ with respect to the mainframe body. Its linear

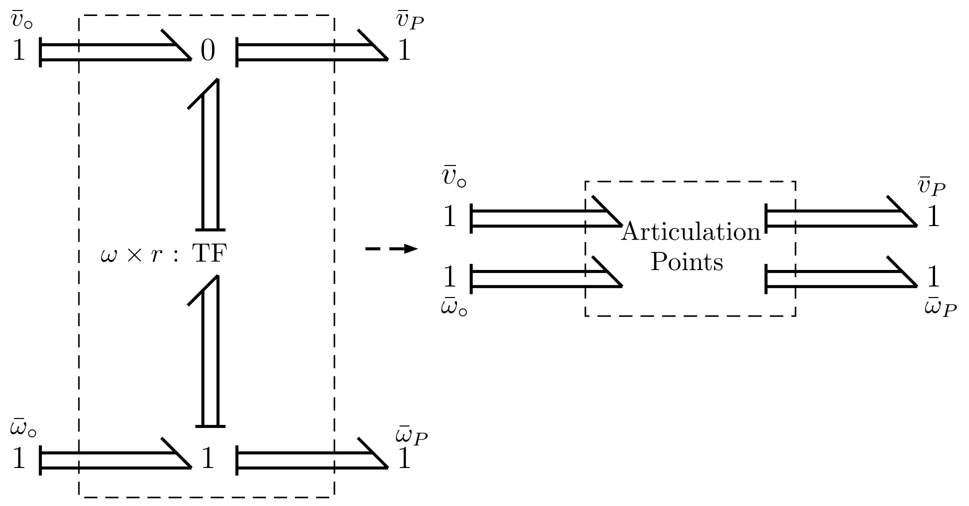


Figure 4.5: Bond graph representation of articulation points of a rigid body.

velocity is provided by:

$$\boldsymbol{v_P} = \boldsymbol{v}_\circ + \boldsymbol{\omega} \times \boldsymbol{r}_P. \tag{4.5}$$

Here, the linear velocity of the center of mass of the mainframe and the angular velocity of the body are denoted by $\boldsymbol{v}_\circ$ and $\boldsymbol{\omega}$ respectively. $\times\boldsymbol{r}$ is the skew symmetric matrix corresponding to the position vector $\boldsymbol{r}$.

Figure 4.5 shows the bond graph representation of the equation system representing how the port variables of two arbitrary points of the given spatial body transform each other using a zero junction and a transformer of a skew symmetric matrix.

4.3.4 Modeling Joint Dynamics

In order to allow the rotor-propeller assembly to spin on one axis while it is attached to one of the four articulation points a revolute joint model is developed based on joint models proposed by Zeid and Chung in [116]. In the case of the revolute joint, three translations and two rotational degrees of freedom are constrained, leaving only one rotation degree of freedom free. Due to this joint condition, only one angle changes between the connected bodies. Accordingly, modulated transformers are employed to

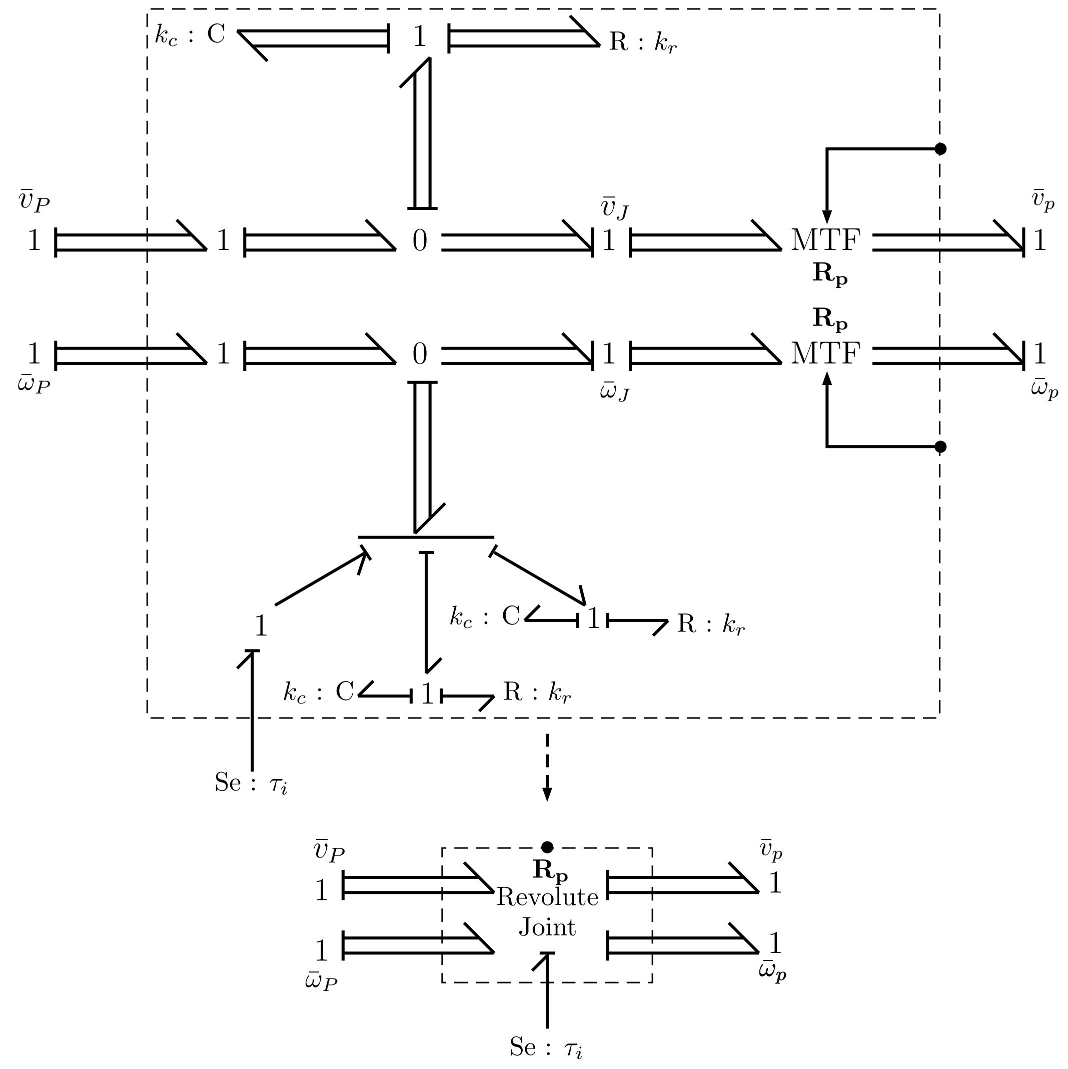


Figure 4.6: Bond Graph representation of the Revolute Joint.

transform the orientation of the propulsion system into the joints mounted on the articulation points on the body.

$$\begin{aligned} \boldsymbol{v_p} &= \boldsymbol{R}_p\, \boldsymbol{v_J}, \\ \boldsymbol{\omega_p} &= \boldsymbol{R}_p\, \boldsymbol{\omega_J}. \end{aligned} \tag{4.6}$$

The bond graph representation of the joint model is shown in Figure 4.6. At the joint point, constraints are assigned through stiff springs and dampers, i.e. introducing parasitic C and R elements instead of using ideal flow sources or zero effort source. This approach was originally developed by Karnopp and Rosenberg in [37] to eliminate derivative causality and algebraic loops, but here the rationale of using parasitic elements is different; namely, to resolve the causal conflict and to allow partial constraining of the multibonds. However, the approach introduces more elements (C and R), the parameter values of which need to be determined. The k_c and k_r parameters must be accurate enough to obtain approximately $v_P = v_J$, $\omega_{P,x} = \omega_{J,x}$, and $\omega_{P,y} = \omega_{J,y}$. The effort source Se creates a port for reactive torque generated by the motor-propeller assembly.

Typically, to approximate ideal constraints sufficiently, high stiffness (R) and damping (C) values are necessary, which add high frequency modes to the system, which in turn requires smaller integration time steps and slows down the simulation experiment.

4.3.5 Modeling Propulsion System

The propulsion system of the quadrotor comprises of a four motor-propeller assembly. For the simplicity of modeling, the propeller is assumed to be a flat bar and attached to the rotor of the motor. The rotor, along with a propeller, is assumed to be one rigid body connected to one end of the revolute joint, and the other end is connected to one of the articulation points of the quadrotor mainframe body. The static section of the motor (stator) is assumed to be part of the mainframe body and contributes to the mass and inertia of the mainframe. The motor-propeller system dynamics can be broken down into two domains: the three-dimensional rigid body dynamics of the rotor-propeller and the electromechanical dynamics that generate the relative rotational motion between the rotor and the stator through the revolute joint.

The bond graph model of the complete propulsion system is shown in Figure 4.7. Following the topology of the system, the rigid body dynamics of the propulsion

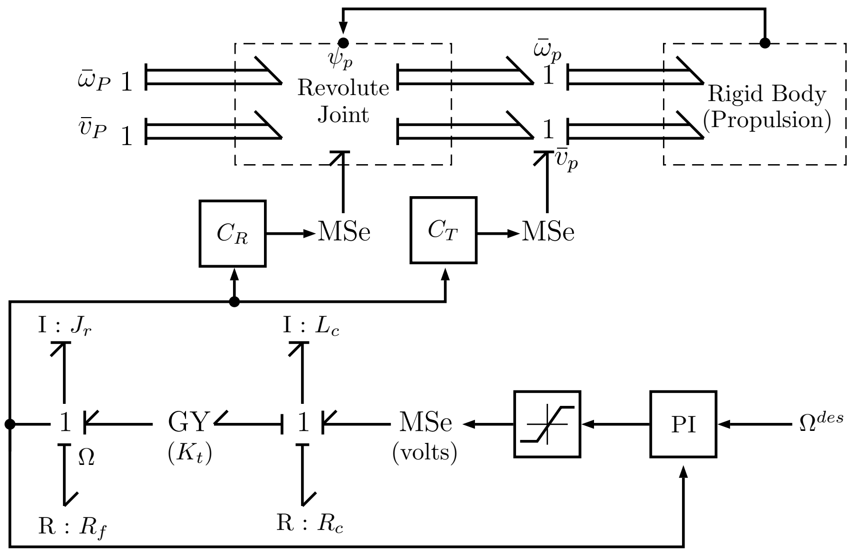


Figure 4.7: The propulsion unit.

system model can be assembled using a rigid body and a joint model that were previously introduced. In order to obtain an elaborate bond graph representation of the propulsion unit, a generic DC motor model [39] is incorporated with coil resistance (R_c), coil inductance (L_c), motor constant (k_t), mechanical inertia (J_r), and bearing friction (R_f). The generated torque is prescribed using a zero junction on the free axis of the revolute joint to model relative rotational motion. To maintain a desired speed in each actuation system (i.e., motor) a proportional-integral (PI) controller is used, that is governed by:

$$V_i = k_{p,m}(\Omega_i^{des} - \Omega_i) + k_{i,m} \int (\Omega_i^{des} - \Omega_i). \tag{4.7}$$

Output of the controller is saturated according to the supply voltage rating of the motor.

4.3.6 Aerodynamic Forces & Moments

The external forces on the mainframe body contributed by the propulsion unit are mainly the rotor thrust and the drag. The drag force is defined as [117]:

$$\begin{aligned} F_{drag} &= \frac{1}{2} C_d \rho A v^2 \\ &= C_D v^2, \end{aligned} \tag{4.8}$$

where C_d is the coefficient of drag, ρ is the density of medium (air), A is the frontal area of the body and v is the velocity. Considering all the constants involved in this equation we can combine the constant terms as C_D and define it as 'Drag Constant'. Therefore, the external force vector exerted on the center of mass of the mainframe body becomes:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} -\text{sign}(\dot{x})\ C_{D_x} \dot{x}^2 \\ -\text{sign}(\dot{y})\ C_{D_y} \dot{y}^2 \\ \sum_{i=1}^{4} T_i - \text{sign}(\dot{z})\ C_{D_z} \dot{z}^2 \end{bmatrix}. \tag{4.9}$$

Here, C_D has components in all three directions and T_i is the thrust generated by the four rotors that are spinning with a speed of Ω, and is given by [117]:

$$\begin{aligned} T &= \frac{\rho}{4}\ \Omega^2\ r^3\ a\ b\ c\ (\theta_t - \varphi_t) \\ &= C_T \Omega^2, \end{aligned} \tag{4.10}$$

where r is the radius of rotor, a is the constant blade lift curve slope, b is the number of blades, c is the blade chord, θ_t and φ_t are pitch at blade tip and inflow angle at the tip respectively. Similar to equation (4.8) we can combine the constant in equation 4.10 terms and define it as C_T. In the bond graph model the propeller generates this thrust through the modulated effort source (MSe) that is a function of the speed of the rotor and is governed by equation (4.10).

The rotational dynamics of a quadrotor consist of three angular motions, namely roll, pitch and yaw. Roll and pitch rotation are produced due to moments generated by the thrust difference of the rotors at the y-axis and x-axis respectively, which are acting at the articulation points in the mainframe body. However, yaw is produced due to reactive torque in the rotor. From equation (4.2) we can define:

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 0 & -C_T & 0 & C_T \\ -C_T & 0 & C_T & 0 \\ -C_R & C_R & -C_R & C_R \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}. \tag{4.11}$$

In the equation above, C_R $[Nm \cdot s^2]$ is the proportionality constant between motor speed Ω and motor reactive torque τ_i. The value of C_R can be estimated through identification of motor-propeller parameters as presented by Hossain and Krouglicof in [13]. However, the reactive torque can also be estimated using the formulation defined in [118]:

$$\tau_i = \lambda \Omega_i^2 + J_r \dot{\Omega}_i,$$

where λ is the reactive torque constant due to drag terms and J_r is the rotational inertia of the rotor. Obviously, only the dominant aerodynamic effects are modeled here, neglecting forces and moments exerted by drag of the propeller which can be extended easily by adding more external forces to their respective directional ports.

4.4 Complete Multi-body Model

Once the individual models of the bodies, the joints and propulsion units have been constructed, they are connected together according to the topology of the actual vehicle using appropriate joints at their respective kinematic structures. A high level bond graph model of the completed quadrotor helicopter is shown in Figure 4.8. Using modular modeling strategy, one of the arm models is created on the right hand side

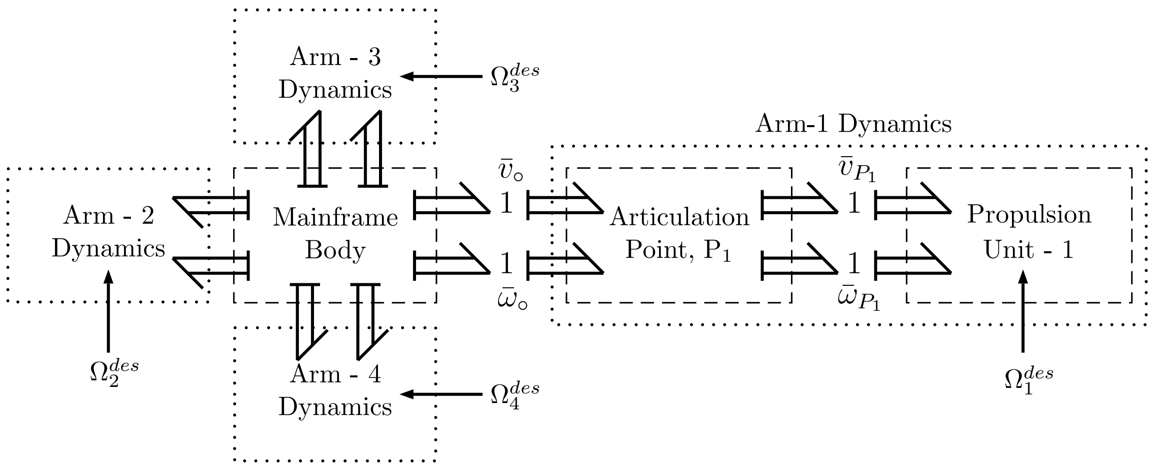


Figure 4.8: Complete multibody model of the quadrotor.

of the figure. Repeating the model for the other three propulsion units completes the quadrotor model. Note that the models for the quadrotor with different configurations or UAVs with a higher number of actuators (e.g., Octocopter) can be obtained simply by changing the connection port and/or re-utilizing the subsystems. This flexibility of model reuse greatly alleviates the possibility of model expansion while retaining the exactness of the model.

4.5 Closed Loop Control Model

The quadrotor UAV is an inherently unstable platform due to its six degrees of freedom with only four actuators. Stabilization is very important for such an under-actuated system to exploit its full potential. A simple PID based control system is developed to track attitude and altitude trajectories in $SO(3)$ that are close to the nominal hover state where roll and pitch angles are small. From the analysis of possible maneuvers presented in section 4.2, the nominal thrust at hover state must satisfy:

$$T_i = \frac{mg}{4},$$

and the motor speeds at hover condition are given by:

$$\Omega_h = \sqrt{\frac{mg}{4C_T}}.$$

Accordingly, the vector for desired rotor speeds can be written as a linear combination of four terms:

$$\begin{bmatrix} \Omega_1^{des} \\ \Omega_2^{des} \\ \Omega_3^{des} \\ \Omega_4^{des} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Omega_h + \Delta\Omega_{lift} \\ \Delta\Omega_\phi \\ \Delta\Omega_\theta \\ \Delta\Omega_\psi \end{bmatrix}, \tag{4.12}$$

where deviations from the nominal desired state vector are $\Delta\Omega_{lift}, \Delta\Omega_\phi, \Delta\Omega_\theta$, and $\Delta\Omega_\psi$. $\Delta\Omega_{lift}$ results in net force along the z axis of the propulsion system., while $\Delta\Omega_\phi, \Delta\Omega_\theta$, and $\Delta\Omega_\psi$ produce moments causing roll, pitch, and yaw respectively. This is similar to the approach described in [68]. Four PID controllers are used to formulate control laws for orientation and motion along the z axis, that take the form:

$$\begin{aligned} \Delta\Omega_\phi &= k_{p,\phi}(\phi^{des} - \phi) + k_{i,\phi}\int(\phi^{des} - \phi) + k_{d,\phi}(\dot{\phi}^{des} - \dot{\phi}), \\ \Delta\Omega_\theta &= k_{p,\theta}(\theta^{des} - \theta) + k_{i,\theta}\int(\theta^{des} - \theta) + k_{d,\theta}(\dot{\theta}^{des} - \dot{\theta}), \\ \Delta\Omega_\psi &= k_{p,\psi}(\psi^{des} - \psi) + k_{i,\psi}\int(\psi^{des} - \psi) + k_{d,\psi}(\dot{\psi}^{des} - \dot{\psi}), \\ \Delta\Omega_{lift} &= k_{p,z}(z^{des} - z) + k_{i,z}\int(z^{des} - z) + k_{d,z}(\dot{z}^{des} - \dot{z}). \end{aligned} \tag{4.13}$$

By varying the speed of the four rotors one can achieve desired attitude and altitude of the system and hence a stabilized platform. Since translational motion along the x and y axes of the mainframe is a function of pitch (θ) and roll (ϕ) angles of the body, deviation in pitch and roll angles from a stable condition ($\theta = \phi = 0°$) can result in motion along the x and y axis respectively. However, translational motion does not contribute directly to the stability of the system; therefore, it is left for future expansion of this work.

Table 4.1: Inertia parameters of the rigid bodies

Body	Mass (Kg)	Principal Moments of Inertia (Kg · m^2)		
		I_x	I_y	I_z
Mainframe	0.30155	6.4E-4	6.4E-4	1.22E-3
Rotor-Propeller	0.011721	6.4061E-7	6.2732E-6	3.008E-5

4.6 Identification of Parameters

In order to join the theoretical and construction branches of the quadrotor development process, the model constants are identified. Once the constant parameters are obtained, numerical simulations of both the open loop bond graph model and the closed loop controlled model will be performed. The constant parameters for the simulation model include: inertial values of the quadrotor frame, inertial values of the motor-propeller assembly, and the actuator and aerodynamics parameters.

4.6.1 Inertial Parameters

For the sake of simplifying the identification process, inertial parameters are approximated using the Solidworks 3-D modeling application. Detailed 3-D CAD models are developed and the material properties are prescribed to obtain mass and the principal inertia values of the respective rigid body. Then the estimated mass parameter is compared with the actual mass of the component to verify the accuracy of the estimation. In this study, inertial parameters are identified for two primary bodies: the mainframe body and the rotor-propeller assembly. The articulation points P$_1$ through P$_4$ are assumed to be on the $x-y$ plane of the mainframe body and located at 12.7 cm distance in both the positive and negative directions of the x and y axis. Table 4.1 presents the inertial constants used to perform the simulation experiment.

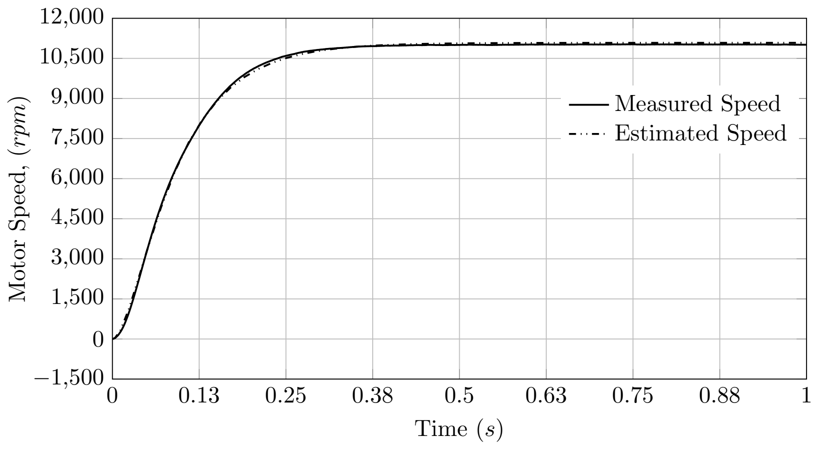


Figure 4.9: Comparison of measured speed response and estimated speed response from a BLDC motor (MN1806) when excited with a step input of 7.4 V.

4.6.2 Propulsion System Parameters

DC Motor parameters are identified using the Matlab/Simulink Parameter Estimation Toolbox. A DC motor model is developed in Simulink to support the estimation task. The basis for the identification is the measured speed response for a given step input of rated voltage. The speed response for an off the shelf BLDC motor (MN1806) is used for this study. The response is obtained for a step input of 7.4 volts. Using measured coil resistance (R_c), speed response for step input, and the motor model, the unknown parameters are estimated; i.e., coil inductance (L_c), rotor inertia (J_r), damping resistance (R_f), and motor constant (K_t). The estimated response generated using the approximated parameters closely matches the measured response seen in Figure 4.9.

The aerodynamic constants (thrust factor and the proportionality constant between thrust and the reactive torque) are identified using a propeller calibration rig developed by Hossain and Krouglicof in [13]. Table 4.2 presents the actuator parameters used in the simulation study.

Table 4.2: Parameters of the propulsion unit

Parameters		Value	Unit
Motor Constant	k_t	5.7094E-3	N/amp
Coil Resistance	R_c	0.37	Ω
Coil Inductance	L_c	7.21E-3	H
Rotor Inertia	J_r	9.37E-6	Kg-m^2
Damping Resistance	R_f	1.0098E-5	N-m-s/rad
Thrust Constant	C_T	1.27E-6	N-s^2
Torque Constant	C_R	1.48e-8E-8	N-m-s^2
Drag Coefficient(assumed)	C_D	$[0.2\ 0.2\ 0.4]^T$	N-m-s^2
Proportional Constant	$k_{p,m}$	300.0	-
Integral Constant	$k_{i,m}$	5.0	-

4.7 Simulation Experiment

The simulation experiment was conducted in two steps. First an open loop experiment was performed to validate the accuracy and functionality of the model and then the loop was closed using the controllers and the control performances were evaluated. For numerical integration of the simulation, the Modified Backward Differentiation Formula (MBDF) method was chosen with a tolerance of 1E-10. This high tolerance value is used due to the high stiffness and damping value of the parasitic elements. Here, the k_c and k_r values are tuned to 1E-2 and 20 respectively.

4.7.1 Freefall and Hover Test

A simple freefall experiment was conducted to validate the kinematic and dynamic constraints of the model. Providing no thrust to the quadrotor causes it to fall down due to gravity at a rate of 9.81 ms^{-1} as seen in the simulation results presented in Figure 4.10. Note that the drag coefficient is assumed zero for the freefall test. On the other hand, for lifting the quadrotor, it is obvious that a minimum combination thrust equal to (mass of mainframe body + 4×mass of each propulsion unit)×9.81 = 3.418 N is required, which leads to a motor speed of Ω_h = 820.26 rad/s from each rotor.

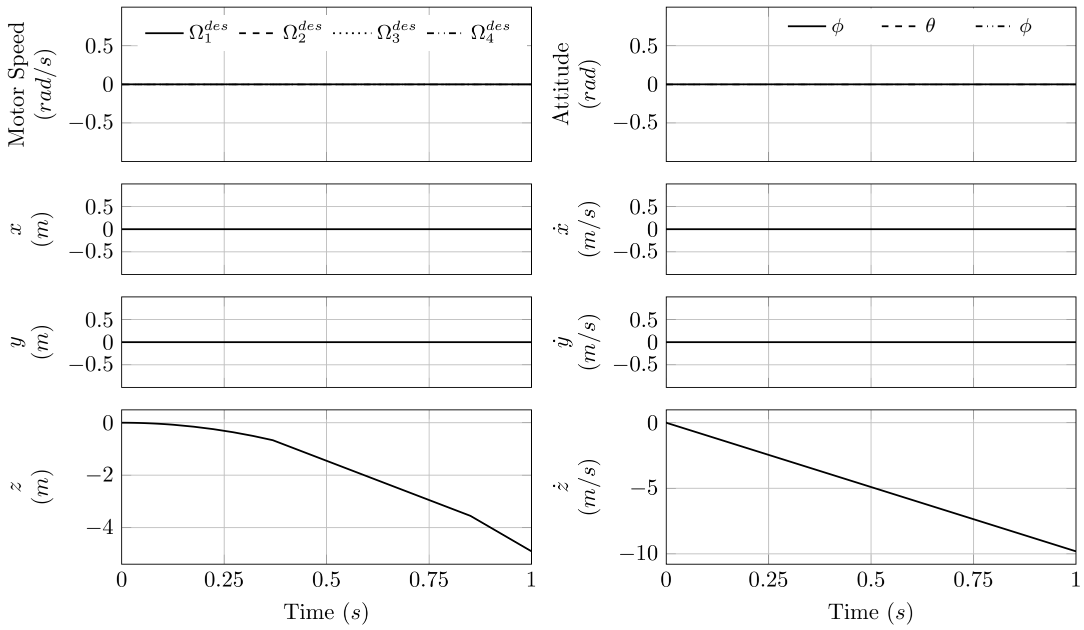


Figure 4.10: Open loop simulation results of free-fall. Motors are deactivated to simulate zero speed from the propulsion unit.

Simulation results presented in Figure 4.11 demonstrate the hovering phenomenon. Since the motors take a short period to attain the desired rpm, the quadrotor initially starts descending. However, as soon as the motors starts operating at the hovering speed, the platform starts hovering. Subsequently, velocity of climb/descend goes to zero, which is in agreement with the theory. During this test, front and back propellers spin in a clockwise direction and the other two spin in a counter clockwise direction, producing zero net torque, which leads to no angular motion (yaw) about the z axis of the mainframe body. Also, there is no pitch or roll motion, as the opposite rotors are spinning at the same speed to counterbalance the moment about x and y axes of the mainframe. The basic free-fall and hover condition validates the basic functionality of the quadrotor model.

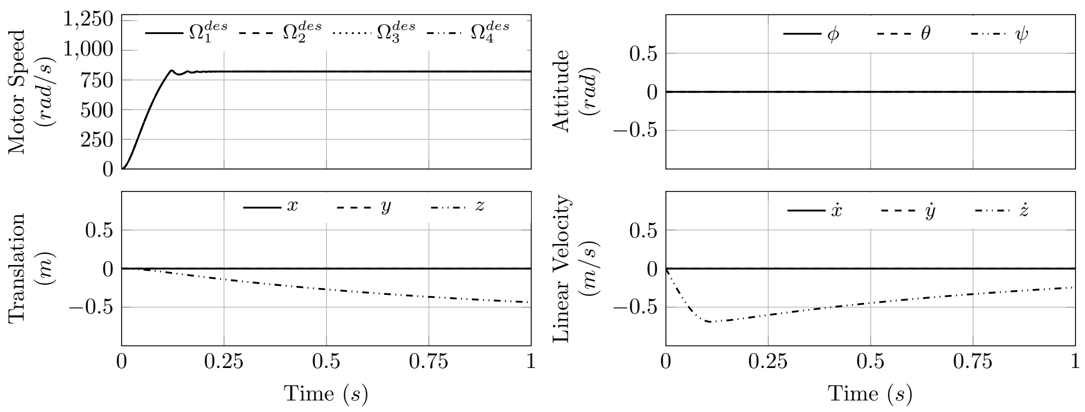


Figure 4.11: Open loop simulation result for motor speed equal to $\Omega_h = 820.26$ rad/s that balances the system in 3-D space.

4.7.2 Analysis of Possible Maneuvers

Once the validation was completed, simulation experiments were conducted to evaluate the theoretical maneuvers. The simulation demonstrates the flight maneuver which satisfies the theoretical trajectories that the quadrotor is supposed to perform at certain combinations of the rotor thrust. Figure 4.12 shows the simulation result of climb action of the quadrotor. In this simulation experiment all four rotors were spun at 1032.98 rad/s, an increase of 212.72 rad/s ($\Delta_{lift} = 212.72$) when compared to the required speed to produce enough thrust to lift its own weight. This leads to a vertical motion of the quadrotor. Similar to the hover condition, there is no angular motion observed in this condition as all four rotors are running at the same speed (rpm). Because translational motion is achieved through deviation in an angular position from the zero condition, there is no translational motion observed in this maneuver.

In order to simulate the roll motion (rotation about the x axis), the right and left rotor are set at 866.98 rad/s and 911.23 rad/s while the other two rotors are operated at 889.36 rad/s. The speeds are chosen carefully to cancel the net torque on the z axis. For example, in this case, the sum of torque generated by the front

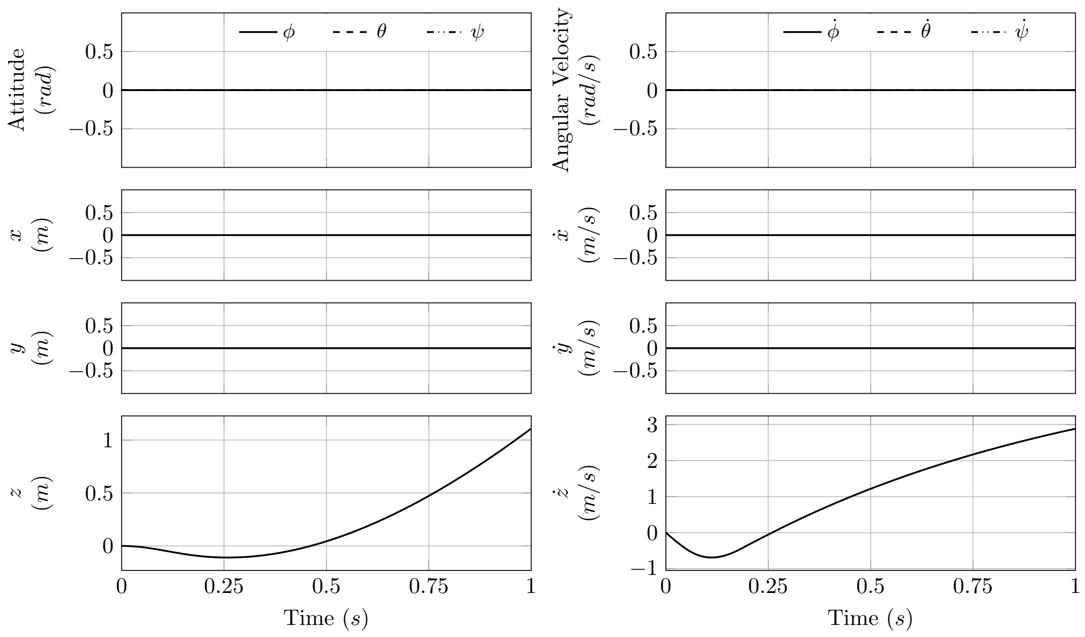


Figure 4.12: Simulation results of open loop climb/lift action. Motors are operated at $\Omega_h + \Delta_{lift} = 1032.98$ rad/s wherein $\Delta_{lift} = 212.72$ rad/s

and back rotor is equal to the total amount of torque generated by the right and left rotor. Simulation results presented in Figure 4.13 show the roll motion of the quadrotor. Motion along the y and z axes suggests that the quadrotor is falling while moving along the negative y direction. This is obvious in the case of rotation about the x axis (roll) in absence of a closed loop controller. The oscillations observed in the angular velocity response can be originated due to the parasitic elements used in construction of the joint model. This can be minimized thorough further tuning of the parasitic constants. The pitch motion presented in Figure 4.14 is identical to the roll motion. Instead of creating thrust difference between the right and left motor, here the front motor and back motor are set to operate at 866.98 rad/s and 911.23 rad/s respectively, while both front and back motors are operated at 889.36 rad/s. Unlike the roll motion, here the quadrotor moves along the positive x axis while it falls down. This is evident, due to the rotation about the y axis (pitch) only.

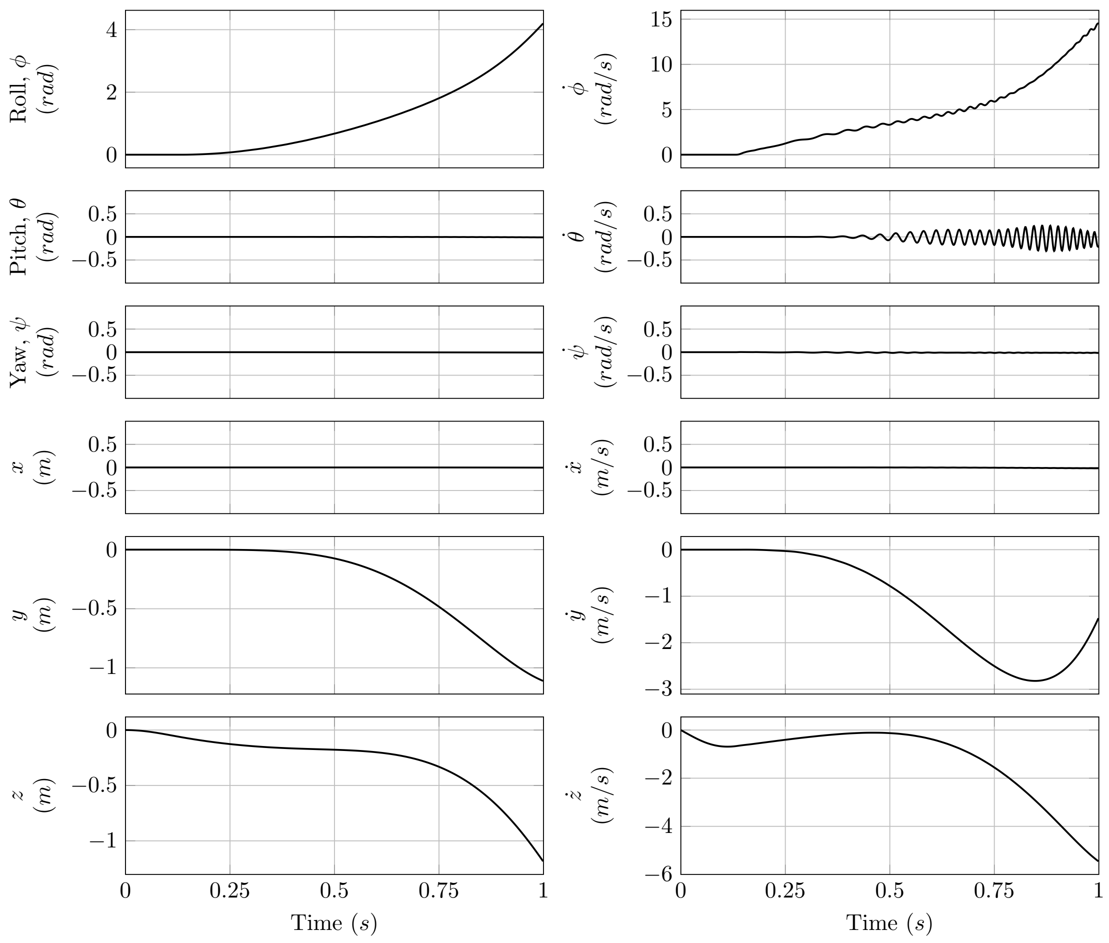


Figure 4.13: Simulation results of open loop roll motion. Left and right motors are running at 911.23 rad/s and 866.98 rad/s rpm respectively. Both front and back motors are running at 889.36 rad/s. The equivalent thrusts added to the hovering conditions are 0.2N, 0.1N, 0.15N, and 0.15N respectively.

Finally, to generate the yaw motion (rotation about the z axis), both the right and left motor are operated at 932.58 rad/s and the other two rotors are operated at 689.93 rad/s. The speed setpoints are chosen to produce zero net torque about the z axis, while also producing the necessary thrust to lift its own weight. This is done by increasing the lift in the right and left rotor and decreasing the same amount of lift from the front and back rotor. The result is only yaw motion with no translation. Simulation results in Figure 4.15 present the yaw maneuver of the quadrotor. As the

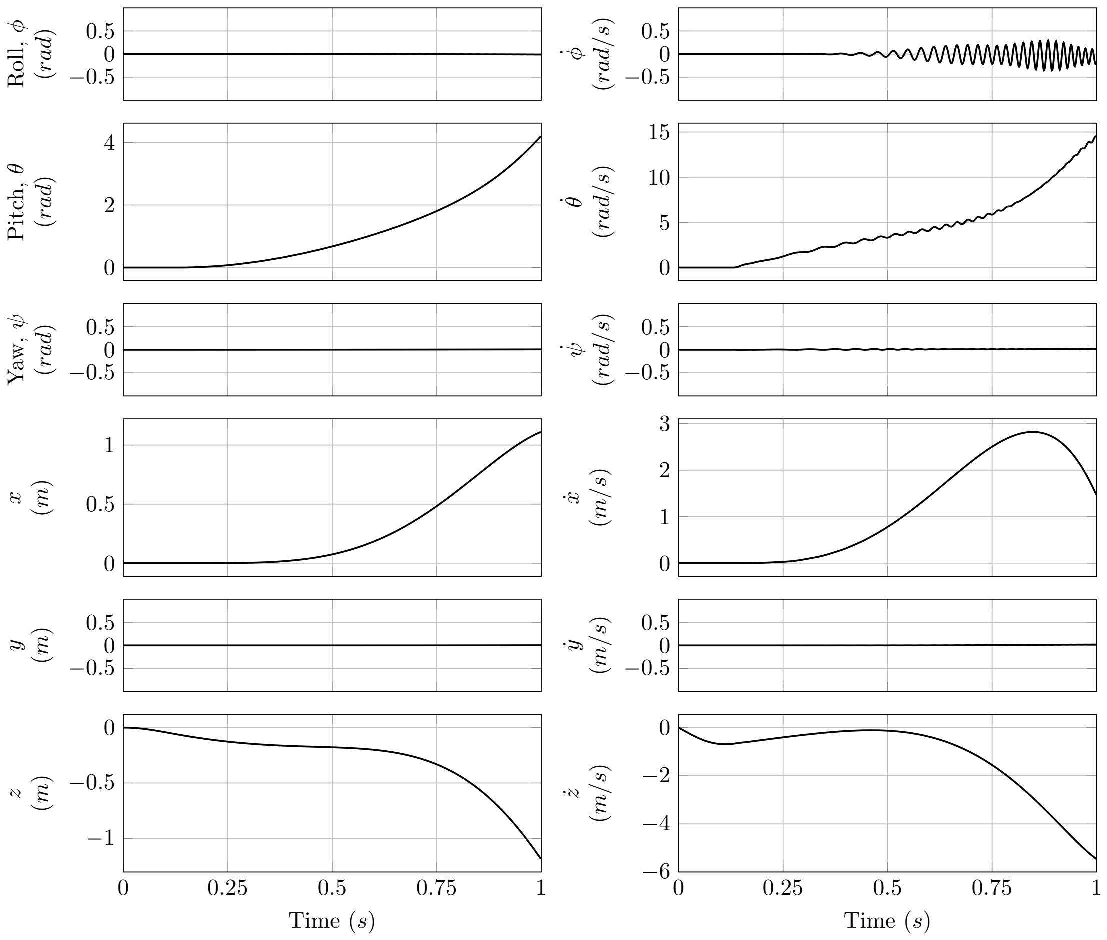


Figure 4.14: Simulation results of open loop pitch motion. Back and Front motors are running at 911.23 rad/s and 866.98 rad/s respectively. Both right and left motors are running at 889.36 rad/s. The equivalent thrusts added to the hovering conditions are 0.2N, 0.1N, 0.15N, and 0.15N respectively.

right and left rotor are rotating clockwise with a higher magnitude of speed compared to the front and back rotor, the reactive torque is dominated by the right and left rotors' rotational direction. Thus the motion of the body is in a counterclockwise direction which creates the positive yaw as seen in the simulation results. Similar to hover condition, a falling tendency of the vehicle due to lag in motor dynamics is also observed at the beginning of the response, which is recovered as soon as the motors reached to its desired speed.

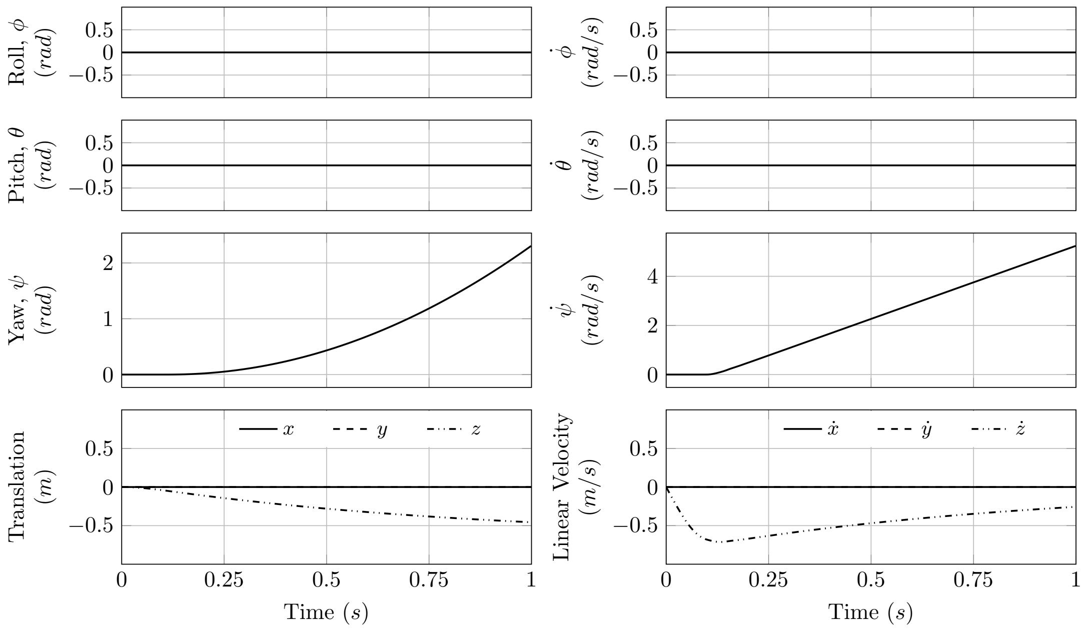


Figure 4.15: Simulation results of open loop yaw motion. Both left and right motors are running at 932.58 rad/s while the front and back rotors are running at 689.93 rad/s. The equivalent thrusts added to the hovering conditions are +0.25N, +0.25N, -0.25N, and -0.25N respectively.

Table 4.3: PID Controller gains used in the simulation

Parameter	Notation	Motion			
		Roll (ϕ)	Pitch (θ)	Yaw (ψ)	Altitude (z)
Proportional Gain	k_p	0.1	0.1	0.05	7.0
Integral Gain	k_i	0.01	0.01	0.0	0.25
Derivative Gain	k_d	0.04	0.04	0.01	2.5

4.7.3 Simulation of the Controlled Response

The open loop responses confirm that the system is highly unstable and coupled. Accordingly, the proposed PID control laws for attitude and altitude stabilization are formulated using the equation submodel feature of the '20-Sim' application. Table 4.3 presents the controller gains used for this simulation experiment. Here the initial conditions were chosen as $\phi = \theta = 15°$. All other initial conditions were set to zero except altitude, which is set to 1 m. Figure 4.16 shows how the quadrotor reaches a stabilized condition (no angular motion) from the perturbed orientation while maintaining an altitude of 1.0 m as prescribed in the controller setpoint. Since the linear motion along the x and y axes is a function of pitch and roll motion, if angular motions are stabilized, the system's linear motions along x and y are also stabilized. The closed loop controller can also maintain desired orientation of the body from a stable position which can be further utilized for controlling the position vector of the quadrotor. There is a steady state error observed in the roll response, this could be improved through further tuning of the controllers. Since the focus of this study is multibody modeling and control law development, optimal tuning of the model is deferred to future work of this study. The execution time for the experimental maneuver is provided by the corresponding largest settling time of the controlled responses; i.e., when motion in each axis reaches a steady state. It cannot be denied that the settling time and the controller performances presented here are not absolute measures of performance, since the gains of the PID controller greatly influence these quantities. However, in this case gains are tuned so that the stabilization responses asymptotically reach their respective steady states.

4.8 Conclusion

The chapter proposes a multi-body simulation approach and the stabilization control law for aerial robotic platforms using bond graph representation. The study demonstrates that a combination of bond graph and multi-body modeling approach facilitates modeling of the complex and coupled dynamics of robotics systems with greater exactness, as well as extension or simplification of the system model or its controller. For this purpose a number of ideas from previous work were amalgamated in the form of an algorithmic procedure for synthesizing a closed loop multi-body dynamic

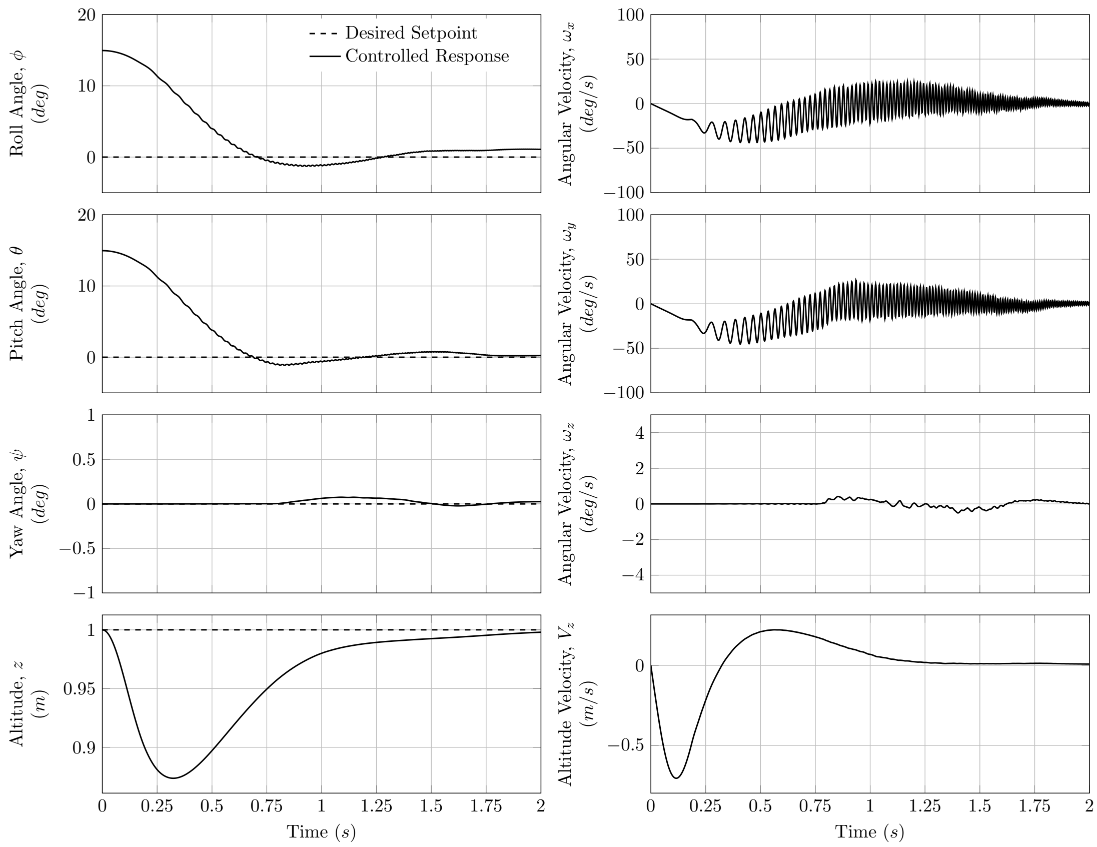


Figure 4.16: Attitude and altitude control response of the quadrotor stabilizing from a perturbed orientation of $\phi = \theta = 15°$ and $\psi = 0°$. Altitude is maintained at 1 m during the stabilization maneuver.

.

model. Subsequently the constructed multi-body dynamic models were simulated using parameters identified from the real components. Simulation study confirmed the theoretical maneuvers and the controlled behaviour of the quadrotor vehicle.

As the quadrotor is a mechatronic system, involvement of a different energy domain is a must. That is why selection of the modeling language is a significant decision if future extension is expected. Future model extensions include higher-fidelity aerodynamics contributed by the propellers. Possible extensions to the controller might involve control of position in 3-D space that may lead the model to be even more accurate and practical.

Chapter 5

FPGA Based BLDC Motor Speed Controller for UAV Platform

5.1 Introduction

The actuation system of a quadrotor consists of four motor-propeller assemblies. The function of the motors is to convert electrical energy into mechanical energy. Considering the ready availability of electricity and the wide use of mechanical energy, it is no surprise that electric motors are widely used. In the case of the quadrotor, the mechanical energy produced by its motors is used to propel the blades to generate required thrust. When the selection of the motor is a focus - one that promises better performance and efficiency is worth a closer look. Such is the case with BLDC motors.

BLDC motors are widely used in the UAV industry because of their high efficiency, low electromagnetic interference, high mechanical reliability, and most importantly, their wide range of speed. The commutation based operation and the low inertia of the BLDC motor make it more responsive and thereby more attractive for UAV applications. However, commutation and speed control of the motor are complex in terms of algorithms, required power electronics [42], and digital hardware based implementations [44].

The motivation of this chapter is to design and develop a flexible, compact, and fast reacting (i.e., high update rate) motor control system for the quadrotor that is driven by single FPGA hardware. In contrast, off-the-shelf motor speed controllers are based on the application of specific microcontrollers [119]. Microcontrollers, which are specified for motor control, have limited capacity in terms of driving channels and position feedback input channels. Due to the limited resources, a finite number of motors could be controlled with this architecture. In the case of the quadrotor (or multi-rotor), where multiple motors are controlled along with sensor data processing (e.g., image data processing, inertial data processing) this can result in hardware resource limitation. Moreover, when multiple sensors and actuators, including a camera are operated with the same microcontroller, the speed and performance of the processor can deteriorate to the point where the performance of the overall flying system is compromised. This happens primarily due to the sequential operation of microcontrollers and limited number of interrupts built into a microcontroller. This is resolved either by interfacing multiple microcontrollers or integrating a microcontroller with an increased number of interrupts and other I/O pins. However, both solutions are not effective in terms of cost and real estate available on an aerial vehicle. FPGA hardware implementation, on the other hand, provides solutions to the problems in a single chip and is conducive to the SWaP constraints faced by a UAV system.

In recent years, FPGA-based motor control has started to draw the attention of researchers [43, 119, 44]. Features of the FPGAs, which are the simplicity, programmability, short design cycle, fast time to market, low power consumption, and high density, enhance the application of FPGAs in the motor control field [44]. FPGA based motor drivers are also less complex and more reliable, flexible and adaptable when compared to DSP (Digital Signal Processor) and microcontroller based implementations.

In order to develop the commutation algorithm, hall sensors are used that are mounted around the periphery of the motor. This is called sensored control as opposed to sensor-less control where the zero crossing back-EMF of the three motor phases are sensed using a current sensor. Though sensor-less commutation and control involve fewer hardware components, the efficiency of the sensor-less approach degrades at low speed operation when the back-EMF generated by three phases is insufficient to produce enough current for the current sensors to accurately detect its zero crossing property. Moreover, implementation of sensor-less commutation is complex compared to sensored commutation and therefore only adopted in situations where sensored commutation is not suitable (e.g., if a motor is submerged in fluid, oil or water). Accordingly, for a wide range of speed as well as for simplicity of implementation, hall sensor based commutation is chosen in this study.

The digital speed control system is developed using the common control principle for a permanent magnet BLDC motor, which is pulse width modulated (PWM) current control. It is based on the assumption of a linear relationship between the phase current and the torque; same as the brushed DC motor [120]. Thus, by adjusting the PWM duty ratio, the phase current is controlled and accordingly the electromagnetic torque is generated to achieve the required rotational speed. In order to control the speed using closed loop control technology a classic PID control technology is chosen in this study. To estimate the speed feedback information for the controller, the frequency and width of the hall sensor signals are used. However, hall sensor signals are noisy and the width of the sensor signal varies with speed. To overcome the inconsistency of the accuracy of speed measurement a speed observer is proposed in this study. Both the speed observer and the closed loop controller are implemented in the FPGA hardware.

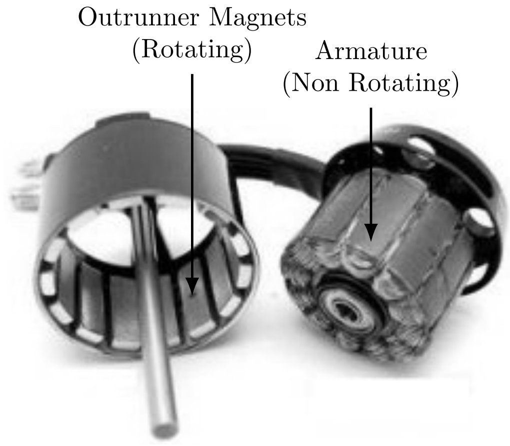
To begin the development of the commutation algorithm it is essential to study the construction and operation principle of the BLDC motor. Accordingly, Section 5.2

presents a review of the construction of a BLDC motor and in the following section the commutation algorithm is developed using hall sensor signals as the basis for commutation. In Section 5.4 a numerical model of the 3-phase motor is then developed for control law development and tuning of the controller. The discrete hardware modules are developed in Section 5.5, which include the discrete formulation of a speed measurement algorithm for hardware implementation. The closed loop controller is developed and implemented in the consecutive section. Finally the performance of commutation, performance of the speed control technique and the closed loop speed controller are evaluated in Section 5.6. A brief discussion is presented in Section 5.7 followed by concluding remarks in Section 5.8.

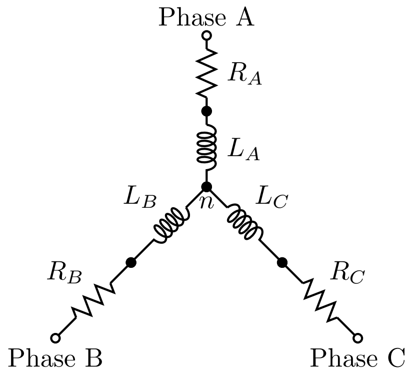
5.2 Brushless DC Motor

A BLDC motor is similar to a DC motor, except that in the case of the BLDC motor, the permanent magnets are attached to the rotor and coils remain stationary. Figure 5.1(a) shows a simplified illustration of BLDC motor construction. The coil windings are electrically separated from each other allowing them to turn on and off in a sequence that creates a rotating magnetic field. This sequential excitation of the coils to achieve proper interaction between rotating magnets is called commutation. The coils come in single phase, 2-phase, and 3-phase. Armature of the 3-phase motor can be modeled as an R-L circuit as presented by the circuit topology in Figure 5.1(b). Here all three electromagnetic circuits are connected to a common point n. Each of the lines in the circuit are called phases of the motor. Here A, B, and C are the three phases of the motor.

The motor is also manufactured as an in-runner or out-runner. The one presented in Figure 5.1(a) is an out-runner, wherein the rotor that holds the magnets is at the peripheral side of the motor and the wire coiled stator poles are at the center of the



(a) Internal Components of the motor.



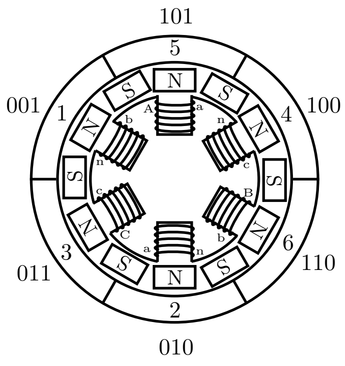
(b) Circuit Topology of a the motor.

Figure 5.1: 3-Phase Brushless DC motor (Out-runner).

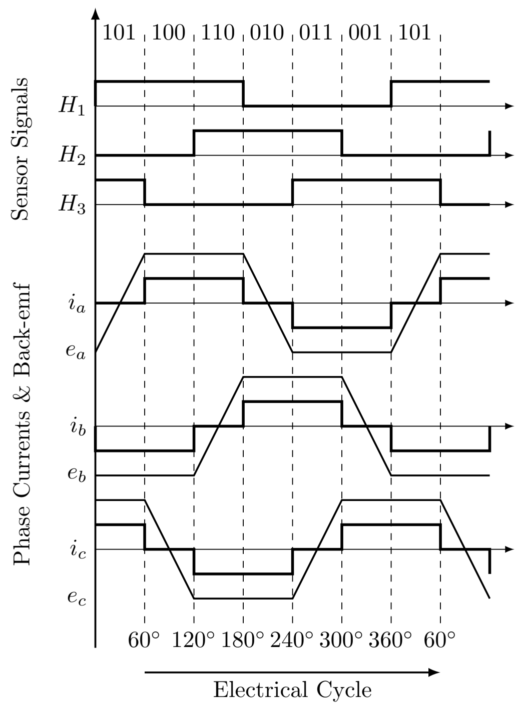
motor. The spinning shaft is attached to the out-runner. One significant advantage of this arrangement is that the commutator does not carry current to the rotor, which eliminates the brushes. However, it is still necessary to know the rotor position so that excitation of the stator field always leads the permanent magnet field to produce torque [121]. Position of the rotor is determined based on state (north pole or south pole) of the outrunner magnets, which is sensed by the hall-effect sensors.

5.3 3-Phase BLDC Motor Commutation

The commutation algorithm decodes the sensor signals and energizes the appropriate stator windings. The hall effect sensors generate binary signals: whenever the rotor magnetic poles pass near the hall sensors, they give a high signal, indicating the north pole is passing near the sensors, and a low signal indicates the presence of a south pole. Digital high output refers to 180 electrical degrees of electrical rotation, and low output to the other 180° electrical rotation. The three sensors are offset from each other by 60 electrical degrees so that each sensor output is in alignment with one of



(a) Definition of hall sensor position.



(b) Current & Back-EMF Profiles during operation.

Figure 5.2: Hall sensor signals corresponding to the driving sequence.

the electromagnetic circuits [44].

By sensing three hall sensors, a 3-bit code can be obtained with values ranging from 1 through 6. Each code value represents a sector on which the rotor is presently located. State '0' and '7' are therefore invalid states. Each code value provides information on which windings need to be excited to turn the rotor. Figure 5.2(a) shows a conceptual drawing for estimating rotor position based on the hall sensor. An associated timing diagram is presented in Figure 5.2(b) the shows the hall-effect sensor outputs and their corresponding motor phase current. The binary codes on the top row of the figure correspond to one of the six states of the commutation phase and the bottom row represents the incremental electrical cycle during rotation. Observe that the three sensors' output overlap in such a way as to create the six

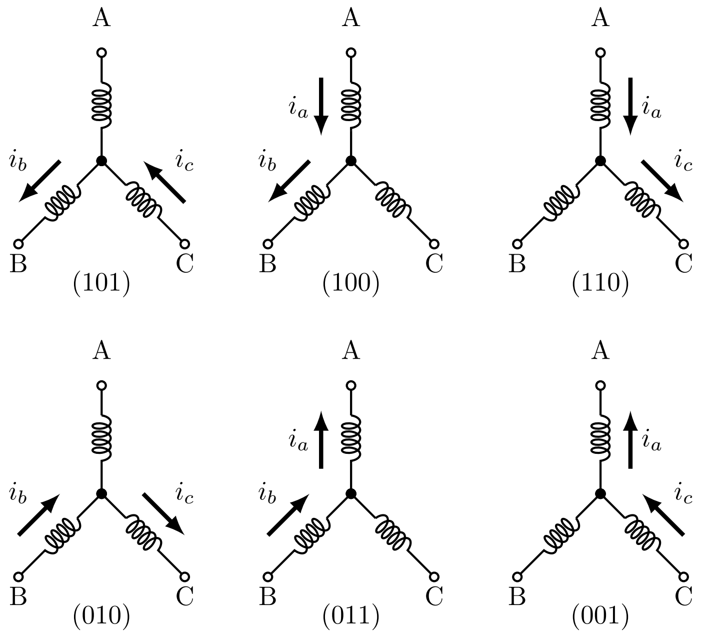


Figure 5.3: Armature coil excitation sequence based on the hall sensor code.

unique three-bit output codes corresponding to each state of the drive phases.

Motor commutation is organized according to a trapezoidal back-EMF profile [46] wherein the commutation is defined in a way that treats a three-phase BLDC motor as if it were a DC motor having just two phases at any instant of time. During the commutation operation, any two of the three phases are energized based on the position of the rotor. Each of the drive states consists of one motor terminal driven high, one motor terminal driven low and one motor terminal floating [122]. Figure 5.2(b) shows the direction of current flow in the phases while the motor is in forward motion (i.e., incrementing electrical cycle). Figure 5.3 presents the phase combination and the commutation sequence responsible for producing the flow of current. A simplified one and a half H-bridge based drive circuit is developed using power electronics (i.e., MOSFETs) to commutate motor coils according to the hall sensor code. The circuit is presented in Figure 5.4.

If the high side switch (MOSFET) of phase A (A_H) is activated, current flows

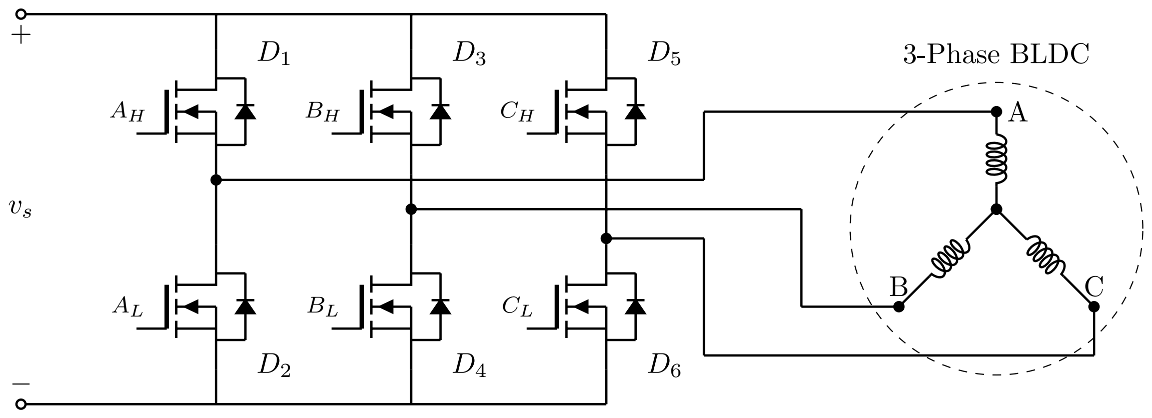


Figure 5.4: A simplified electronic drive system for three-phase BLDC motor.

Table 5.1: States of hall sensor and MOSFET switches in clock-wise (CW)

State	Hall 1	Hall 2	Hall 3	A High	A Low	B High	B Low	C High	C Low
1	1	0	1	0	0	0	1	1	0
2	1	0	0	1	0	0	1	0	0
3	1	1	0	1	0	0	0	0	1
4	0	1	0	0	0	1	0	0	1
5	0	1	1	0	1	1	0	0	0
6	0	0	1	0	1	0	0	1	0

into phase A, and when the low side switch of phase A (A_L) is activated, current flows out of phase A to the ground. Table 5.1 and Table 5.2 show the drive state of inverter switches corresponding to the position of the rotor for both clock-wise (CW) and counter clock-wise (CCW) rotation respectively. Since the three hall sensors are spaced 60 electrical degrees apart, they will change their state after every 60 electrical degrees. Given this, it takes six steps to complete one revolution of the electrical cycle. However, one electrical cycle may not complete one mechanical rotation of the rotor. The relation between mechanical rotation and electrical cycle depends on the number of poles in the motor. For one pole pair, one electrical cycle is completed. Accordingly, the number of electrical cycles per rotation is equal to the number of pole pairs present

Table 5.2: States of hall sensor and MOSFET switches in counter clock-wise (CCW)

State	Hall 1	Hall 2	Hall 3	A High	A Low	B High	B Low	C High	C Low
1	1	0	1	0	0	1	0	0	1
2	1	0	0	0	1	1	0	0	0
3	1	1	0	0	1	0	0	1	0
4	0	1	0	0	0	0	1	1	0
5	0	1	1	1	0	0	1	0	0
6	0	0	1	1	0	0	0	0	1

in the rotor. So, at every 60 electrical degrees, the phase current switching needs to be updated to move to the next pole pair.

5.4 Mathematical Framework

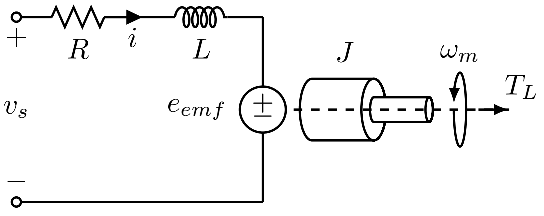
The BLDC motor is a DC motor with three phases working in a synchronous fashion. To review the mathematical model of a BLDC motor, a brief model of the DC motor is in order [123]. The topology of a simple DC motor is shown is Figure 5.5(a). R and L are the armature resistance and inductance respectively, v_s is the supply voltage, i is the armature current and e is the back-emf. The simple $L - R$ circuit can be described as:

$$v_s = Ri + L\frac{di}{dt} + e. \tag{5.1}$$

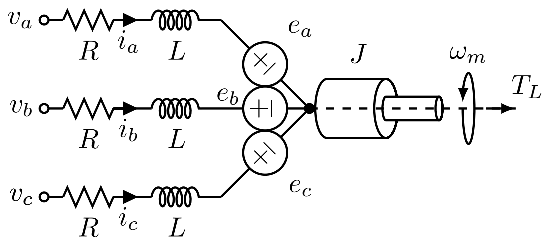
Mechanical torque produced by the rotor can be modeled as:

$$T_e = k_f\omega_m + J\frac{d\omega_m}{dt} + T_L. \tag{5.2}$$

Here, T_e, k_f, J, and T_L respectively denote the electrical torque, friction constant, rotor inertia, and an external load on the rotor. The back-emf and the electrical



(a) DC Motor



(b) 3-phase BLDC Motor

Figure 5.5: Electrical topology of DC motors

torque can be modeled as:

$$e = k_e\omega_m, \quad \text{and} \quad T_e \; = k_t\omega_m. \tag{5.3}$$

Here, k_e and k_t are the back-emf and the torque constant respectively. In the case of a three-phase BLDC motor, three armature coils are used to generate motion. The circuit is presented in Figure 5.5(b). The mechanical model for the BLDC motor remains the same as that of the DC motor; however, the electrical circuit is different in both construction and commutation. The electrical and mechanical model of the BLDC motor can be described as:

$$\begin{aligned} v_{ab} &= R(i_a - i_b) + L\frac{d}{dt}(i_a - i_b) + e_a - e_b, \\ v_{bc} &= R(i_b - i_c) + L\frac{d}{dt}(i_b - i_c) + e_b - e_c, \\ v_{ca} &= R(i_c - i_a) + L\frac{d}{dt}(i_c - i_a) + e_c - e_a, \\ T_e &= k_f\omega_m + J\frac{d\omega_m}{dt} + T_L. \end{aligned} \tag{5.4}$$

Symbols v, i, and e denote the phase-to-phase voltage, phase current and phase back-emfs respectively for the three phases a, b, and c. The resistance R and the inductance L are per phase values and T_e and T_L are the electrical torque and load torque. J is the rotor inertia, k_f is the friction constant and ω_m is the rotor speed. The trapezoidal back-emfs and accordingly the electrical torque can be modeled with:

$$\begin{aligned} e_a &= \frac{k_e}{2}\,\omega_m\,F(\theta_e), \\ e_b &= \frac{k_e}{2}\,\omega_m\,F(\theta_e - \frac{2\pi}{3}), \\ e_c &= \frac{k_e}{2}\,\omega_m\,F(\theta_e - \frac{4\pi}{3}), \\ \frac{k_t}{2} &= F(\theta_e)i_a + F(\theta_e - \frac{2\pi}{3})i_b + F(\theta_e - \frac{4\pi}{3})i_c, \end{aligned} \tag{5.5}$$

where k_e and k_t are the back-emf constant and the torque constant. The electrical angle is equal to the rotor angle times the number of pole pairs. The function $F(\cdot)$ implies the trapezoidal waveform of the back-emf. One period of this function can be written as:

$$F(\theta_e) = \begin{cases} 1 & 0 \le \theta_e < \frac{2\pi}{3} \\ 1 - \frac{6}{\pi(\theta_e - \frac{2\pi}{3})} & \frac{2\pi}{3} \le \theta_e < \pi \\ -1 & \pi \le \theta_e < \frac{5\pi}{3} \\ 1 + \frac{6}{\pi(\theta_e - \frac{5\pi}{3})} & \frac{5\pi}{3} \le \theta_e < 2\pi \end{cases} \tag{5.6}$$

Since each voltage relation in equation (5.4) is a linear combination of the other two voltage equations, only two voltage equations are needed. The two equations can be formulated using Kirchhoff's Current Law (KCL) applied at the common node:

$$i_a + i_b + i_c = 0. \tag{5.7}$$

Accordingly, the voltage equations become:

$$\begin{aligned} v_{ab} - e_{ab} &= R(i_a - i_b) + L\frac{d}{dt}(i_a - i_b), \\ v_{bc} - e_{bc} &= R(i_a - 2i_b) + L\frac{d}{dt}(i_a - 2i_b). \end{aligned} \tag{5.8}$$

To make it convenient for simulation with Matlab/Simulink, the equations above can also be modified to allow a state space representation before modeling.

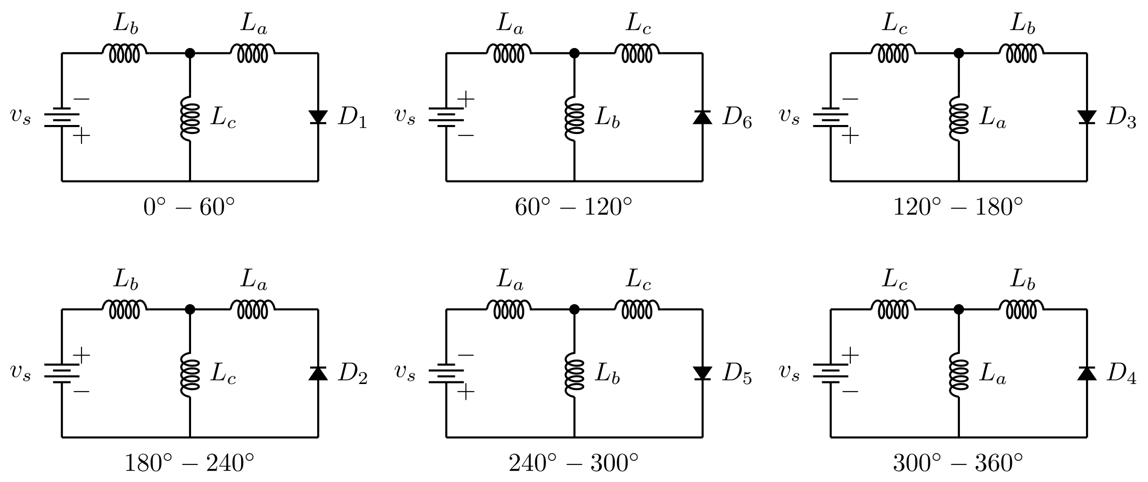


Figure 5.6: Circuit configuration formed by the phases during motor commutation.

5.4.1 Inverter Model

The inverter model of a BLDC motor depends on the circuit configuration formed during each phase of the commutation sequence. The effect on the source voltage is modeled while switching from one state to the next state takes place. For simplicity of the simulation model, hall sensor signals are not generated; instead, zero crossing of current in the motor phases is used to trigger the phase switching during commutation. Figure 5.6 presents the circuits at every 60° of the electrical cycle during the commutation process.

The switches and the diodes are assumed to be ideal devices throughout the operation. The phase current that is being turned off will flow through a freewheeling diode while the current in the phase that is turned on is rising from zero. Looking at the second state ($60° - 120°$ position) in Figure 5.6, when the diode current is nonzero, the phase-to-phase voltages are $v_{ab} = v_s$, $v_{bc} = 0$ and $v_{ca} = -v_s$. When the diode current has reached zero, voltages v_{bc} and v_{ca} will have a different value, which depends on back-emfs. In the event of Matlab simulation only v_{ab} and v_{bc} are sufficient to simulate the dynamics of the motor. Figure 5.7 presents a closer view

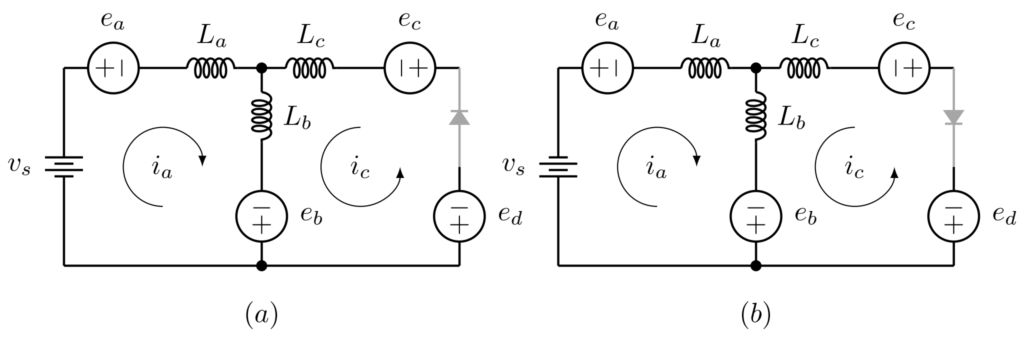


Figure 5.7: Two circuit topologies during $60° - 120°$ electrical cycle.

of the two circuit topologies at $60° - 120°$. The effect of the back-emf and diode as a voltage source is considered wherein the voltage source e_d simulates the diode behaviour [124].

In the case of the non-zero diode current, the value of e_d is zero; however, when the diode current has reached zero, e_d must generate a voltage that forces the current to remain at zero until the next phase starts. This value of e_d is calculated using i_c as a function of voltage sources, setting it equal to zero and solving for e_d. For the circuit in Figure 5.7(a) the phase-to-phase voltage becomes:

$$i_3 \neq 0 \begin{cases} v_{ab} = v_s \\ v_{bc} = 0 \\ v_{ca} = -v_s \end{cases} \tag{5.9}$$

$$i_3 = 0 \begin{cases} v_{ab} = v_s \\ v_{bc} = \frac{1}{2}(-v_s + e_a + e_b - 2e_c) \\ v_{ca} = \frac{1}{2}(-v_s - e_a - e_b + 2e_c), \end{cases} \tag{5.10}$$

Table 5.3: Inputs for the inverter model

Electrical Cycle θ_e	Phase Current	$v_{ab} - e_{ab}$	$v_{bc} - e_{bc}$
$0° - 60°$	$i_a \neq 0$	$v_s - e_a + e_b$	$-v_s - e_b + e_c$
	$i_a = 0$	$\frac{1}{2}(v_s + e_b - e_c)$	$-v_s - e_b + e_c$
$60° - 120°$	$i_c \neq 0$	$v_s - e_a + e_b$	$-e_b + e_c$
	$i_c = 0$	$v_s - e_a + e_b$	$\frac{1}{2}(-v_s + e_a - e_b)$
$120° - 180°$	$i_b \neq 0$	$-e_a + e_b$	$v_s - e_b + e_c$
	$i_b = 0$	$\frac{1}{2}(v_s - e_a + e_c)$	$\frac{1}{2}(v_s - e_a + e_c)$
$180° - 240°$	$i_a \neq 0$	$-v_s - e_a + e_b$	$v_s - e_b + e_c$
	$i_a = 0$	$\frac{1}{2}(-v_s + e_b - e_c)$	$v_s - e_b + e_c$
$240° - 300°$	$i_c \neq 0$	$-v_s - e_a + e_b$	$-e_b + e_c$
	$i_c = 0$	$-v_s - e_a + e_b$	$\frac{1}{2}(v_s + e_a - e_b)$
$300° - 360°$	$i_b \neq 0$	$-e_a + e_b$	$-v_s - e_b + e_c$
	$i_b = 0$	$\frac{1}{2}(-v_s - e_a + e_c)$	$\frac{1}{2}(-v_s - e_a + e_c)$

and from Figure 5.7(b):

$$i_3 \neq 0 \begin{cases} v_{ab} = -v_s \\ v_{bc} = 0 \\ v_{ca} = v_s \end{cases} \tag{5.11}$$

$$i_3 = 0 \begin{cases} v_{ab} = -v_s \\ v_{bc} = \frac{1}{2}(v_s + e_a + e_b - 2e_c) \\ v_{ca} = \frac{1}{2}(v_s - e_a - e_b + 2e_c). \end{cases} \tag{5.12}$$

It can be seen from equations (5.9) to (5.12) that the output voltage of the inverter not only depends on the source voltage but also on the value of back-emfs and whether the phase currents are zero or nonzero. To accommodate the inverter dynamics in the simulation model, the difference between phase-to-phase voltage and their corresponding back-emfs is used. Table 5.3 provides the governing equations for the inverter model used in simulating the behaviour of the BLDC motor.

5.4.2 Closed Loop Speed Control Model

A closed-loop PID control system shown in equation (5.13) is used to control speed of the BLDC motor. In the feedback control system, $e(t)$ corresponds to error between the controlled variable $y(t)$ (i.e., rotational velocity) and the setpoint/desired value $y_d(t)$, which is prescribed at a higher control level. The goal of the feedback controller is to eliminate the error $e(t)$ between $y(t)$ and $y_d(t)$. In the case of motor speed control, the value of $y(t)$ is measured by the speed estimator, which is then compared with $y_d(t)$ to generate the error $e(t)$. The output of the controller, $u(t)$ is a function of $e(t)$. Typically, this is a signal that requires scaling to match the plant (i.e., motor). The PID controller is described with the differential equation as:

$$u(t) = k_p \Big[e(t) + \frac{1}{T_I} \int e(t)dt + T_d \frac{de(t)}{dt} \Big]. \tag{5.13}$$

Here, k_p is the proportional gain, T_I is the integral time constant, and T_d is the derivative time constant.

5.5 Hardware Implementation

The system architecture of the proposed hardware realization process for the BLDC motor driver and its associated speed controller is depicted in Figure 5.8. The driver and controller are achieved in hardware in two main steps: open loop motor driver implementation and closed loop speed controller integration.

First an open loop circuit is derived to operate the motor for a given direction and PWM width. This is supported by a PWM generator circuit, the commutation circuit, and the switching electronics connected to the battery. The PWM generator generates the PWM signal based on the duty ratio. The commutator circuit uses the PWM signal and the hall sensor signal to generate an appropriate drive signal for the electronic switches, which in turn operate the motor. The PWM duty ratio defines

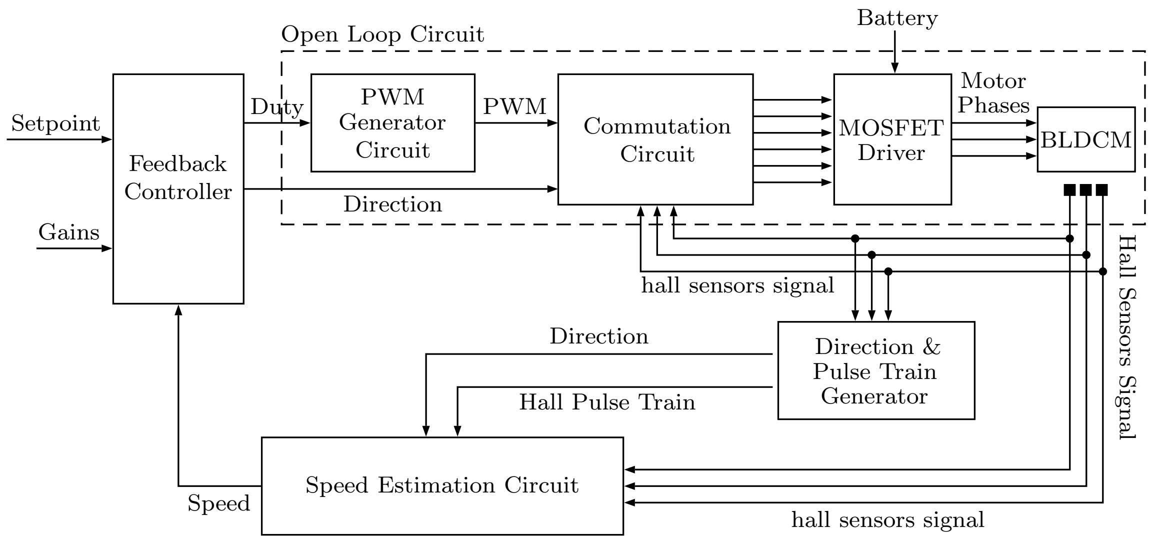


Figure 5.8: Block diagram of the FPGA implementation approach.

the average voltage across the two active phases of the BLDCM. The voltage in turn produces current to generate torque in the motor shaft.

In order to close the loop for a speed controller, a speed estimation circuit is required, which acts as the feedback sensing element for the control law. Therefore, a speed estimation circuit is employed that relies on hall sensor input, direction of motion, and the hall sensor pulse train. Both direction and pulse train are generated using a separate digital circuit, which also depends on the hall sensor signal. The speed estimation signal is then compared with its reference value (set-point) to evaluate feedback error. The feedback error is processed in the feedback controller to provide the controlled PWM duty ratio to minimize the error and maintain the speed of the motor at the desired speed.

5.5.1 Implementation of the Commutation Core

In a digital circuit, an oscillator based clock signal (i.e., master clock) is used to keep all activities and signals in synchronicity. A commutation clock *clk* is required to drive the circuits associated with the commutation cores as presented in the data

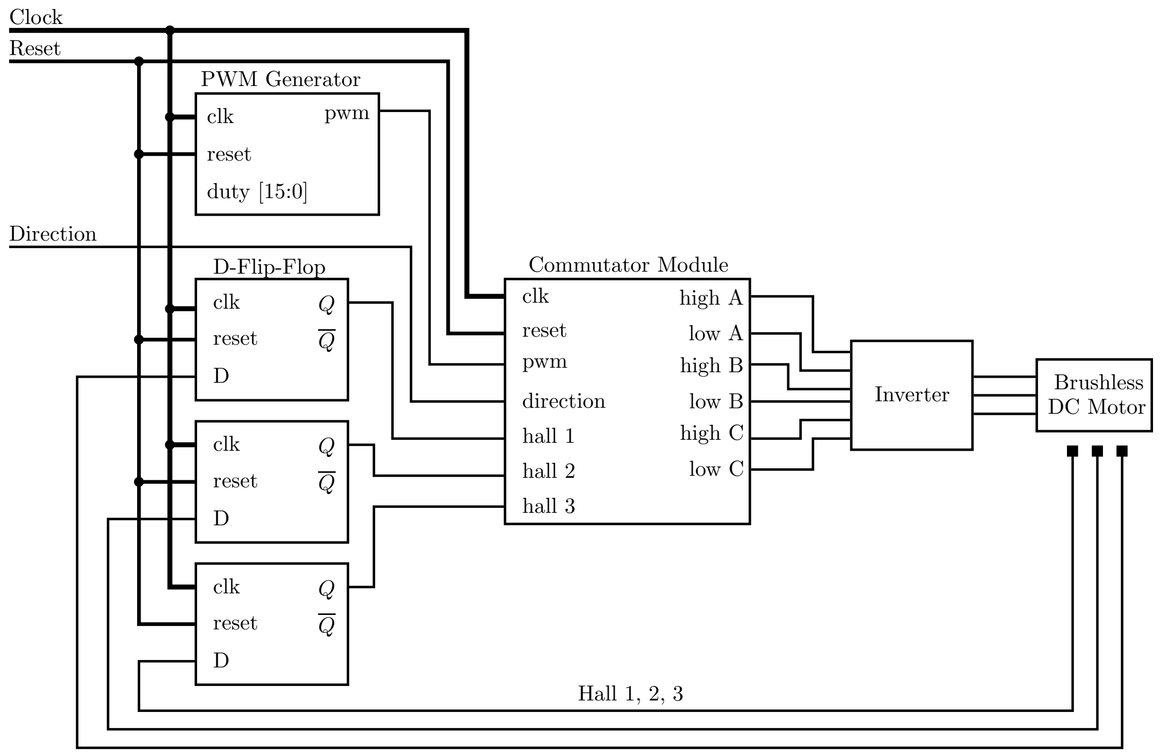
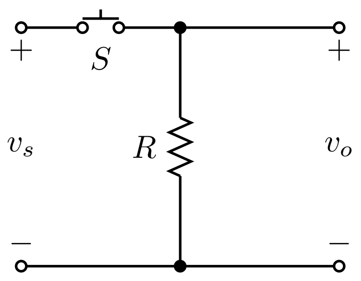
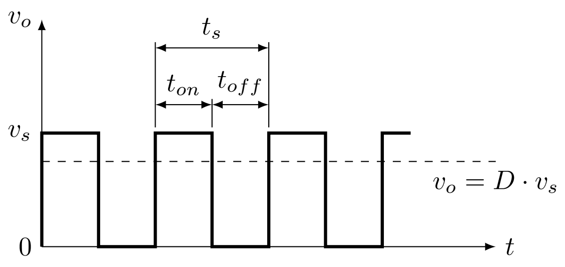


Figure 5.9: Block diagram showing the commutation driver implementation for FPGA Hardware.

flow diagram in Figure 5.9. The clock is essentially a scaled version of the master FPGA clock. The master clock operates at 50MHz: a square wave with 20 ns of period and 50% duty. An asynchronous signal is used to reset the system. The PWM module generates a fixed frequency PWM signal *pwm*, wherein the positive width (i.e., duty ratio) of the signal is used to regulate the conduction time of the high side MOSFET switches of the commutation driver circuit. Essentially, the technique helps to control the average output voltage [46]. Three D-flip-flops are used to synchronize the hall-effect signal with the commutation clock. The sensor signals are then fed to the commutator module along with the PWM signal to define the states of the semiconductor switches at any instant of the hall sensor code (i.e., the combination of hall-sensor signals).



(a) A dc-to-dc converter representing the PWM technique.



(b) A typical PWM signal in action.

Figure 5.10: Pulse Width Modulation Signal.

5.5.2 Implementation of the PWM generator

In order to implement the PWM generator and to understand the concept of the PWM a simple analysis with a switch is in order. The analysis is conducted assuming a resistive load as shown in Figure 5.10(a).

In practice the load is resistive-inductive as is the case with any electric machine. When the switch S is closed (i.e., turned on) the input voltage v_s is applied to the resistive load R. The on-time t_{on} is defined as the amount of time for which the switch is closed resulting in $v_o = v_s$. Off-time t_{off} is when the switch is open leaving no voltage applied to the load ($v_o = 0$). To understand the relationship between the input voltage and the output voltage as a function of on-time and off-time of the switch, two parameters are defined; namely, duty ratio (D) and period of switching (t_s). The switching time period is the sum of the on-time and off-time as shown in Figure 5.10(b). The duty ratio is:

$$D = \frac{t_{on}}{t_s}. \tag{5.14}$$

Taking the integral of the output voltage over one time-period and dividing by the time period yields the input to output relationship.

$$\begin{aligned} v_o &= \frac{1}{t_s}\int v_s(t)dt \\ &= \frac{t_{on}}{t_s}v_s = D \cdot v_s. \end{aligned} \tag{5.15}$$

Duty ratio can be varied either by varying frequency of the PWM signal or by varying positive width of the signal as in this case. The reason for choosing constant frequency PWM control is that the harmonics produced are at the switching frequency and its multiples, makes filtering much easier [125]. In addition, using a fixed frequency PWM, we can filter acoustic noises of the motor provided that the operating frequency is above the audible frequency of the human ear.

To implement the PWM generator in FPGA hardware, a counter is defined that increments on the positive edge of the scaled clock and rolls over when it reaches to its highest value. This is a very common practice in digital hardware coding as it uses bit vectors to define signal (e.g., a 3 bit counter starts from 0 and rolls over after $2^3 - 1 = 7$).

Using the counter, a saw-tooth signal is generated in the FPGA hardware as seen in Figure 5.11. The duty cycle value or the output voltage control signal is compared with the saw-tooth signal to produce the PWM signal. If the saw-tooth signal value is less than the voltage control signal the PWM signal is high (i.e., the switch is turned on) and otherwise it remains low. By varying the length of the counter the frequency of the PWM signal is varied. However, the frequency of the PWM signal is limited by the frequency of the input clock of the FPGA hardware and the switching limit of the power MOSFETs.

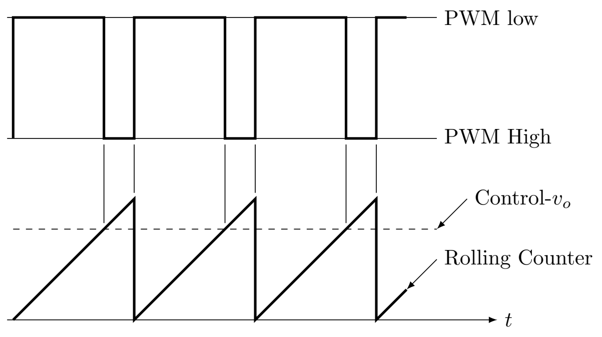


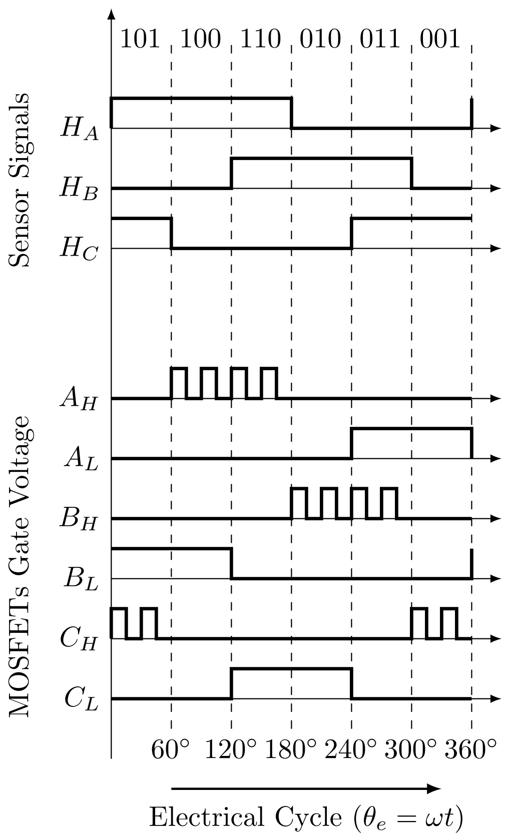
Figure 5.11: Pictorial description of how the PWM signal is generated.

5.5.3 Implementation of the Commutation Protocol

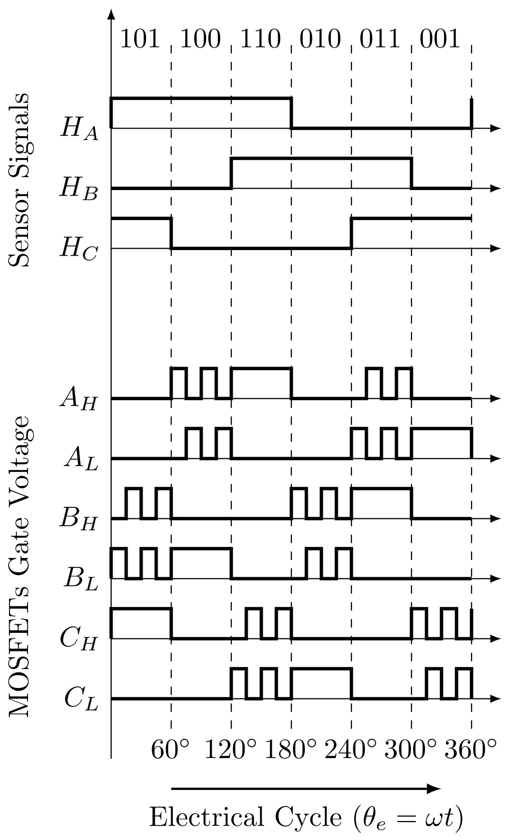
In order to get constant output torque and constant output power, the current is driven through a motor winding during the flat portion of the back-emf waveform as seen in Figure 5.2(b). During that time only two phases are connected in series with the DC power bus while the third phase remains open. The required phase is activated with its respective inverter switches. Two lookup tables are implemented in the FPGA hardware to define the state of the switches according to Tables 5.1 and 5.2 for clockwise rotation and counter clockwise rotation respectively. Figure 5.12(a) presents the commutation waveform during a clockwise rotation. Note that the high side activation signal in the tables is replaced with a PWM signal during implementation of the lookup table. This provides the controllability of motor speed based on the duty ratio of the PWM signal in real-time.

The commutation process naturally involves conduction losses and switching losses [126] due to the use of a power device in the inverter. While off the shelf motor controllers do not allow access to the commutation and cannot guarantee efficient commutation, this research work identifies the potential of custom commutation to reduce these losses with only minimal effort (i.e., changing the commutation sequence).

In the case of PWM waveforms for the conventional approach shown in Figure



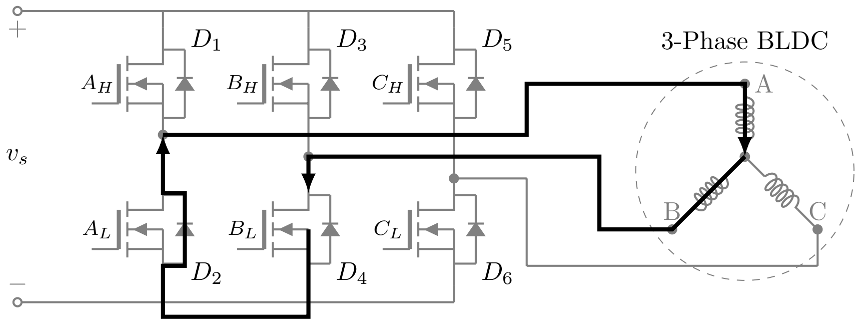
(a) Conventional PWM waveform



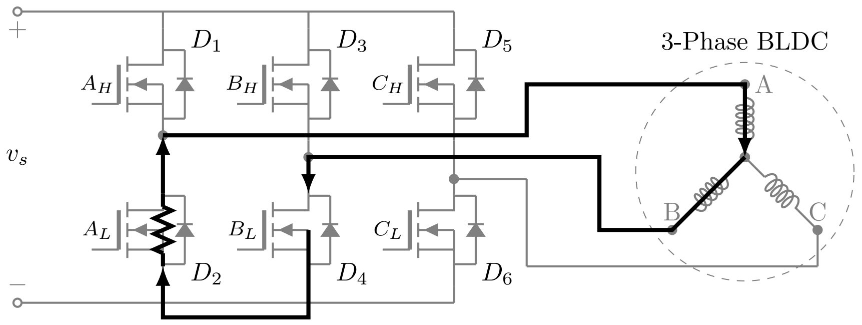
(b) Low-loss PWM waveform

Figure 5.12: Power efficient commutation waveforms

5.12(a), the high power size is chopped in the 2/6 fundamental period and the duty ratio of the PWM is defined from the speed reference. When the high side power switch is chop controlled, the associated low side switch is not triggered and remains "off". A similar control signal is applied to the low-side power switch at a 180° shift, except in this case the low side is clamped to a negative dc link. These control signals are applied to the other two phases with a 120° shift as shown in the same figure. Figure 5.13(a) illustrates the current path during PWM commutation using $\omega t \in [60° - 120°]$ and $A_H = off$, as an example. Here power switches $A_L = off$ and $B_L = on$ respectively. Under this condition the current flows through the anti-parallel diode of the power switch A_L, and thereby results in significant conduction losses,



(a) Current path during classical commutation technique



(b) Current path during low-loss commutation technique.

Figure 5.13: Illustration of current path in conventional and low-loss commutation technique.

which is equal to the product of forward drop voltage of diode and load current [47]. In contrast, when the PWM waveform proposed in [47] presented in Figure 5.12(b) is implemented, the conduction loss can be improved. In this technique, the high side power switch is chopped in the 1/6 fundamental period and clamped to the positive dc link in the consecutive 1/6 fundamental period. As the high side device has chop control, the associated low side power device is triggered by the inverse signal of chop control. The "on-state" for the low side power device indicates that the output terminal is connected to the negative dc link. Similarly, when the low-side power device has chop control, which has a half-fundamental delay as compared to that for high-side control, the associated high side power switch is triggered by the inverse of the chop control signal. These control signals are applied to the other two phases with 120° shift.

The current path for this technique is presented in Figure 5.13(b) using $\omega t \in [60° - 120°]$ and $A_H = off$ as an example. Under the conditions, when $A_H = off, A_L = on$, and $B_L = on$, the loss for the lower side, consisting of the power switch and antiparallel diode of phase A, equals the product of turn on resistance of MOSFET ($R_{DS_{on}}$), and the square of the load current rather than the product of the forward voltage drop of diode and load current, as in conventional technique.

5.5.4 Hall Sensor Based Direction of Motion Estimation

In the case of closed loop speed control, the ability to change the speed of the motor in both directions can improve the dynamics of the speed control system [44]; hence, it is necessary to identify the direction of rotation. As mentioned earlier, the hall effect sensor signal is utilized in this work to estimate both motor speed and direction of rotation. In the case of three hall sensors, the signals from the sensors are shifted with respect to each other by 2/3 of the period and their sequence is a function of the direction of rotation. Thus, the sequence is captured with D-type flip-flops to estimate the direction as developed in [51]. Figure 5.14 presents the direction estimation circuit using three hall effect sensor signals and three D-type-flip-flops. A signal from an arbitrarily chosen sensor is connected to the clock input of the flip-flop that triggers the flip-flop, whereas the signal from the subsequent sensor is connected to the flip-flop input. In this solution the motor reversal will change the output state of one of the flip-flops after 1/3 of the rotation for a motor with two poles. For a larger number of poles the detection time improves. Flip-flop outputs are connected to the three-input logical operator AND. The AND gate responds to the opposite change of state. Upon the change of the direction of rotation, the outputs of the AND gates assume opposite states (during 2/3 of the rotation).

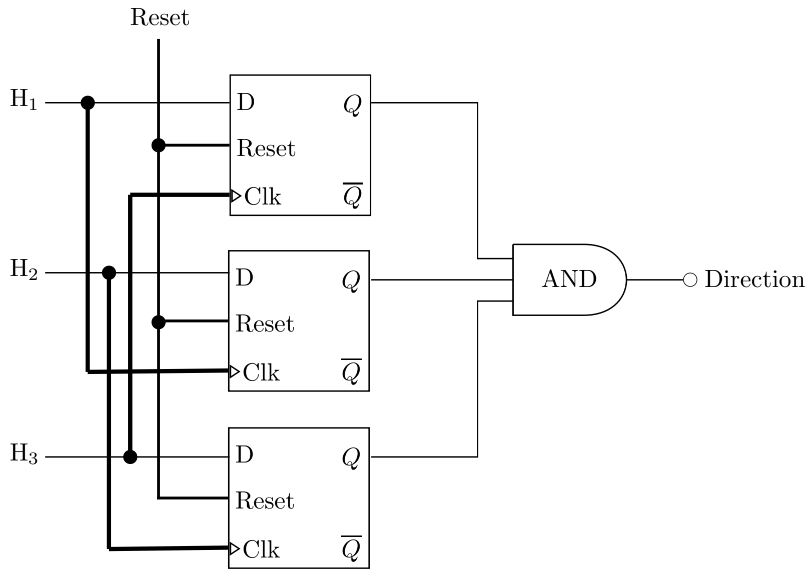


Figure 5.14: Direction estimation circuit using D-Flip-Flop registers.

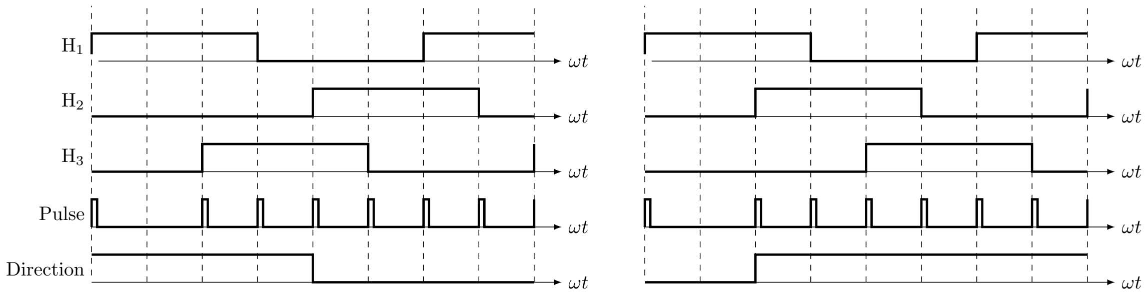
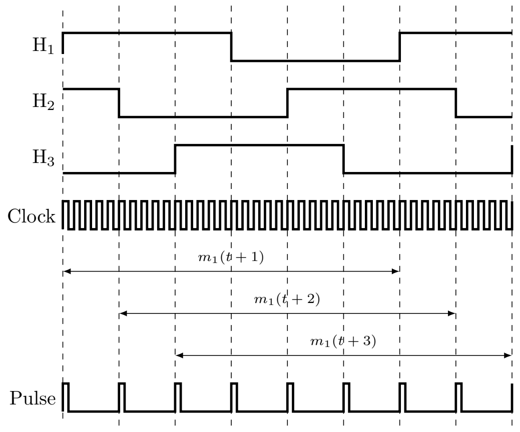
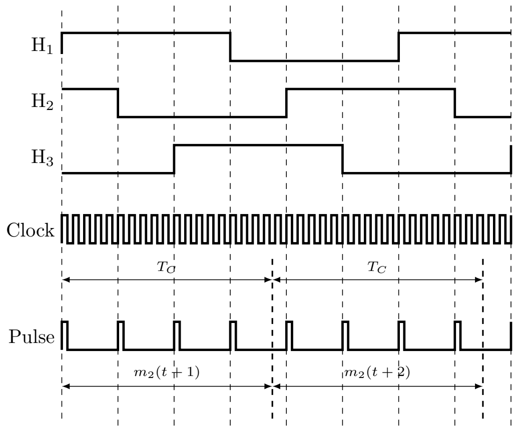


Figure 5.15: Hall sensor pattern indicating the direction of rotation. The left hand waveform is for change in direction from clockwise to counterclockwise motion whereas the right hand waveform is for counterclockwise to clockwise motion.

Figure 5.15 presents representative waveforms of the hall sensor signals during change in the direction of rotation. The output state of the flip-flop represents the direction of the rotation of the rotor. If, for one direction the output is high, then the low state is for the opposite direction. In most drive systems, it is important to be able to detect a change in the direction of rotation as fast as possible (e.g., to implement speed controller, detect the speed zero crossing or stop the drive). In such cases, determining the direction of rotation from just one phase signal is insufficient. Hence, the direction detection algorithm uses all three hall sensor signals.



(a) Waveform showing fundamentals of period M method during speed measurement.



(b) Waveform showing fundamentals of period T method during speed measurement.

Figure 5.16: Timing diagram of the classical time based speed estimation methods: M-method and T-Method.

5.5.5 Hall Sensor based Speed Estimation

The speed estimation in this work is performed using the pulses generated by the same hall sensor signals that measure the direction of rotation. Since the sensor identifies the poles of the permanent magnetics of the spinning rotor and produces logic high signal for the north pole and low signal for the south pole of the magnets, it is possible to measure speed of the motor provided that the number of motor poles are known [127].

5.5.5.1 Time Based Speed Estimation

The classical and probably the simplest method to measure rotor speed is the direct measure of the period of the signals generated by the hall sensors as developed in [50]. The method is also knownn as the Period Method or M-Method. A pictorial representation of the technique is presented with a timing diagram in Figure 5.16(a).

The discrete algorithm for the Period Method can be implemented in terms of a timer that generates clock ticks (m_1) between two successive rising edges of a hall sensor pulse. Given the frequency of the clock pulse (f_c), the speed estimation in revolutions per minute (rpm) can be expressed by the formula:

$$N_f = 60\frac{f_c}{m_1}. \tag{5.16}$$

Since all three sensor signals are equivalent, the measurement update rate can be increased by measuring the period of all three hall sensors as seen in 5.16(a). However, there is a condition when the resolution and the accuracy of this estimation method degrade. At high speed the hall pulses can become close to the resolution of the driving clock pulse introducing inaccuracy to the result. Since the resolution and quantization error for this estimator is variable and the variation is a function of the speed of operation it is less preferable for the use with a feedback control system [128], where the sampling speed is expected to be a constant for effective performance of the controller.

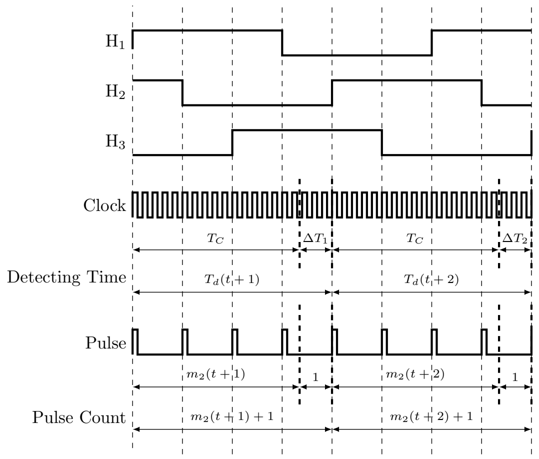
To address the weakness of the M-Method, the number of hall pulses is counted in a fixed time frame. This method is called the frequency method or T-Method [50]. The number of hall pulses m_2 is counted in a fixed time frame T_C, as seen in Figure 5.16(b). The speed in rpm can be calculated using:

$$N_f = 60\frac{m_2}{P \cdot T_C}. \tag{5.17}$$

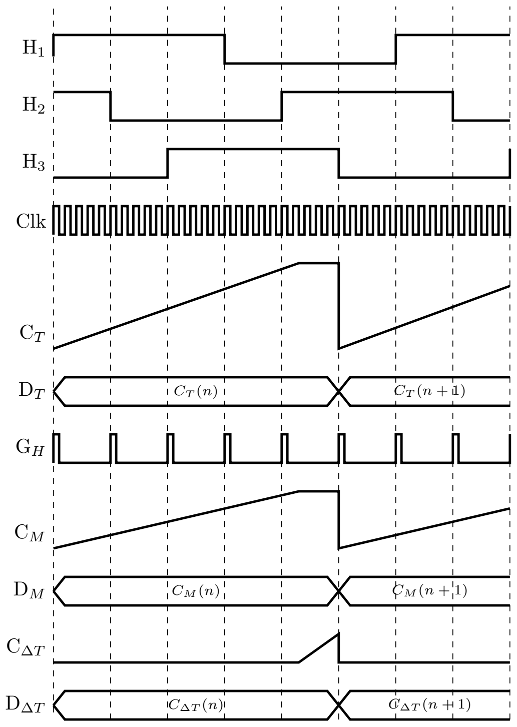
Here, P is the number of hall sensor pulses per rotation. If the motor has p number of pole pairs then the number of pulses can be calculated as:

$$P = 6 \cdot \frac{p}{2}. \tag{5.18}$$

Though the frequency of data output is fixed in the T-Method, it fails to measure speed accurately in the low speed region, where the gap between two successive hall pulses



(a) Waveform showing fundamentals of M/T method of speed measurement.



(b) Timing diagram of M/T method showing the digital signals during implementation.

Figure 5.17: Timing diagram of the mixed mode M/T-method.

becomes longer then T_C; consequently, measurement data produces an inaccurate value or in an extreme case, zero. Therefore, a combination of both the M-method and T-method is chosen in this work, which is called the M/T method in [128]. The M/T method borrows the fundamental concept of previous methods and implements them in a way that overcomes the issues of both the low speed and high speed regime.

The timing diagram of the mixed mode M/T method is shown in Figure 5.17(a). First the number of pulses generated from hall sensors m_2, is computed within a prescribed time period T_C. As soon as the counting is over, a time counter records the time ΔT until the next pulse is observed. Then the total detecting time T_d is determined.

If the detecting time $T_d = T_c + \Delta T$ is digitized with the clock pulse of the frequency (f_c) then the speed N_f(rpm) is obtained by:

$$N_f = 60 \cdot \frac{(m_2 + 1)}{P \cdot T_d}. \tag{5.19}$$

Here, P is the number of hall sensor pulses per rotation. The mixed M/T method improves the quantization and accuracy problem at both extremes of the operating range of motor.

In order to implement the mixed mode M/T method in hardware, it is necessary to determine the frequency of the hall sensor pulse signals for a given time frame. The resolution of the measurement is also increased using the phase locked characteristics among the three hall sensor signals (six transitions that occur in one revolution for one pole pair). Figure 5.17(b) presents the waveforms associated with the discrete implementation of this technique.

A short pulse (G_h) is generated at each change in the signals from the hall sensors (H_1, H_2, H_3). The pulses are detected using an independent process that employs a three-bit D type Flip-Flop register and a high-speed clock with known frequency (f_{clk}). The register captures the combination of the three hall sensor signals at every instant of the rising edge of the fast clock and compares it with the previous instant to find a mismatch. The mismatch between two consecutive signals signifies that there is a change in the hall sensors' state and hence a pulse (G_h) is generated. A counter (C_m) is employed to capture the number of pulses within a given time period (T_c) calculated in terms of the number of clock ticks elapsed in the high-speed clock. The amount of clock ticks is tracked using a counter (C_T). As soon as fixed period based counting hits the predefined maximum, a second counter $(C_{\Delta T})$ starts counting the clock ticks until the next pulse in the G_h signal is encountered. The counters are reset at this moment and the counter values are copied into data registers D_M and D_T.

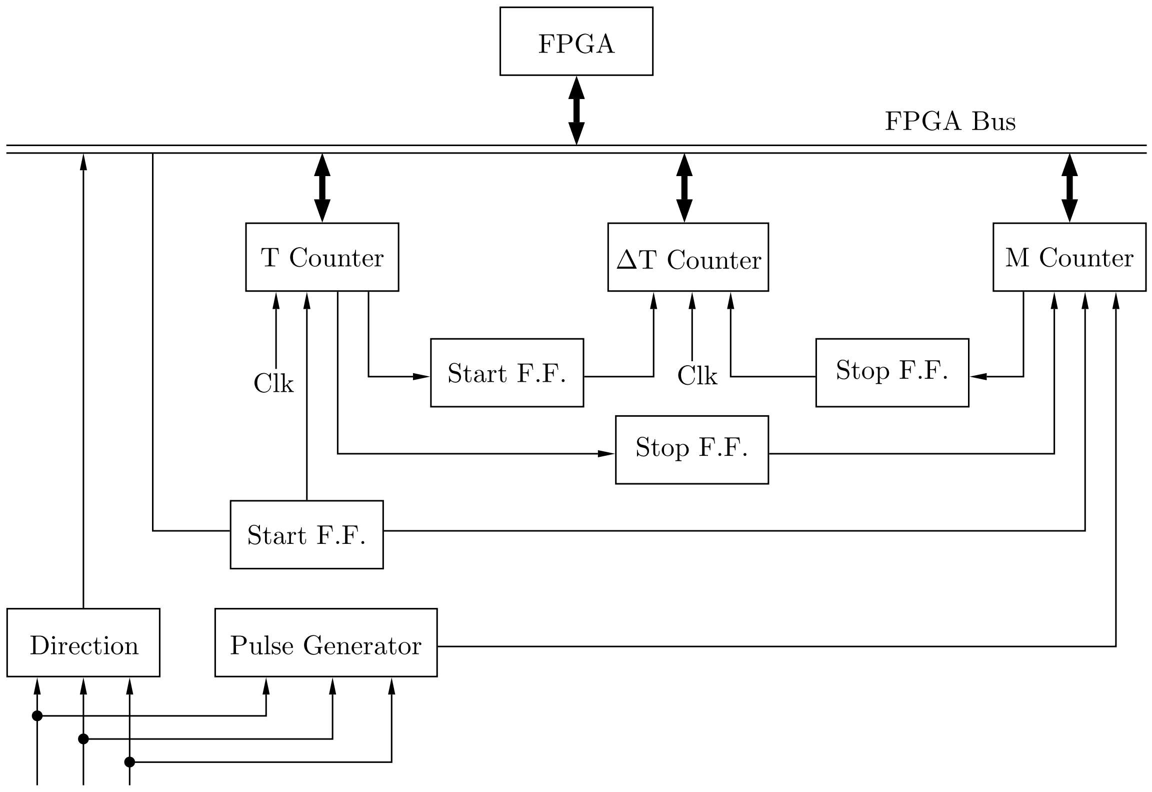


Figure 5.18: Hardware architecture of M/T speed estimation core.

To integrate the algorithm into FPGA hardware, a master controller is defined to control the activity of three hardware modules; namely, the T counter, ΔT counter, and M counter, which are based on the driving clock. The modules are connected to the FPGA bus that shares the driving clock, registers and triggers signals from the master controller. The direction module estimates the direction of motion using the hall sensor signals, while the pulse generator generates pulses for any change in either of the hall sensor signals. The direction information is fed directly to the FPGA. The T counter and the ΔT counter are fed with clock ticks generated by the driving clock while the M counter counts the pulses elapsed for a given period of time (T_C), which depends on the pulse signal fed from the pulse generator.

Though the M/T-method for speed measurement overcomes the inconsistency of the dynamic range of the motor speed, it still suffers from a quantization error as

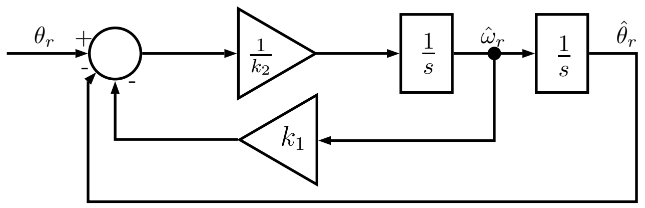


Figure 5.19: Block diagram of the speed observer.

the algorithm is based on a fixed period update rate. In the case of discrete control systems where computations take place at a finite sampling interval, the quantization error occurs at low frequency; it results in the fluctuation of the motor current and, consequently, in a fluctuation in its driving torque [51]. In order to address this phenomenon, a state observer based speed measurement algorithm is developed, which can numerically estimate the angular velocity for a given angular position of the rotor at two successive sample times.

5.5.5.2 State Observer Based Speed Estimation

The deterministic state observer developed in [129] is adopted for this application and the discrete formulation of the observer is implemented in FPGA hardware. A block diagram of the state observer is presented in Figure 5.19 and the associated governing equation can be written as:

$$[(\theta_r - \hat{\theta}_r) - k_1\hat{\omega}_r]\frac{1}{k_2 s}\frac{1}{s} = \hat{\theta}_r, \tag{5.20}$$

where θ_r and $\hat{\theta}_r = \frac{\hat{\omega}_r}{s}$ are the measured (from the hall sensor pulse train) and estimated (from the observer) rotor positions (units in radian) respectively. $\hat{\omega}_r$ is the estimated rotor speed in rad/s, and $1/s$ is the operator for integration.

The speed observer is a continuous model of a speed estimation algorithm that is subject to a set of tuning parameters k_1 and k_2. Since this is an estimation model, a slight delay in the estimation signal is expected. The effect of delay can be made insignificant through tuning of the parameters.

If the discrete implementation of the observer is operated at a high sampling rate, the algorithm becomes insensitive to the quantization error. Besides, the observer method enables the estimation of position, overcoming the problems related to the adoption of the low-resolution hall sensor signal when the position control is required. Hence, the observer method is preferred in this research work. However, the hardware implementation of the observer requires discretization of the continuous time domain algorithm into a discrete time algebraic equation.

Approximating the integration routine in the algorithm by a discrete summation and evaluating the algorithm using the zero order hold (ZOH) method with sampling time T, the discrete approximation is obtained as:

$$\begin{aligned} \hat{\omega}(t+2) &= K_1 \cdot [\theta(t+1) - \theta(t)] - K_2 \cdot \hat{\omega}(t+1) - K_3 \cdot \hat{\omega}(t) \\ \hat{\omega}(t) &= K_1 \cdot [\theta(t-1) - \theta(t-2)] - K_2 \cdot \hat{\omega}(t-1) - K_3 \cdot \hat{\omega}(t-2) \end{aligned} \tag{5.21}$$

Here, $K_1 = \frac{T}{k_2}, K_2 = \frac{k_1}{k_2}T - 2, K_3 = 1 - \frac{k_1}{k_2}T - \frac{T^2}{k_2}$, and T is the sampling time period. To optimize the consumption of FPGA hardware logic resources, the constants and their derivatives are calculated before the synthesis of the algorithm in the hardware.

To estimate the speed of the motor ($\hat{\omega}_r$), the rotor position (θ_r) is required for this observer. In this implementation, the position value is obtained in the form of a pulse count ($\psi(t)$) generated at the pulse generator output. Therefore, the relation between the counter value and θ_r at any instant can be expressed as:

$$\theta_r(t) = \frac{2\pi}{6 \cdot P} \cdot \psi(t), \tag{5.22}$$

where P is the number of pole pairs. The counter can be a very large number if the motor operates at high rpm. Implementation of this large number (unknown maximum value) is impractical in FPGA hardware. Therefore, the counter is used

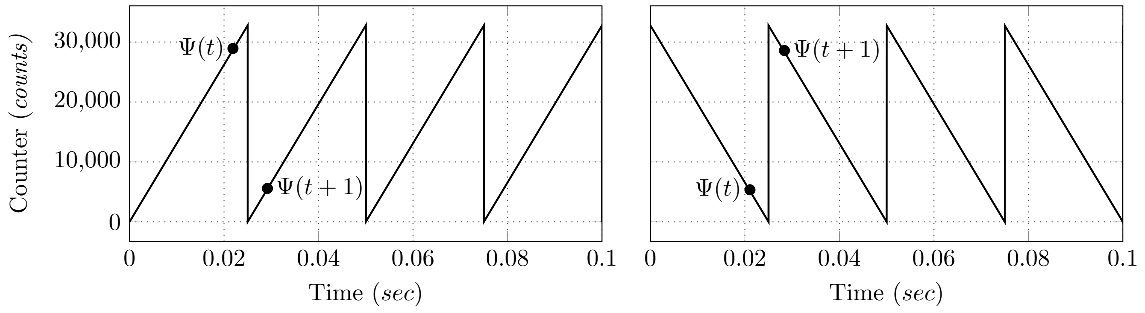


Figure 5.20: Pictorial representation of counter overflow and underflow.

as a standard logic vector with finite bit length (e.g., 16 bit counter) instead of the absolute measure of the position in terms of pulse count. However, the counter signal can overflow or underflow (depending on the direction of rotation) as it reaches its maximum or minimum value respectively. But the discrete algorithm of the observer only requires the difference between the previous position and the present position as seen in equation (5.21); hence, an algorithmic approach can be adopted to estimate the difference in position by using the rollover counters.

For instance, assume the motor is rotating in the positive direction at a constant speed and a 16 bit counter is incrementing. When the counter has reached the maximum ($2^{16}-1$) value, an overflow occurs and the counter value resets to zero and then increments again. On the other hand, for rotation in negative direction the counter is decrementing an underflow occurs as soon as the counter value reaches zero. In this case, the counter will reset to its maximum value and start decrementing again. The phenomenon is illustrated in Figure 5.20.

To address the counter overflow or underflow, consider equation (5.21). It can be seen that, only the difference between the previous counter value and the present counter value, i.e. $\theta(t+1)-\theta(t)$ is sufficient to calculate the speed of the motor. So, in a normal situation when the counter is incrementing and without overflow, then the value of $\psi(t+1)$ should be larger than $\psi(t)$. On the other hand, when the counter is decrementing and without underflow, then the value of $\psi(t+1)$ should be less than that of $\psi(t)$.

However, during counter overflow or underflow, these conditions do not hold anymore. Accordingly, the correct value of $\psi(t+1) - \psi(t)$ is obtained by adjusting the value of $\psi(t)$ as soon as the overflow or underflow is detected. In the case of overflow, instead of using $\psi(t)$, the value of $[\psi(t) - 2^{15}]$ is used, so that the calculated value of $\psi(t+1) - \psi(t)$ is correct. In the case of counter underflow, $\psi(t)$ is replaced by $[\psi(t) + 2^{15}]$.

The discrete formulation of the observer is implemented in FPGA hardware using three multipliers and three adders as seen in equation (5.21). It needs to be noted that the observer based speed estimation method consumes more hardware resources compared to the time based speed estimation method due to algebraic computations involved in the implementation of the observer. However, the improvement in quantization error may offset the odds of the observer based speed estimation algorithm.

5.5.6 Hardware Implementation of the PID Controller

In order to implement the PID controller in FPGA hardware it is necessary to reformulate the PID algorithm in Equation (5.13) so that it can accommodate the discrete nature of the hardware. Reconstructing the PID control law as a difference equation [130], wherein the derivative term is defined with a first-order difference expression and the integral by a summation, the control law for a small sample interval T can be formulated as:

$$u(t) = k_p\Big[e(t) + \frac{T}{T_I}\sum_{i=0}^{t} e(i)dt + \frac{T_d}{T}\{e(t) - e(t-1)\}\Big]. \tag{5.23}$$

Since T_I and T_d are constant, we can rewrite equation (5.23) as:

$$u(t) = k_p e(t) + k_i \sum_{i=0}^{t} e(i) + k_d\{e(t) - e(t-1)\}, \tag{5.24}$$

where $k_I = k_p \frac{T}{T_I}$ is the integral coefficient, $k_d = k_p \frac{T_d}{T}$ and is the derivative coefficient. However, to compute the sum, all past errors, $e(0)...e(t)$, have to be stored in the FPGA hardware, which is impractical. As an alternative, a counter could be used that resets when an overflow or underflow occurs. But both these cases can have an impact on the performance of the controller.

As a solution to this problem, a recursive algorithm [131] is characterized by the calculation of the control output, $u(t)$ based on $u(t-1)$ and the correction term $\Delta u(t)$. To derive the recursive algorithm, first calculate $u(t-1)$ based on equation (5.24),

$$u(t-1) = k_p e(t-1) + k_I \sum_{i=0}^{t-1} e(i) + k_d\{e(-1) - e(t-2)\}, \tag{5.25}$$

then calculate the correction term as:

$$\begin{aligned}\Delta u(t) &= u(t) - u(t-1) \\ &= K_0 e(t-1) + K_1 e(t-1) + K_2 e(t-2).\end{aligned} \tag{5.26}$$

Here, $K_0 = K_p + K_I + K_d$, $K_1 = -K_p - 2K_d$, and $K_2 = K_d$. Equation (5.26) is also called the *incremental algorithm.* The control output is calculated as:

$$\begin{aligned}u(t) &= u(t-1) + \Delta t \\ &= u(t-1) + K_0 e(t-1) + K_1 e(t-1) + K_2 e(t-2).\end{aligned} \tag{5.27}$$

Since the incremental algorithm only involves algebraic addition and multiplication, it is well suited for digital implementation. Therefore, a combinational parallel implementation architecture is chosen to implement the controller in FPGA hardware, which has its own arithmetic unit, either an adder or a multiplier. In contrast a serial design could be composed of sequential logic to allow all operations to share only one

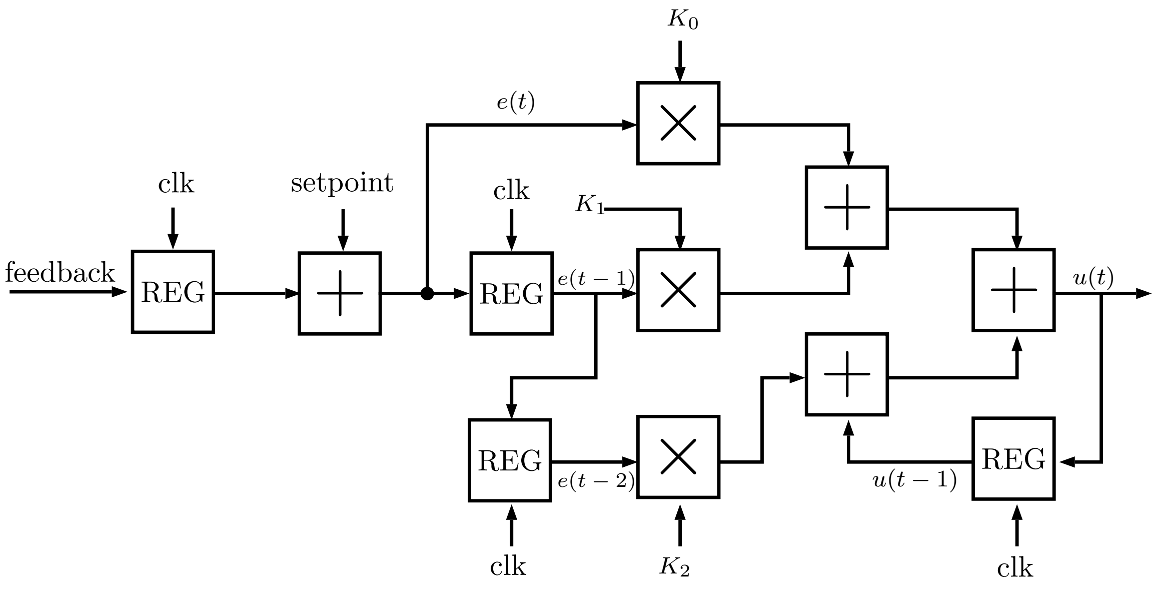


Figure 5.21: Register Transfer Level (RTL) architecture of the incremental PID controller.

adder and one multiplier, which is resource optimized but complex in implementation.

Figure 5.21 presents the RTL flow diagram of the PID incremental algorithm that is implemented in the FPGA hardware. Four adders and three multipliers are required as defined by equation (5.26).

All registers are updated with a clock signal *clk*, which is operated at control loop frequency. Since this is a motor velocity controller, the $feedback$ signal represent the current angular speed $y(t)$ of the motor. The signal is negated before adding to the *setpoint* signal. At the rising edge of clock signal, the error signal $e(t)$ from previous iteration is latched at its corresponding register to evaluate $e(t-1)$ of the current cycle. Similarly, signals $e(t-2)$ and $u(t-1)$ are recorded with the help of two other registers by latching $e(t-1)$ and $u(t)$ respectively. As defined by asynchronous reset, at any given time, all registers can be reset to zero by asserting high to the $reset$ signal. To saturate the output value of control signal $u(t)$ a bounder logic is defined, which limits the output data within the user-defined range prescribed to the controller.

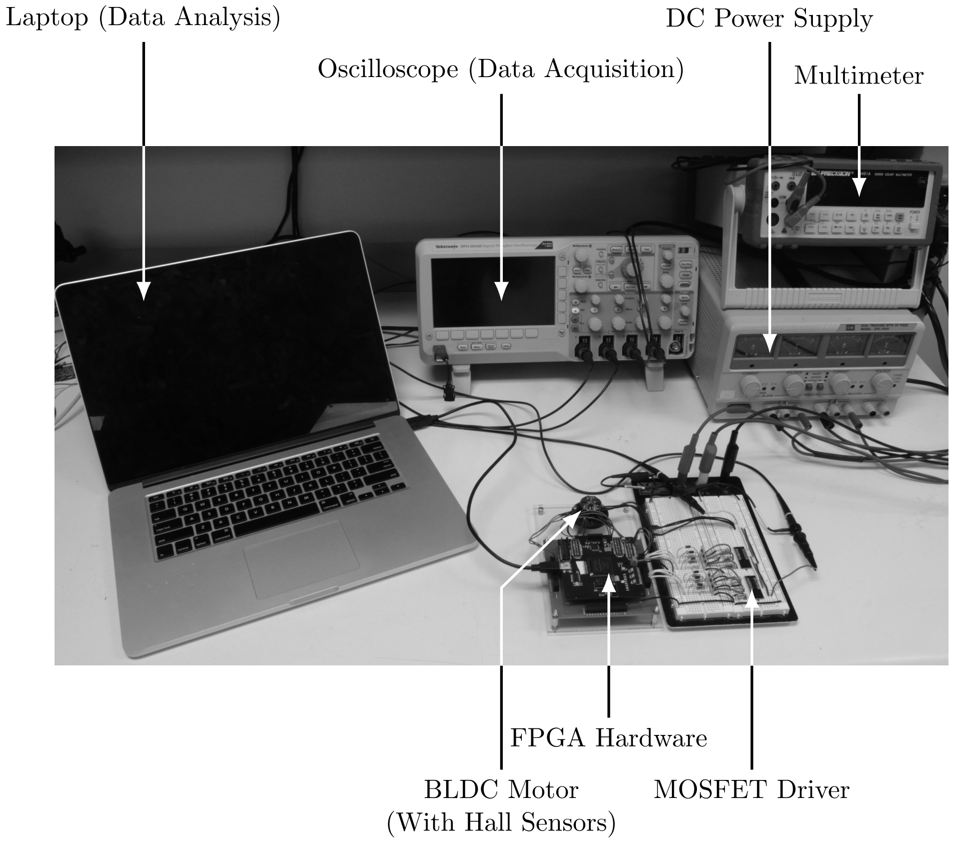


Figure 5.22: Pictorial representation of the experimental setup.

5.6 Experimental Results

An experimental setup was designed and constructed to implement the FPGA hardware cores and to evaluate the performance and functionality of the commutation technique and the control law. The setup is shown in Figure 5.22.

A custom FPGA board was designed to obtain the hardware circuitries. It is the same FPGA board that was designed to control the quadrotor as discussed in Chapter 3. The board is based on an Altera Cyclone III EP3C40F484C6 FPGA chip with 39,600 logic elements. A serial communication protocol (i.e., RS-232) was developed in FPGA to establish communication between the FPGA chip and a personal computer, to send command signals (i.e., duty ratio, desired speed etc.) and capture motor

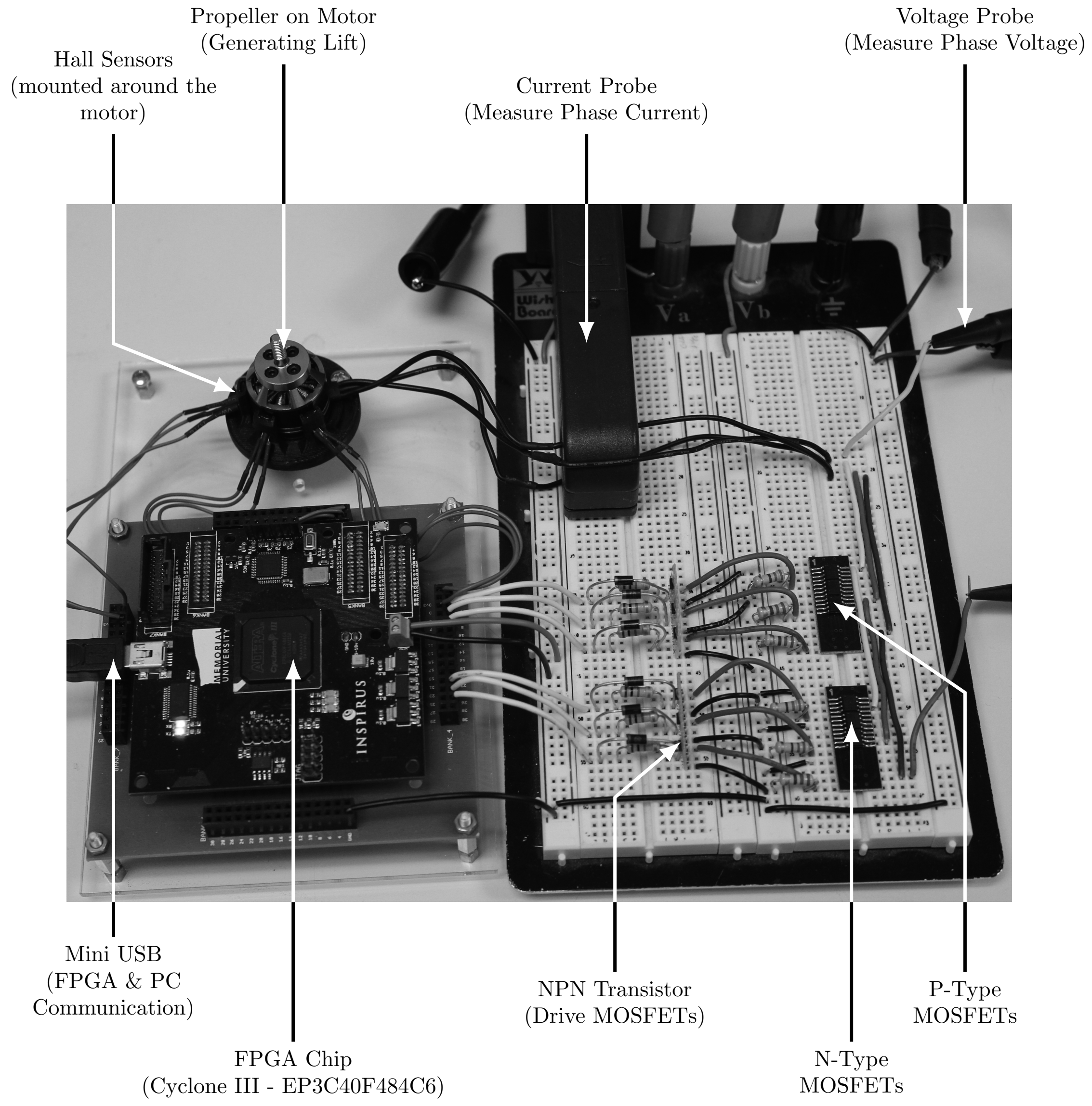

Propeller on Motor (Generating Lift)
Hall Sensors (mounted around the motor)
Current Probe (Measure Phase Current)
Voltage Probe (Measure Phase Voltage)
Mini USB (FPGA & PC Communication)
FPGA Chip (Cyclone III - EP3C40F484C6)
NPN Transistor (Drive MOSFETs)
N-Type MOSFETs
P-Type MOSFETs

Figure 5.23: Close up view of the inverter components in the experimental setup.

responses. An oscilloscope capable of measuring current and storing data was used to capture real-time voltage and current responses from the motor phases. An off-the-shelf hobby motor (A2202L) was chosen for this study. In order to construct the driver circuitry, three P-type MOSFETs were used as the high side switches of the inverter and three N-type MOSFETs were used as the low side switches. Six NPN-transistors were used to drive the MOSFET switches that are controlled by digital

Table 5.4: Motor parameters used in simulation.

Parameter	Value	Units
Number of Poles	14	Poles
Nominal Voltage	7.4	Volts
Terminal Resistance	0.35	Ohm
Terminal Inductance	50E-6	Henry
Torque Constant	9.8E-3	Nm/A
Rotor Inertia	23.5E-6	kg·m^2
Friction Constant	1.1E-5	N·m·s

signals sent from the FPGA board via I/O peripherals. A close up view image of the drive electronics is presented in Figure 5.23. It needs to be noted that, this circuit is used in developement of motor drivers of the quadrotor presented in Chapter 3. Accordignly, the quadrotor is flown and tested by employing this driver to commutate and control all four motors of the vehicle.

To obtain the hardware of the digital circuitries, VHDL was used as the preferred hardware coding language. For defining logic signals and logic vectors, fixed-point implementation was adopted instead of floating point implementation. A widely used fixed point package [132] was used to preserve the accuracy and resolution of the data signals.

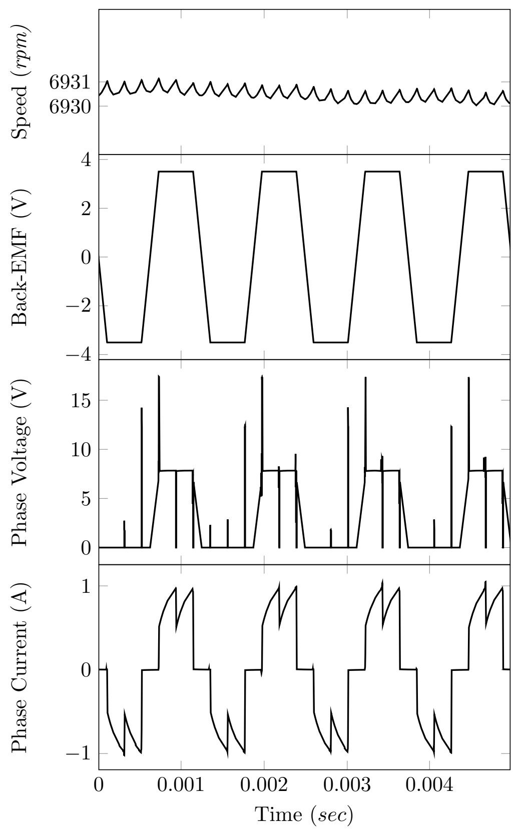
In order to validate the numerical model with experimental results, a simulation study was conducted using the motor parameters. Since the motor used in this study was an off-the-shelf hobby motor and there were no parameter specifications reported in the literature, an identification approach was adopted based on [133]. However, bearing friction was not characterized through this method. Hence, to complete the identification process and obtain an estimated bearing friction, the Matlab/Simulink Parameter Identification Toolbox was used. The identified parameters are presented in Table 5.4.

5.6.1 Experimental Observations from the Commutation Core

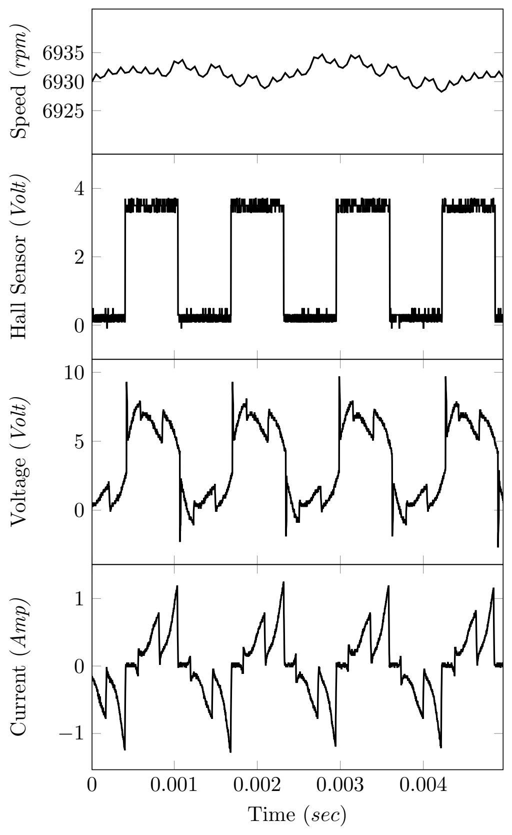
The experimental evaluation is conducted in the same chronological order as the implementation that had taken place. First the commutation circuit was tested in an open loop condition. The current and voltage responses were verified by comparing the results with simulated data. In the next step the speed control algorithms were evaluated and finally the closed loop controlled responses were collected and verified.

Figure 5.24(b) presents the steady state response from one of the motor phases at 100% duty ratio. It can be seen that the back-emf profile generated by the motor model is trapezoidal and the hall sensor signal in 5.24(a) coincides with the zero crossing of the back-EMF. It needs to be noted that the simulation results are not synchronous with the experimental response. However, for the commutation action within a fixed time frame, is sufficient for comparison of the results. Both the phase voltage and current readings confirm the successful implementation of the commutation circuit. The responses of phase voltage and phase current presented in the oscillograms are obtained with respect to the ground of the circuit.

In order to further evaluate the commutation circuit at a different duty ratio, the motor was excited with a lower end duty ratio of 20%. The responses are reported in Figure 5.25. It can be seen that the phase voltage follows the input PWM signal with 20% duty ratio as expected for a PWM based commutation principle. However, the simulation only provides the average phase voltage as the numerical model was simulated for 20% of the rated voltage (i.e., 1.48V). Since the commutation technique is implemented using a fixed frequency PWM signal, the frequency was set to 16kHz where the high frequency acoustic noise was found to be inaudible.



(a) Simulation Results.

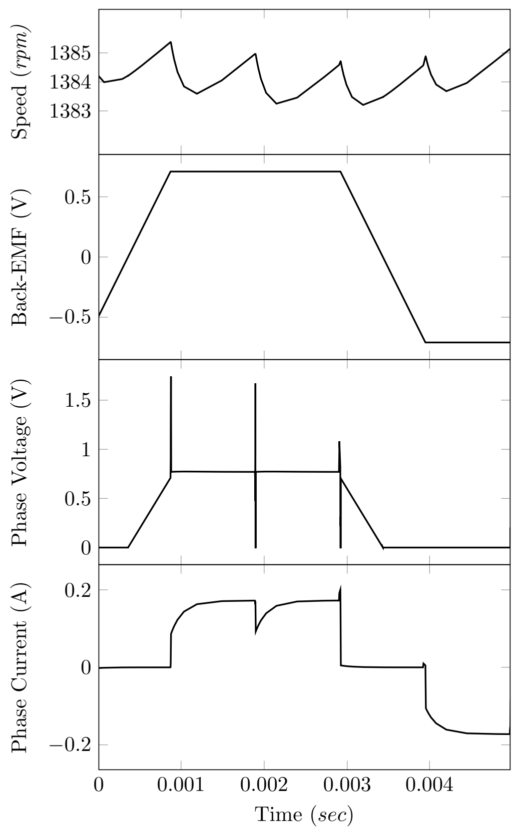


(b) Experimental Results.

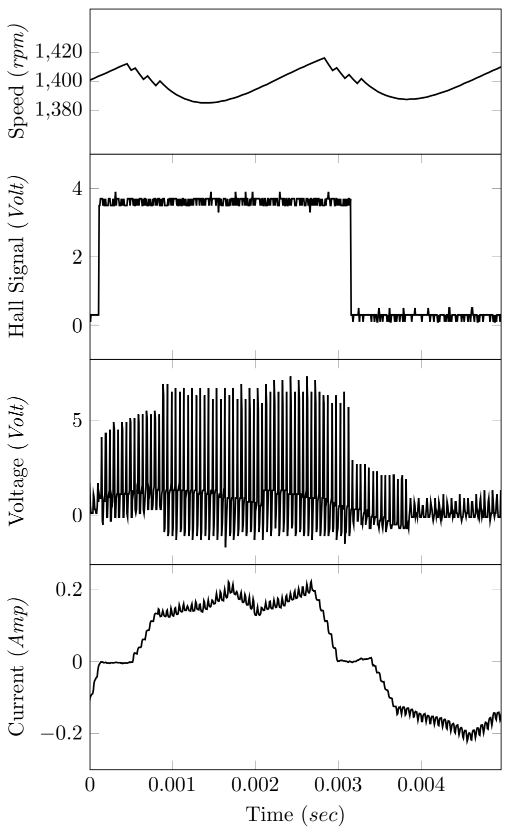
Figure 5.24: Steady state response of a phase of the motor at 100% PWM duty ratio.

5.6.2 Performance Evaluation of the Low Loss Commutation

In order to evaluate the power efficiency of the low switching loss commutation technique, the motor was tested at various rpm ranges generated using different duty ratios. It was observed that as the rpm increases the performance improvement by the low loss commutation technique increases. This is evident due to the increase in the number of switchings during high speed operation. Figure 5.26 presents the comparative current consumption results between classical commutation and the low-loss commutation technique. A maximum of 4% improvement was observed within the entire range of operation. Representative current consumption waveforms at two



(a) Simulation Results.



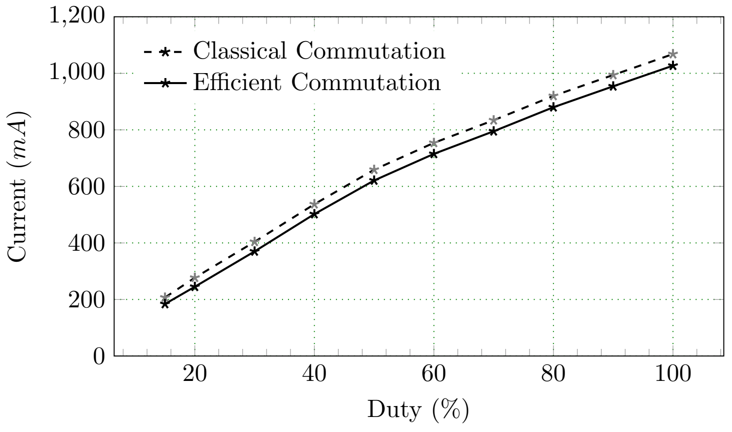
(b) Experimental Results.

Figure 5.25: Steady state response of a phase of the motor at 20% PWM duty ratio.

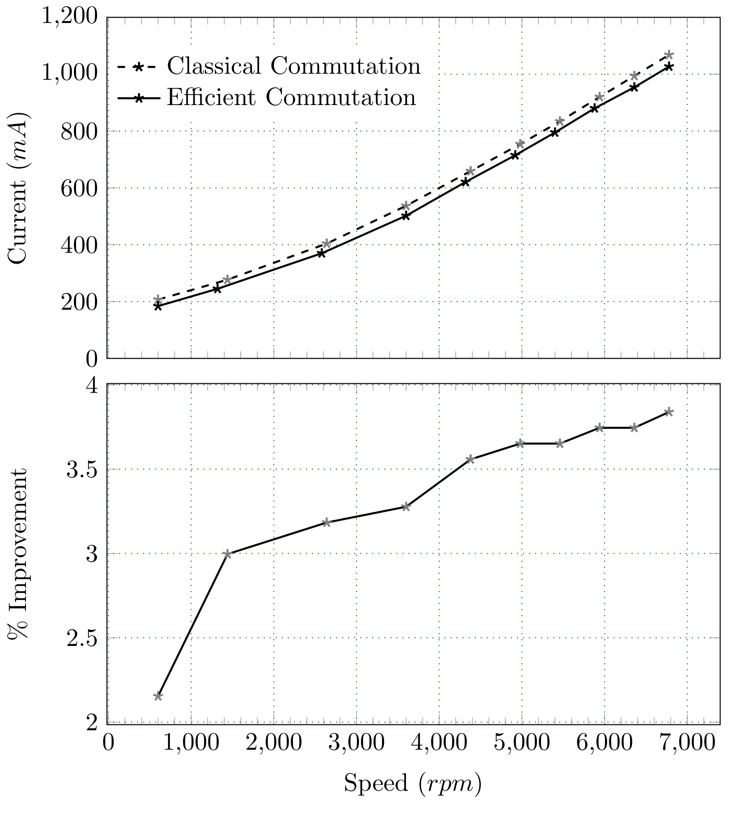
extreme operating speeds are presented in Figure 5.27. It needs to be mentioned that, the percentage of improvement obtained during this study is not an absolute measure of the performance instead a comparative result for this particular study, as this may change with the change of the associated drive electronics.

5.6.3 Performance Results of the Speed Estimators

Before the closed loop control law implementation, the speed estimation technique was implemented in the FPGA hardware. During hardware implementation it was observed that the time based speed estimation M/T-method algorithm requires fewer hardware resources than the observer based speed estimation. This is largely due to

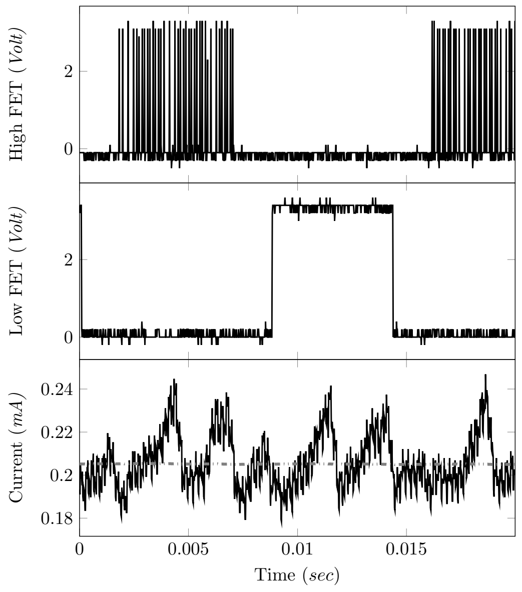


(a) Current consumption when varying duty ratio.

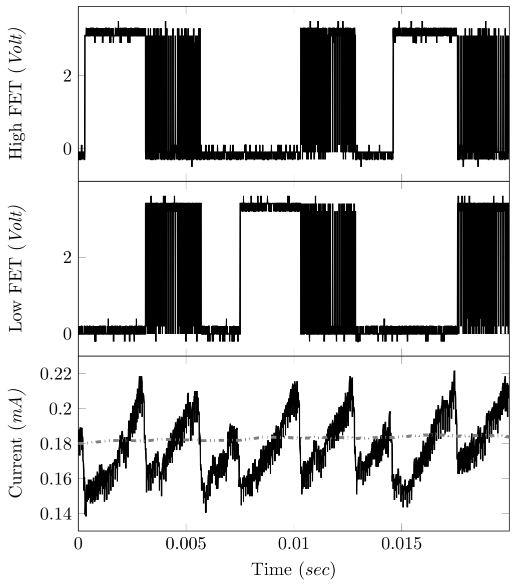


(b) Current consumption when varying speed.

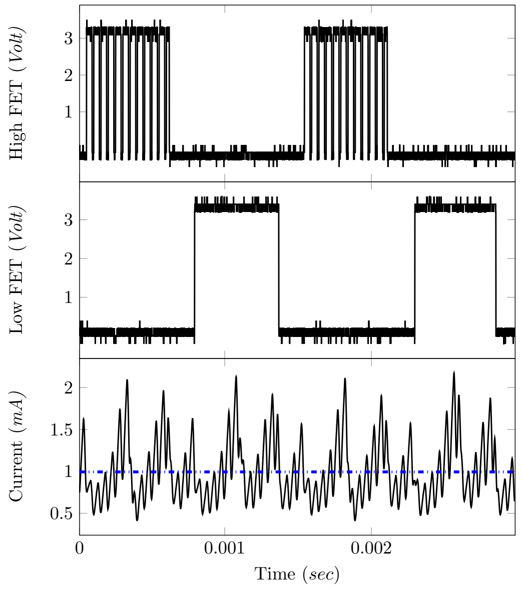
Figure 5.26: Comparing current consumption for the two different PWM based commutation techniques.



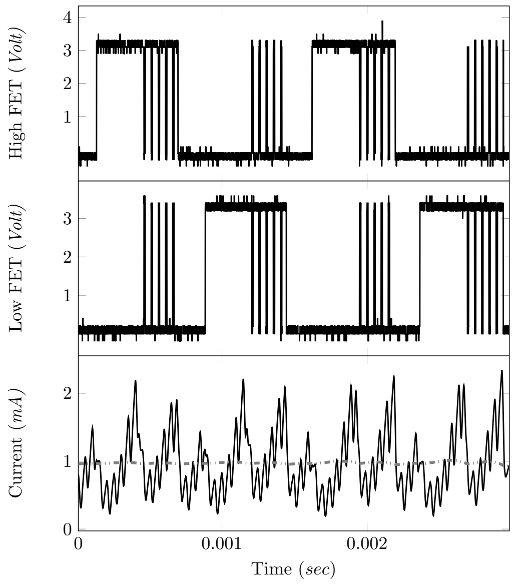
(a) Waveforms of the conventional commutation at 600 rpm



(b) Waveforms of the low loss commutation at 600 rpm



(c) Waveforms of the conventional commutation at 6300 rpm



(d) Waveforms of the low loss commutation at 6300 rpm

Figure 5.27: Representative Waveforms comparing performance between power efficient commutation and conventional commutation technique.

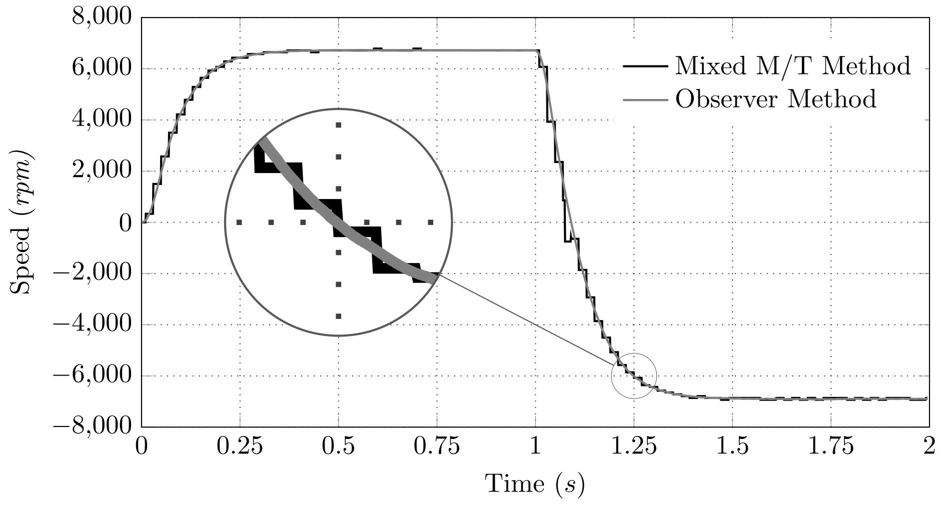


Figure 5.28: Performance comparison between mixed M/T method and Observer based speed estimation method.

the algebraic operations (addition and multiplications) associated with the observer implementation. However, the degraded quantization performance of the M/T speed estimation method offsets the odds of observer based speed estimation. The quantization performance is demonstrated with representative test results presented in Figure 5.28. During steady state speed both speed estimation techniques performed very similarly. However, during transition from one speed setpoint to another speed setpoint, resolution of the measurement fluctuated significantly for the case of the time based speed estimation approach. Hence, the choice between the two speed estimation approaches depends on the available hardware resources and the required quantization performance. Table 5.5 compares the hardware resource consumption by the two speed estimation techniques. Although the observer based speed estimation technique requires same amount of FPGA logic elements the number of multipliers consumed by the M/T method is one third than that of the speed observer method.

Table 5.5: Resource cost of FPGA implementation of the M/T method and Observer method of speed estimation

Estimation Method	M/T-Method	Observer Method
Logic elements	956 (2.4%)	1080 (2.7%)
Multipliers	10 (4%)	34 (14%)

*The utilized resources expressed in parentheses are the percentage of total available resources on an EP3C40F484C6 chip.

5.6.4 Functional Evaluation & Gain Tuning of the Speed Controller

For implementation of the PID controller and its feedback sensor (i.e., observer based speed estimation core) a 16kHz clock was used. Though performance of the controller does not vary after a certain maximum loop update rate due to the limitation in the motor dynamics, it is clear that FPGA hardware can assume any update rate within its capacity, which is defined by the frequency of its master clock (i.e., in this case 50 MHz). A higher closed loop control frequency of the motor controller is critical to utilize the full potential of a quadrotor vehicle or any aerial platform.

The controller was tuned after the discrete implementation of the PID control circuitry. The closed loop dynamics model was used to obtain an initial guess of the tuning parameters of the controller, as well as to evaluate the performance characteristics. The gains were empirically tuned to achieve the fastest settling time with no oscillation, no overshoot, and minimal or no steady state error. It was observed that increasing the proportional gain increased the system response speed, and helped to decrease steady-state error but did not eliminate it completely. Increasing the integral gain reduced the steady state error but tended to introduce oscillation into the system. In this scenario derivative gain resolves the oscillation and obtains a desired control performance. The phenomenon is also reported in [134]. The controller gains are reported in Table 5.6.

Table 5.6: Tuning parameters used in development of the FPGA based speed control module.

Gain Parameters	Values
Proportional Gain	30.0
Integral Gain	0.3
Derivative Gain	0.0

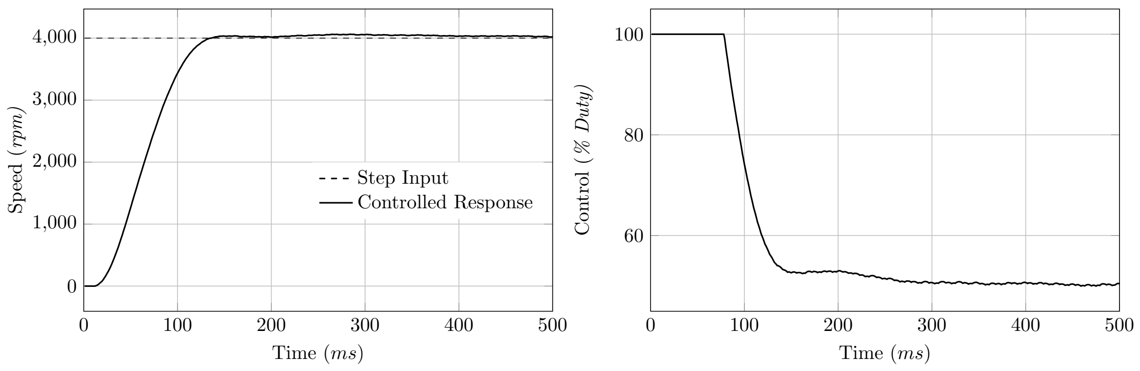


Figure 5.29: Performance of the FPGA based motor speed control system. Left: speed response to a step input and right: controller output.

Once the tuning was completed, the controller was evaluated for a step input. A representative experimental result is presented in Figure 5.29. The desired speed was set to 4000 rpm during this trial. Based on the controlled response, it is evident that the incremental algorithm in FPGA hardware performs similarly to the classical formulation. In addition, the tuning of the incremental PID algorithm did not require contribution from the derivative action. Note that, the step response characteristics parameters (settling time, rise time, steady state error etc.) observed in Figure 5.29 are not an absolute measure of the control performance as they were tuned to observe the behavioural response of the FPGA implemented speed controller.

5.7 Discussion

A 3-Phase motor driver was designed to operate as the actuator of a quadrotor. Due to the nature of the application, a high update rate, power efficient, and closed loop speed controller was required. Therefore, a complete 3-Phase motor motor speed control system was developed and implemented in FPGA hardware.

A complete mathematical model of the 3-Phase BLDC motor was developed with its commutation algorithm and closed loop speed control. Commutation circuitry for FPGA implementation was developed wherein a low switching loss based commutation algorithm was chosen from literature to improve the power efficiency of the speed controller. A maximum of 4% improvement in the power consumption was observed from experimental evaluation of FPGA based low-loss commutation. A discrete implementation algorithm for two different motor speed estimation technique was developed, one of which was FPGA resource efficient with quantization error, and the other consumed high FPGA resource but provided superior performance. An incremental PID algorithm was used to develop the discrete PID hardware circuitry for FPGA implementation. The controller was implemented and operated at 16kHz control loop frequency. The simulated response of the control law was validated using experimental data.

5.8 Conclusion

At typical off-the-shelf BLDC motor speed controller for aerial robots can operate at 400Hz (maximum 1kHz with with expensive versions) and does not offer closed loop speed control. The contribution of this study not only resulted in a motor speed controller that is superior in terms of functionality and update rate but also is power efficient and can provide parallel processing of multiple avionics nodes even in the presence of a large number of actuators and avionic components. In addition, the motor controller can operate at low speed as it was designed based on hall sensor technology instead of back-EMF, which makes it suitable for a large operating range. Further, the flexibility of the FPGA hardware provides a development platform that offers the required parallelism as well as meets the SWaP requirements of small-scale agile UAV platforms.

Chapter 6

Attitude Stabilization of Quadrotor using Active Disturbance Rejection Control Technology

6.1 Introduction

The quadrotor is an under-actuated dynamic system with six degrees of freedom and four input forces (the thrust provided by each propeller). This multi-variable, nonlinear, high-order and strong coupling system imposes severe difficulties for its flight control design [63]. The backbone of this flight controller is the attitude control law, also known as the stabilization controller.

To this date, many control approaches have been taken to stabilize the attitudes of the quadrotor, as presented in Section 2.4.2. The control techniques have ranged from conventional proportional-derivative (PD) [56] control to more advanced techniques, such as backstepping control [59, 60], LQR control [64], feedback linearization [61], etc. depending on the nature of the problem. Particularly, robustness issues can be critical for the quadrotor due to the complex aerodynamic effects (which make difficult the obtainment of an accurate dynamic model), as well as the errors from sensors and the external disturbances (e.g., wind). The first two issues have been studied in the context of linear control and more recently using nonlinear techniques.

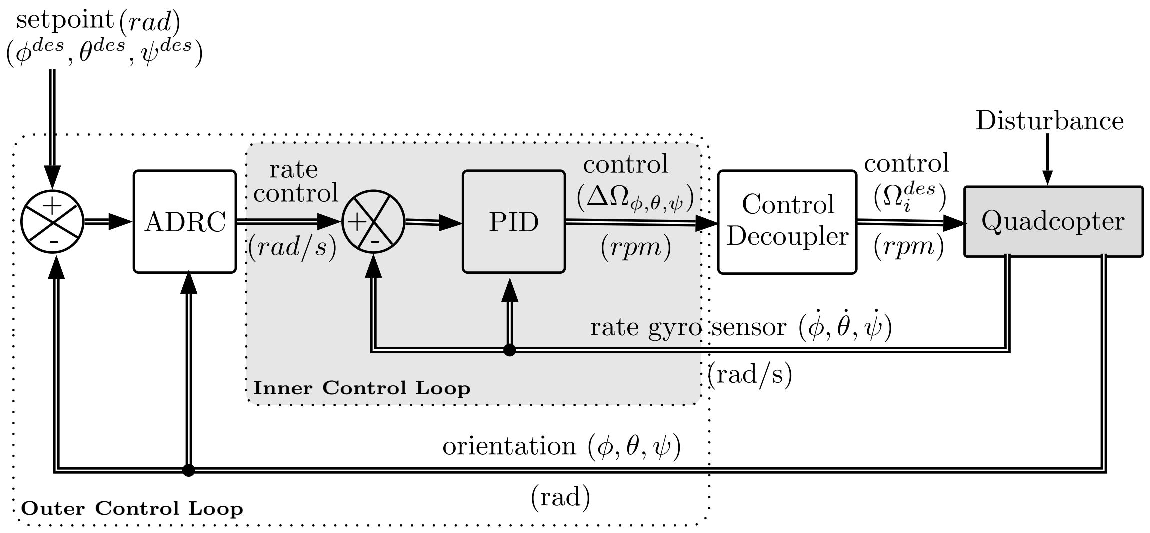
Even though there are so many modern control techniques that have been adopted for the robust stabilization control problem, classical control laws remain a very interesting and effective solution due to their weak dependence on the exact dynamics model of the controlled plant. Also, with the advantage of simple structure, it is easy to achieve the controller in discrete hardware (i.e., micro-computer or micro-controller). However, its disadvantages and deficiencies are still clear when dealing with the control process with remarkable uncertainties or subject to intensive perturbations. To complement this requirement and yet to develop a model independent robust attitude controller, a novel cascaded stabilization control law is proposed in this work. The cascaded controller employs a proportional-integral-derivative (PID) controller in the inner loop and an Active Disturbance Rejection Control (ADRC) control law in the outer loop. The inner loop PID controller controls the angular velocity of the system and the outer loop ADR controller controls the attitude of the system while also rejecting external disturbance.

The ADRC, which was developed by Prof. Han almost three decades ago [10], has unique characteristics to actively reject both internal and external disturbances in real time without the need of an accurate dynamics model. Fundamentally, the ADRC directly estimates the system dynamics and the total disturbances which are extended as a new system state in real time using an Extended State Observer (ESO) and then compensating for them in a linear or a non-linear PD controller. The ADRC has already been applied in fixed wing UAV control research starting from autonomous takeoff of the UAV [71] to auto-landing of the UAV in the presence of wind disturbances [72].

The improvements over previous work presented in this chapter are twofold; first, a cascaded ADRC-PID based robust stabilization control law is developed for the quadrotor attitude control, and second, the stabilization system, including the control law, is prototyped into single chip FPGA based parallel processing hardware.

Since stability of the control system is directly affected by the consistency of the control loop rate and the traditional microprocessor based control implementation fails to effectively maintain a constant control loop rate due to its sequential nature of operation, FPGA hardware is chosen as the prototyping platform for this work. Proper implementation of the control subsystem on FPGA hardware can guarantee a constant control loop rate with only clock skew induced errors [135]. Parallel processing also guarantees that the performance of one module will not be affected by the addition of a second module, which is critical in the case of UAV applications where the flexibility of adding new sensors/actuators technology or improving the existing one becomes necessary to meet mission specific requirements. If necessary, FPGAs also allow for integration of softprocessors for computationally expensive firmware cores. This can be utilized in designing systems using Hardware/Software co-design methodology [136]. FPGA embedded hardware design also meets SWaP requirements leading to a power efficient flight control system and increased flight time.

Section 6.2 describes the robust model independent cascaded control architecture for attitude control of the quadrotor, wherein the inner loop PID controller handles the non linearity of the system and the outer loop ADRC handles the external disturbances and internal noises observed by the system. In Section 6.3 the hardware implementation of the controller is presented. Implementation of the control system is broadly divided into three main stages: Data acquisition (DAQ), control law implementation, and actuator output generation. DAQ is responsible for gathering data from the AHRS module (i.e., inertial sensor). The controller evaluates the control signal based on the sensor feedback information and the actuation module generates the appropriate actuator control signals based on the corrections from the controller. For robustness of the system integration, full hardware implementation of the con-



setpoint (rad)
$(\phi^{des}, \theta^{des}, \psi^{des})$
+
-
ADRC
rate control
(rad/s)
+
-
PID
control $(\Delta\Omega_{\phi,\theta,\psi})$
(rpm)
Control Decoupler
control (Ω_i^{des})
(rpm)
Disturbance
Quadcopter
rate gyro sensor $(\dot{\phi}, \dot{\theta}, \dot{\psi})$
(rad/s)
Inner Control Loop
orientation (ϕ, θ, ψ)
(rad)
Outer Control Loop

Figure 6.1: Block diagram of the proposed Stabilization Control Law based on cascaded control architecture of PID and ADR control laws.

trol architecture is preferred over hardware-software co-design. A tuning method is presented in Section 6.4 that allows the achievement of optimal tuning parameters of the controller. Finally, experimental results are presented in Section 6.5 followed by discussion and concluding remarks in Section 6.6.

6.2 Proposed Cascaded Stabilization Control Law

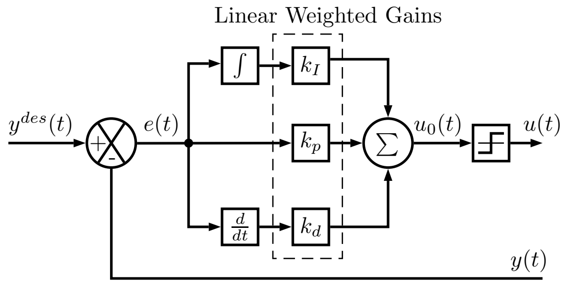
The non-linearity and the under-actuated nature of the quadrotor platform makes the vehicle sensitive to external disturbances and internal noises, which in turn demands a control law that can effectively compensate for the disturbance while handling the non-linear behaviour of the system. In addition, the control laws need to be model independent, scalable, and implementable in FPGA hardware.

In order to address all the requirements with a single solution, a novel cascade control architecture is developed as presented with a block diagram in Figure 6.1. The primary ADRC and the primary dynamics are components of the outer loop, which controls the angular position (i.e., orientation) of the platform while rejecting external disturbances that act on the vehicle. Since the primary controller calculates the set point for the secondary controller loop, the inner loop PID based angular rate controller is also a part of the outer loop. Furthermore, the inner loop controls the fast dynamics of the system (i.e., body angular rates), whereas the outer loop handles relatively slower dynamics (i.e., position control). The assumption allow restraining interaction that can occur between them and improves stability characteristics. Accordingly, a higher gain in the inner loop can be adopted. In addition, the plant nonlinearities are handled by the inner loop controller leaving no meaningful influence of that on the outer loop [137].

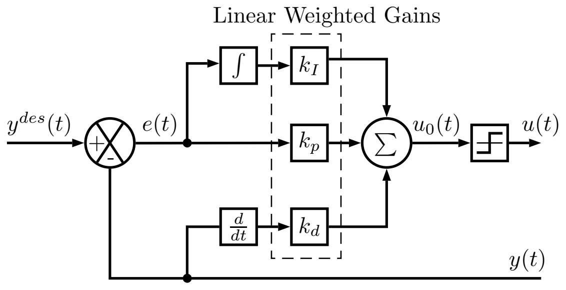
Basically, in the cascade control schema the plant has one input and two or more outputs [138]. This requires an additional sensor to be employed so that the internal fast dynamics can be measured. In this case, the AHRS used in the quadrotor with the rate gyro sensor, measures the angular velocity at a higher rate while estimating attitude information at a lower rate due to the computational complexity associated with the estimation of the orientation through fusion of discrete MEMS sensors (i.e., gyroscope, accelerometer, and magnetometer).

6.2.1 Inner loop PID Control (Body Rates Control)

The inner loop angular rate controller is designed based on a Type-B PID instead of an ideal or conventional PID (Type-A). The problem with the ideal PID controllers is their reaction to a step change in the input signal which produces an impulse function in the controller action [138, 139]. In a typical PID controller, there are two sources of the aggressive controller reaction, the proportional action and derivative action. The idea is to reroute the derivative part from the main path to the feedback path. Discontinuity in desired angular rate input $y^{des}(t)$ will still get transferred through a proportional gain into the control signal, but it will not have so strong effect as if it



(a) PID - Type A.



(b) PID - Type B.

Figure 6.2: Comparison of different PID control loop architecture.

were amplified by the derivative element. This makes it more suitable for practical implementation [139]. A graphical comparison of the two different types of PID is presented in Figure 6.2. Note that, to generalize the control law formulation, the reference input and the feedback input are denoted with y^{des} and y respectively. The formulated control law of inner loop Type-B PID is then:

$$u = k_P(y^{des} - y + k_I \int_0^t (y^{des} - y)dt - k_D \frac{d}{dt} y. \tag{6.1}$$

As near the nominal hover state, the Euler angle rate parameters can be approximated by the body angular velocity parameters (i.e., $\dot{\phi} \approx \omega_x$, $\dot{\theta} \approx \omega_y$, and $\dot{\psi} \approx \omega_z$). The inner loop controller for roll axis takes the form:

$$\Delta\Omega_\phi = k_{P,\omega_x}[\omega_x^{des} - \omega_x] + k_{I,\omega_x} \int_0^t [\omega_x^{des} - \omega_x]dt - k_{D,\omega_x} \frac{d}{dt} \omega_x \tag{6.2}$$

The control law for pitch and yaw orientation take the same form as the roll controller.

6.2.2 Outer loop ADR Control (Orientation Control)

The ADRC allows a user to consider all the unknown and hard to model elements in the system as additional state variables. Both internal uncertainties of plants' dynamics and external disturbances, together referred to as total disturbance, can be estimated using only the system's input and output data. The idea of ADRC is to

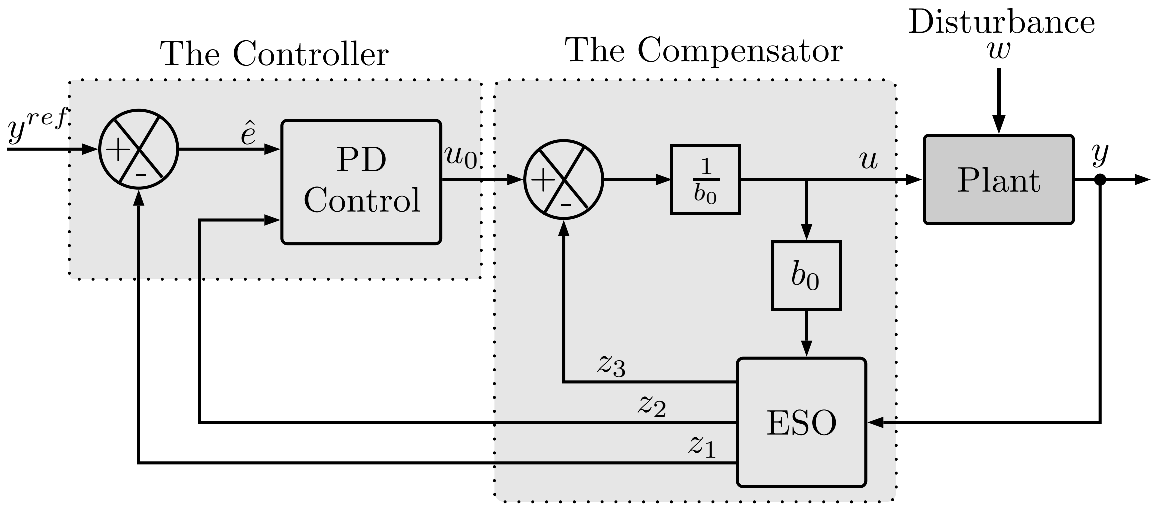


Figure 6.3: Block diagram of the ADRC algorithm, where y^{ref} is the desired setpoint signal.

estimate a supplementary state in real time and then compensate for its effects. Two main modules can be distinguished in the ADRC concept as seen in Figure 6.3:

- *the compensator*, which is responsible for compensating for the effects of total disturbances, estimated by the ESO.
- *the controller*, which is responsible to obtain the desired signal by using the feedback control law.

6.2.2.1 Extended State Observer (The compensator)

Because of the assumption of not knowing a precise mathematical model of the system, made earlier in this chapter, quadrotor dynamics (for motion around one axis) is considered to be a following second order system:

$$\ddot{y} = g(t, y, \dot{y}) + w + bu, \tag{6.3}$$

where y is the output signal, u is the input signal, $g(\cdot)$ is the plant's dynamics (generally unknown), w is the external disturbances, and b is the system parameter.

By merging unknown plant dynamics and external disturbances into one variable, the plant can be rewritten as:

$$\ddot{y} = f(t, y, \dot{y}, w) + bu, \tag{6.4}$$

where $f(\cdot)$ is the *total disturbance*. The system can be defined in state space representation as follows:

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = f(t, y, \dot{y}, w) + bu, \\ \dot{y} = x_1, \end{cases} \tag{6.5}$$

where x_1 and x_2 are state variables. The total disturbance can be obtained by solving the equation:

$$f(t, y, \dot{y}, w) = bu - \dot{x}_2, \tag{6.6}$$

but in order to do that, the value of $\dot{x}_2$ needs to be measured or estimated, which is usually problematic from an implementation point of view.

The ADRC, however, is based on an idea that as long as $f(\cdot)$ is estimated closely in real time, analytical expression of total disturbance is not required. In the ADRC law, a linear Luenberger Observer based ESO is proposed as a method for estimating $f(\cdot)$. To introduce it, the state space model in equation (6.5) is first extended to:

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = x_3 + bu, \qquad x_3 = f(t, y, \dot{y}, w), \\ \dot{x}_3 = \dot{f}(t, y, \dot{y}, w), \\ \dot{y} = x_1. \end{cases} \tag{6.7}$$

The additional element $x_3 = f(t, y, \dot{y}, w)$ estimates total disturbance and is now the augmented state of the system. All the states can be estimated with third order ESO, which is defined as:

$$\begin{cases} \dot{z}_1 = z_2 - \beta_{01}\hat{e}(t), \\ \dot{z}_2 = z_3 - \beta_{02}\hat{e}(t) + b_0 u, \\ \dot{z}_3 = -\beta_{03}\hat{e}(t), \end{cases} \tag{6.8}$$

where β_{01}, β_{02}, β_{03} are observer gains, $\hat{e} = y - z_1$ is the estimation error of state x_1, b_0 is a constant value, an approximation of b from equation (6.3), and z_1, z_2, z_3 are the estimates of x_1, x_2, x_3 respectively.

The control signal (output of the controller - Figure 6.3) in ADRC compensates for the effects of the *total disturbance*, estimated by the ESO. It is defined as:

$$u = \frac{u_0 - z_3}{b_0}, \tag{6.9}$$

where u_0 is the output signal from the feedback controller as seen in Figure 6.3.

6.2.2.2 The Controller

The controller in ADRC is designed to be inherently robust against plant variations. Selection of a proper controller is usually related to a given task. However, a common choice in literature regarding the ADRC is a linear PD controller, which in this case can be considered as a state-feedback controller of quadrotor orientation. Its output signal is defined as:

$$u_0 = k_P\hat{e}(t) + k_D z_2(t), \tag{6.10}$$

where k_P and k_D are the proportional and derivative gains respectively.

6.2.3 Control Decoupler

The goal of the control decoupler is to evaluate the algorithm to calculate the actuation signal for each BLDC motor and decouple the control channels during operation. The control inputs from cascade controllers, about each axis $\Delta\Omega_\phi$, $\Delta\Omega_\theta$ and $\Delta\Omega_\psi$ are therefore combined to generate reference motor control inputs $\Delta\Omega_1^{des}$ to $\Delta\Omega_4^{des}$, for motors 1 to 4 respectively. For ease of reading this chapter, the formulated control decoupling relations are recalled from equation (4.12):

$$\begin{bmatrix} \Omega_1^{des} \\ \Omega_2^{des} \\ \Omega_3^{des} \\ \Omega_4^{des} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Omega_h + \Delta\Omega_{lift} \\ \Delta\Omega_\phi \\ \Delta\Omega_\theta \\ \Delta\Omega_\psi \end{bmatrix}. \tag{6.11}$$

6.3 FPGA Prototyping of the Stabilization System

Although a stabilization system has been implemented in every UAV platform developed in the past, a corresponding FPGA implementation is rare. However, control systems intended for other applications are reported to employ FPGA hardware (e.g., [70, 140]). Since the design philosophy adopted in this work is to encourage efficient utilization of FPGA resources, fixed point arithmetic is chosen over floating point implementation. It should be noted that floating point format generally provides good accuracy and dynamic range; however, comparable performance is also obtainable from a fixed point format, if implemented carefully [140]. In this work, VHDL is chosen as the preferred language for FPGA prototyping. The design also utilizes the IEEE fixed point package [132] extensively. The library is instrumental to minimize quantization error of the control law, while consuming significantly fewer resources required to implement a fixed point algorithm.

6.3.1 Embedded Control Architecture

An embedded flight stabilization system represents one of the most difficult applications in system integration, due to the catastrophic consequences of even the smallest errors. The goal of designing embedded systems is to incorporate a good separation

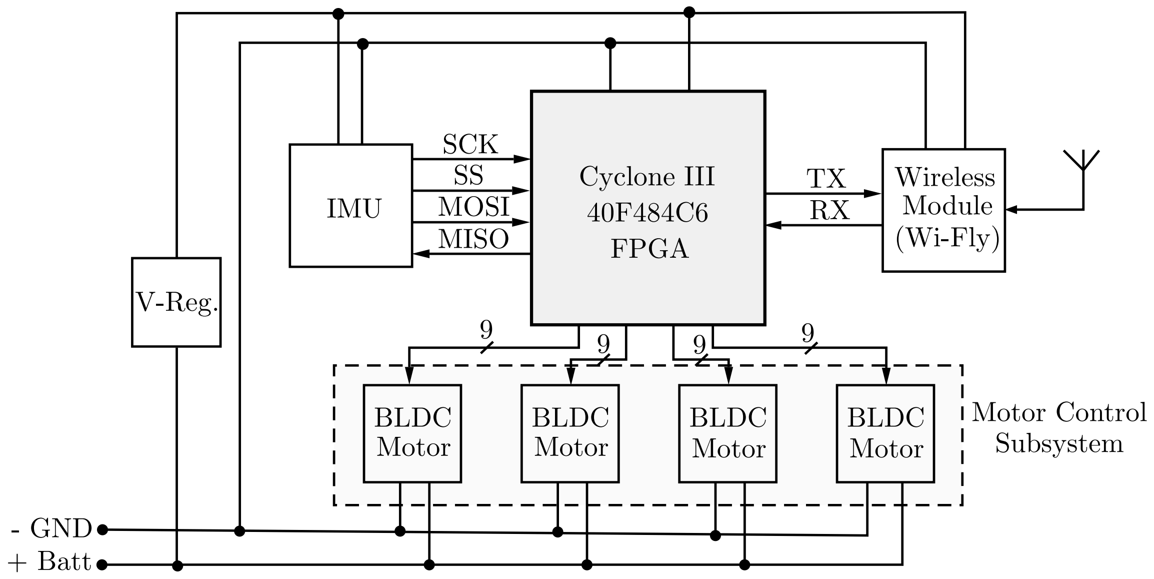


Figure 6.4: Schema of the Embedded Control System.

of the system's functionality into hardware as well as an efficient and correct implementation of the hardware components and their integration into a complete system. A schema of the embedded control system in shown in Figure 6.4.

The proposed stabilization control system integrates an AHRS module as the exteroceptive sensor, a Wi-Fi to serial UART (RS-232) module to collect pilot command, and four actuation modules to operate brushless DC motors. A single chip Cyclone III (40F484C6) FPGA chip based hardware was adopted to embed the entire stabilization system that includes the interfaces for sensor suites and the computational elements associated with the control laws. The custom FPGA hardware is described earlier in this thesis (see Chapter 3).

Feedback interfacing of the controller consists of angular positions and angular rates from the AHRS. Command from the pilot includes the desired orientation of the platform. A Serial Peripheral Interface (SPI) protocol was used to develop the communication interface between the FPGA hardware and the AHRS module while a serial RS-232 UART protocol is used to collect pilot information via an off-the-shelf

Wi-Fi module (i.e., Wi-Fly). Here the Wi-Fi signal acts as the carrier of the UART data only.

6.3.2 Firmware Design

At the center of the firmware, there is a master control core that manages the processes required to support the tasks associated with the stabilization control system. Real-time sensory information and pilot command are captured through a data acquisition core that consists of the SPI communication protocol and the serial RS-232 communication protocol. Three cascaded control cores evaluate the required control signal for the stabilization of the platform, wherein each core is employed to stabilize angular motion about one rotational axis. The control signal is then fed to the actuation core to calculate the motor speeds required to achieve the desired orientation and hence the stabilization of the platform. Figure 6.5 presents the firmware design layout of the proposed embedded controller.

6.3.3 Implementation of the Master Control Core

Though FPGA hardware is well suited for parallel computation, sometimes it becomes necessary to implement controllers that operate in a sequential pattern. One example is the Master Control Core of the stabilization system. The master control core dictates the sequential routine that will activate the correct subroutines at any given time of operation. The subroutines within the main routines can be executed either in sequence or in parallel. Since the states in the master control core are dependent only on previous input and the previous state, a Moore Finite State Machine (FSM) [141] is chosen to implement the master controller into the FPGA hardware. A functional FSM block diagram of the master controller is presented in Figure 6.6.

At power on, the motor driver core (i.e., Actuation Core) is initiated with a nominal speed ($<< \Omega_h$), just enough to spin all four motors to verify its functionality

.

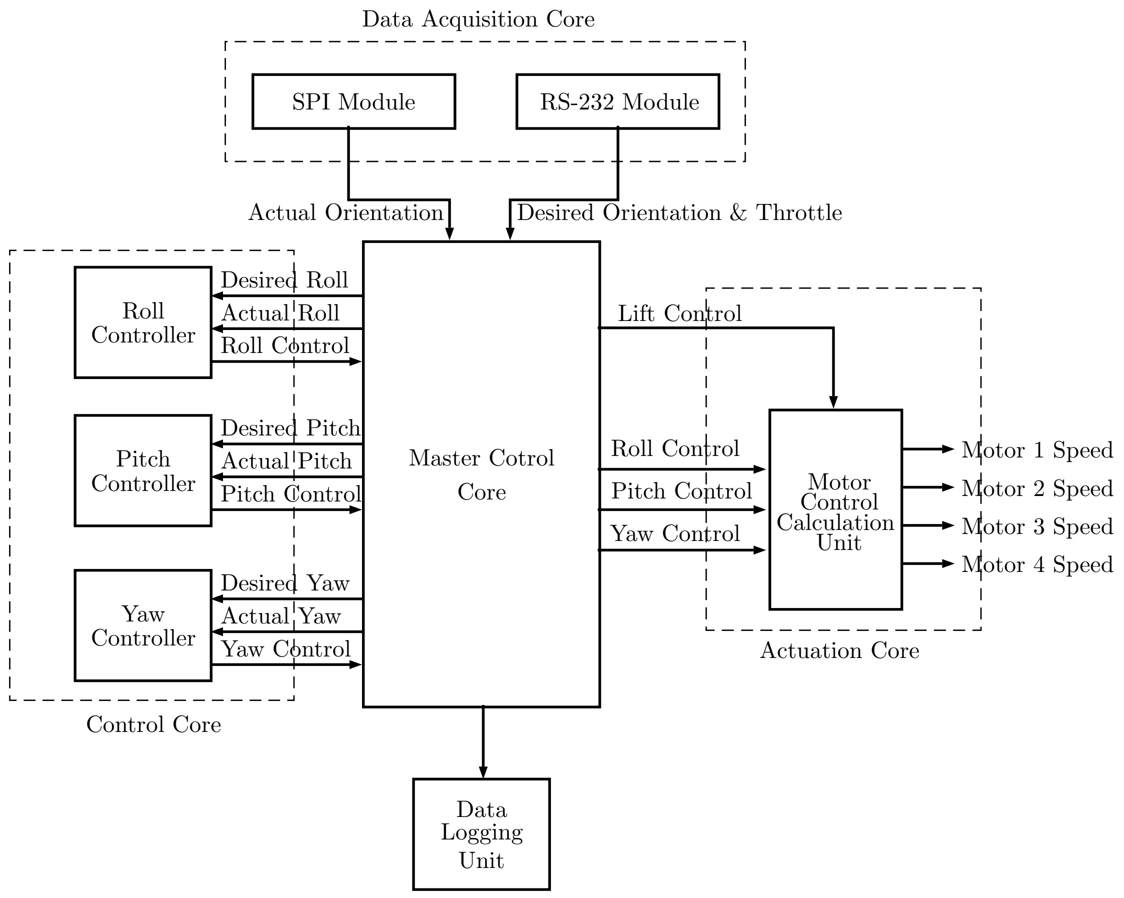


Figure 6.5: Firmware architecture of the FPGA based stabilization system.

and the direction of rotation. At this state, there is no attitude control command generated by the stabilization unit. After the startup of the system, the state machine begins with the setup and initialization of the registers that will be used for the execution of the FPGA cores/modules. Then, the input signal detection module is enabled. The event is triggered at a fixed interval defined by the FSM loop frequency. In this case, a 50 kHz clock is used to operate the state machine. Note that each state of this state machine is an FPGA hardware core; therefore, each one has its own clock to operate its respective hardware circuitry. Before the state machine enters the main loop (i.e., stabilization control states), it checks for a valid signal incoming from the receiver, verifying the length of a predefined pulse; if no signal is present, the state

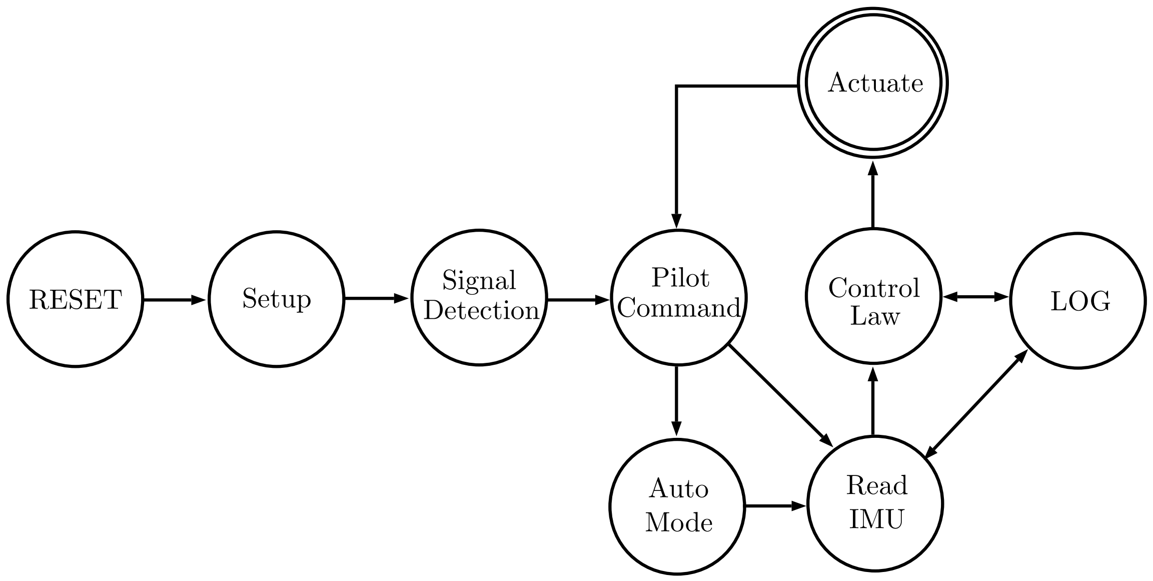


Figure 6.6: Master Controller based on Finite State Machine.

machine waits for a valid signal. For safety strategy, when a valid signal is recognized, the state machine remains waiting until the pilot throttle/propeller thrust level is set to its minimal value or auto-flight mode is selected. Once the state-machine exits of safety loop and pilot command loop, the main loop execution begins.

The main loop consists of five states. The first state reads the angular position and angular rates from the AHRS module. Once the attitude feedback is obtained, the second state takes place immediately to compute the control law to generate required motor speeds. The third state reloads the motors speed as soon as the control command is updated while the fourth state logs all the avionics information. This state is independent of the rest of the states in the FSM and is activated or deactivated only on demand by the pilot.

6.3.4 Data Acquisition Core

Since the off-the-shelf AHRS module (i.e., CHR-UM6) is designed to work as a slave in the SPI communication bus, a master SPI controller is designed and implemented

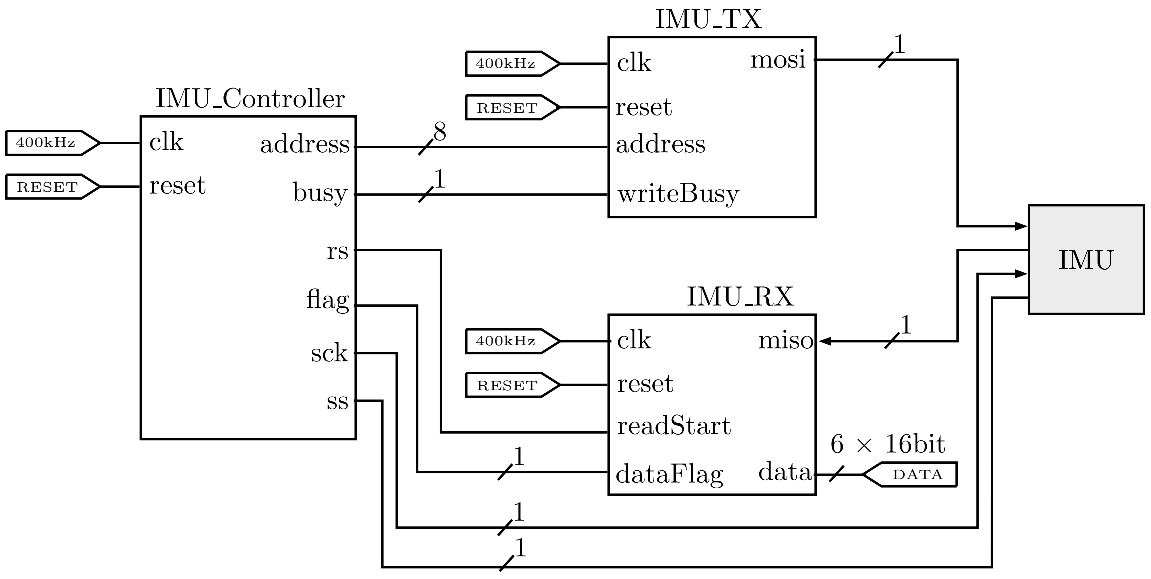


Figure 6.7: Architecture of IMU communication.

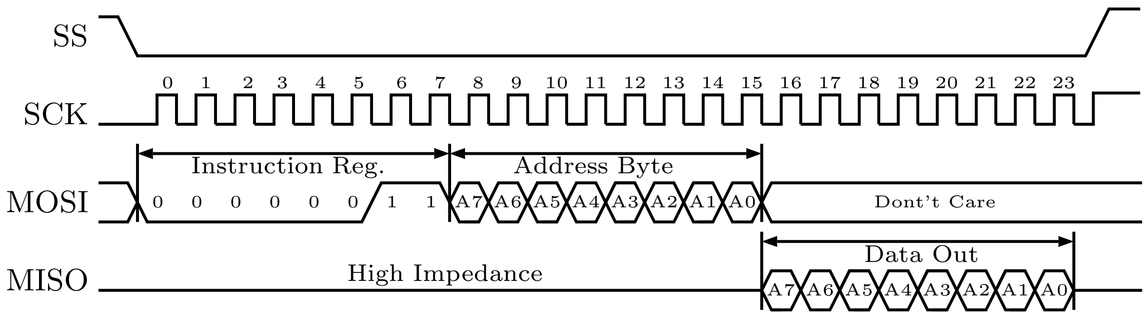


Figure 6.8: Typical SPI Timing diagram.

in the FPGA hardware. The master controller bus interface consists of four logic signals; namely, Serial Clock (SCK), Slave Select (SS), Master Out Slave In (MOSI), and Master In Slave Out (MISO). A block diagram of the AHRS interface is presented in Figure 6.7. First a generic 1-byte SPI receive and transmit module is implemented according to 'mode 0' SPI protocol as presented in [142]. In order to read the interesting sensory information from the AHRS, the timing diagram of the communication protocol is redrawn in Figure 6.8.

The instruction register in the timing diagram carries the information to define

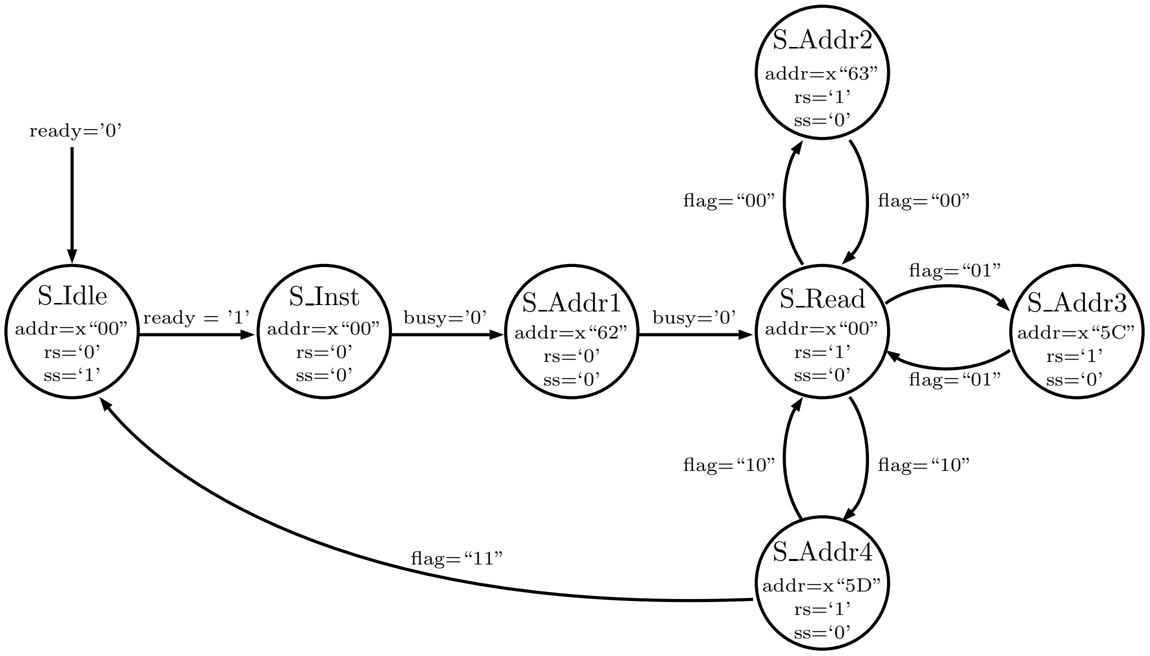


Figure 6.9: State Machine of the SPI protocol.

the mode of operation (i.e., read or write). In order to request a sensor reading, the respective register address is written in the MOSI line followed by a read operation to acquire the requested data in the MISO line. While a read operation is in progress the address byte is set to x"11". A Mealy FSM is designed to collect six sensory readings (i.e., 3-orientation data and 3-angular rate data) embedded in four 32-bit data registers.

The FSM is presented in Figure 6.9. The state machine is composed of four co-states, namely, S_Idle, S_Inst, S_Addr, and S_Read. The state machine triggers at 400kHz frequency, the same as the SPI clock frequency (SCK). Each state controls three signals: address (*Addr*), slave select (*ss*), and read start (*rs*). Initially, after the module is reset, the FSM enters into the S_Idle state to initiate the output signals while waiting for a ready signal (*ready*) to move to the next state. Once the *ready* signal is asserted high, the state is updated to S_Inst and the slave wakes by pulling *ss* signal to low. The master SPI then proceeds with the *addr* signal transfer using the SPI transmit subroutine: transmitting each bit (MSB to LSB) on MOSI on the negative edge of SCK. During this write operation the states are not updated. The status of the transmission is monitored using a *busy* signal. The transmission is completed with the *busy* low signal, which is captured to update FSM to S_Addr1 state and the *addr* is overwritten with the address of a sensor data register (e.g., x“62”). The address is then transmitted via the transmit module. The FSM enters into S_Read state after *addr* is completely transferred via the MOSI line. In the S_Read state, the *rs* signal is asserted high, which in turn activates the SPI receive module. During the read operation the sampling for reading incoming bits in MISO continues until the complete incoming data is read. Similar to the transmit operation, the states are not updated during a read operation. The process is repeated three more times and the state flow between address update (S_Addr) and read state (S_Read) is controlled using a $flag$ signal. The $flag$ maintains the sequence of the read register and a counter is employed in each S_Read state to keep track of the bytes' segments during a read operation, so that the two independent 16-bit sensor data can be identified from a 32-bit incoming signal.

In addition to the AHRS data acquisition module, the pilot command input module was implemented using an existing RS-232 FPGA core developed by Pong Chu [143]. A custom data packet of 80-bit data length containing pilot command information (i.e., orientations (3×16 bits), hover speed (16 bit), a header (8 bit), and a footer (8 bit)) is transmitted from the ground control station to define reference signals for the stabilization controller.

6.3.5 Cascaded Attitude Control Core

The cascaded attitude control core is essentially the discrete implementation of both PID and ADRC control laws presented in equations (6.1), (6.8), (6.9), and (6.10). To encourage modular FPGA prototyping of the control system, the core is designed to support only one of the three orientation axes; therefore, three cores are required to complete the stabilization control system. Each control law is reformulated to support discrete implementation in the prototyping hardware that uses basic logic operations comprising only adders and multipliers.

6.3.5.1 Discrete implementation of type-B PID

The PID control law presented in equation (6.1) requires computation of past errors, e(0)..e(t), which have to be stored. This can lead to large FPGA resource consumption by only a single PID module. Therefore, an alternative recursive algorithm is characterized by the calculation of control output $u(t)$, based on $u(t-1)$ and a correction term $\Delta u(t)$. To derive the discrete recursive algorithm, first the control law is translated into a difference equation using the discretization method [130]:

$$\begin{aligned} e(t) &= y^{des}(t) - y(t), \\ u(t) &= k_p e(t) + k_i \sum_{j=0}^{t} e(j) - k_d \Big[y(t) - y(t-1) \Big]. \end{aligned} \tag{6.12}$$

Next, $u(t-1)$ is calculated based on equation (6.12):

$$u(t-1) = k_p e(t-1) + k_i \sum_{j=0}^{t-1} e(j) - k_d \Big[y(t-1) - y(t-2) \Big] \tag{6.13}$$

Then calculate the correction term as:

$$\begin{aligned} \Delta u(t) &= u(t) - u(t-1) \\ &= (k_p + k_i) e(t) + k_p e(t-1) + k_d \Big[y(t-2) - y(t) \Big] \end{aligned} \tag{6.14}$$

Equation (6.14) is called an incremental algorithm and the control output is calculated as:

$$u(t) = u(t-1) + (k_p + k_i) e(t) + k_p e(t-1) + k_d \Big[y(t-2) - y(t) \Big] \tag{6.15}$$

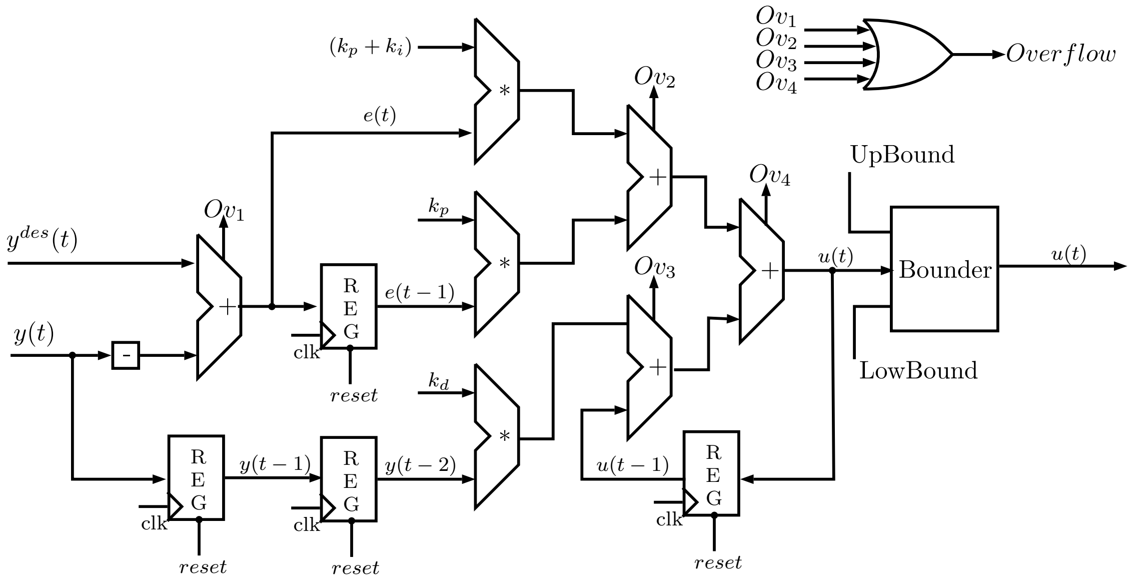


Figure 6.10: Hardware design of Incremental Type-B PID.

The incremental algorithm can avoid accumulation of all past errors and can achieve smooth switching from manual to automatic operation [130].

Using four adders, three multipliers, and four registers, the incrementation form of the PID control law is prototyped into the FPGA hardware. The hardware design is presented in Figure 6.10. All bond signals in the figure are I/O ports, while others are internal signals. The clock signal *clk* is used to control the sampling frequency of the controller. The feedback signal $y(t)$ and the $y^{des}(t)$ represent the current angular velocity feedback value and the reference angular velocity input respectively. The negation of $y(t)$ is generated by bit-wise complementing and adding 1. At the rising edge of the control, signals $e(t)$, $y(t)$, $y(t-1)$, and $u(t)$ of the last cycle are latched at their respective registers REG, to generate $e(t-1)$, $y(t-1)$, $y(t-2)$, and $u(t-1)$ respectively. The registers can be set to the initial value of 0 by asserting the reset signal, *reset*. Control can become unsteady and fail in the event of an overflow in any of the adders. Ov_i is asserted in the case of an overflow in i adder. All overflow signals are read together to generate the $Overflow$ signal. When asserted, this signal can be used to reset the controller. A bounder logic circuit is incorporated to prescribe saturation of the control output using a user defined range of *UpBound* and *LowBound*.

6.3.5.2 Discrete implementation of ADR Controller

Similar to [144], a linear ESO as in (6.8) is adopted for the discrete implementation of the disturbance estimation observer.

$$\begin{aligned} \hat{e}(t) &= z_1(t) - y(t), \\ \sigma(t) &= T(z_3(t) - bu(t)), \\ z_1(t+1) &= z_1(t) + Tz_2(t) - \beta_{01}\hat{e}(t), \\ z_2(t+1) &= z_2(t) + \sigma(t) - \beta_{02}\hat{e}(t), \\ z_3(t+1) &= z_3(t) - \beta_{03}\hat{e}(t). \end{aligned} \tag{6.16}$$

Here T is the sampling period of the discrete ESO. Similar to (6.16), the compensator and the PD controller are discretized as:

$$\hat{u}_0(t+1) = \{k_P\hat{e}(t) + k_D z_2(t) - z_3(t)\} \times (\frac{1}{b_0}). \tag{6.17}$$

Since equations (6.16) and (6.17) involve only multiplication and addition operations, implementing them on the FPGA is similar to the approach presented for type-B PID implementation. However, the inverse of b_0 in ADRC is hard-coded into the FPGA hardware as a constant instead of an inversion logic implementation, which saves both hardware logics and compilation time.

6.3.6 Actuation Control Core

The actuation control core derives the desired rotational speed signal for the BLDC motors. The core comprises a generic control decoupler and two BLDC motor drivers cores; a comprehensive study of the BLDC driver core along with its development is presented earlier in this thesis (see Chapter 5). In order to implement the control decoupling equations (6.11), algebraic formulations based on adders and multipliers are chosen over matrix representation.

$$\begin{aligned}
\Omega_1^{des} &= \Omega_h + \Delta_{lift} - \Delta\theta - \Delta\psi, \\
\Omega_2^{des} &= \Omega_h + \Delta_{lift} + \Delta\phi + \Delta\psi, \\
\Omega_3^{des} &= \Omega_h + \Delta_{lift} + \Delta\theta - \Delta\psi, \\
\Omega_4^{des} &= \Omega_h + \Delta_{lift} - \Delta\phi + \Delta\psi.
\end{aligned} \tag{6.18}$$

6.3.7 Complete Stabilization Core

The data acquisition core, the control core and the actuation control core are interconnected through their peripherals and a complete stabilization system is developed. A data-flow diagram for one axis attitude control module is presented in Figure 6.11. Each hardware circuit is operated with its respective driving clock derived from a 50 MHZ clock and PLL clocks are used to design the scaled clocks. Though the data acquisition circuit is operated with a 400 kHz clock to support the SPI communication protocol, the angular orientation output is available at 500 Hz and the angular rate data is updated at 1 kHz as designed by the sensor manufacturer. The pilot command core is driven by the same clock that drives the FPGA hardware at 50 MHz. However, the baudrate for serial RS-232 communication is set to 11,5200. The inner loop and the outer loop attitude controllers are set to operate at 5 kHz and 500 Hz respectively. The control decoupler is not driven by any clock; instead, it is implemented in pipelined architecture which is evaluated at the FPGA main clock at 50 MHz. Therefore, it outputs the desired motor speed as soon as the control signal is available at the input. The desired motor speed is directly fed to the input of the motor driver core. Note that the motor driver cores presented in the figure are obtained from earlier development work presented in this thesis.

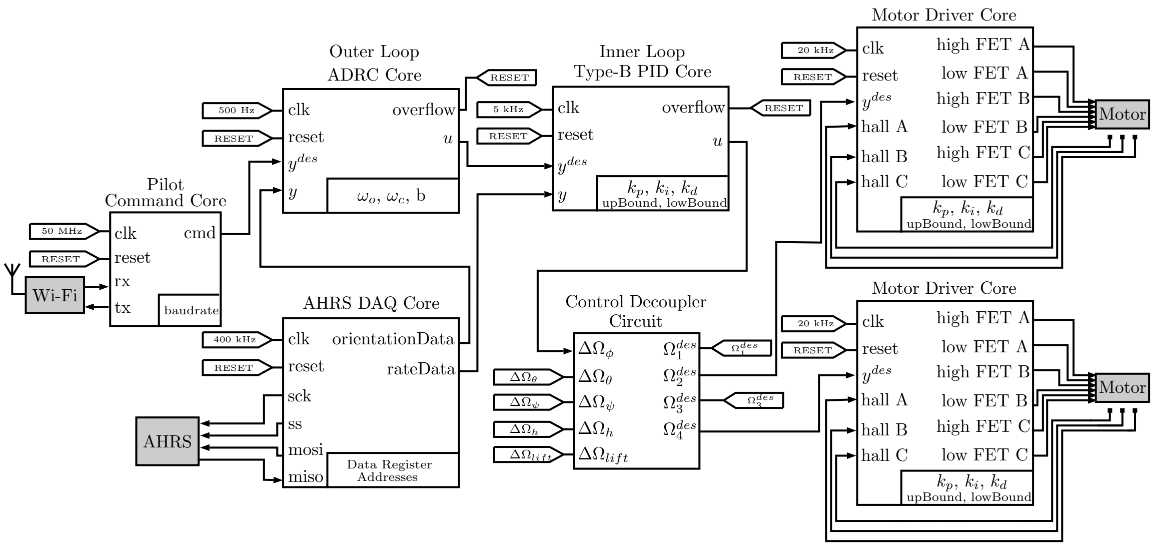


Figure 6.11: Dataflow diagram of the stabilization core implemented to control roll axis motion.

6.4 Tuning of the Controller

In order to find the tuning gains for the observer of the outer loop ADRC, various analytical techniques can be used. One example is a pole-placement method used in [145]. For the purpose of simplification, the poles of the characteristics equations are placed in one location (ω_o) and the observer gains for the discrete ESO are calculated in [146] as follows:

$$\begin{aligned}
\beta_{01} &= 1-\beta^3, \qquad \beta = e^{-\omega_o T}, \\
\beta_{02} &= (1-\beta)^2(1+\beta)\frac{3}{2T}, \\
\beta_{03} &= (1-\beta)^3\frac{1}{T^2}.
\end{aligned} \tag{6.19}$$

Here, $\omega_o > 0$ is the bandwidth of the observer. A large observer bandwidth improves the convergence speed of state estimation of the ESO, which is achieved at the cost of degraded noise tolerance. Thus, ω_o is tuned to balance tracking performance against the noise sensitivity of ESO [70]. In addition, the tuning procedure of the system parameter b in (6.16) is also provided in [145].

The PD controller gains associated with the outer loop ADRC can also be expressed as a function of controller gains ω_c as used in [147]. According to [145] the controller gains in (6.17) are determined as:

$$k_P = \omega_c^2, \qquad k_D = 2\omega_c. \tag{6.20}$$

A larger controller bandwidth ω_c demands high dynamics from the system, so that the response can rapidly track the set-point. However, this may lead to undesirable oscillations or instability because the system may be marginally capable or even completely incapable of delivering the demanded dynamics. Hence, the tuning exercise of ω_c attempts to find an acceptable compromise between the requirements of performance and stability margin [148] while also obtaining a critically damped response of the controlled state of the vehicle.

The inner loop PID controller is tuned to asymptotically converge the controlled response to the steady state with a greater convergence speed. A high value of k_P in (6.15) improves the convergence speed of the response but may introduce oscillations which can be resolved through tuning the k_D gain.

6.5 Experimental Results

The experimental platform consists of a quadrotor equipped with a custom FPGA board based on the Cyclone III FPGA chip that is running at 50 MHz. Also, the quadrotor has an AHRS module (CHR-UM6) that computes its attitude information at 500 Hz and its angular rates at 1000 Hz. In order to evaluate the performance of the control law and its disturbance rejection capabilities a 1-DoF test-bench was developed using 80/20 aluminum as framing material. The setup only allows angular motion about one of the rotational axes. To generate external disturbance a pneumatic air-gust generating system was integrated into the experimental setup. A

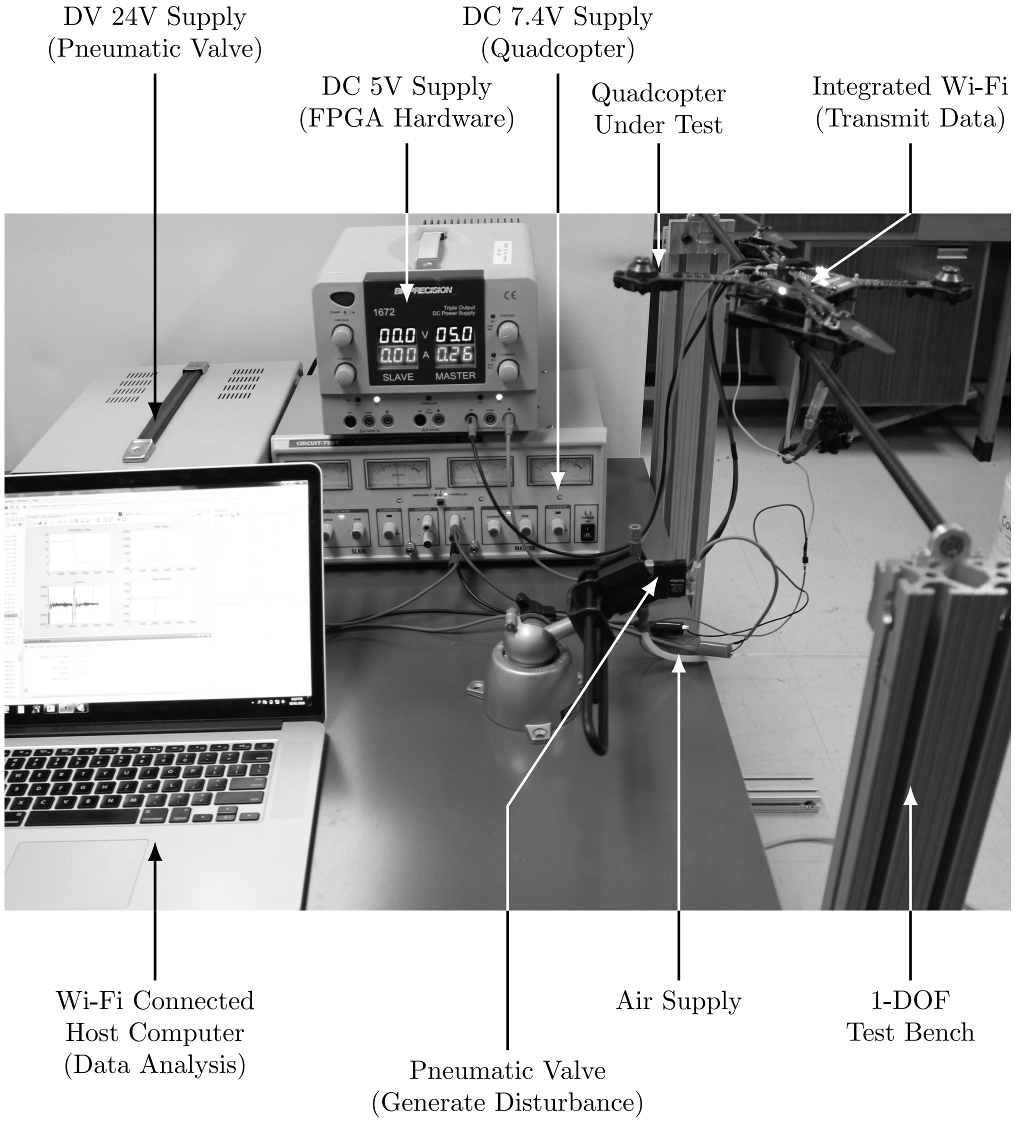


Figure 6.12: Experimental Setup used to evaluate FPGA based ADRC-PID cascaded control law.

pneumatic solenoid valve is used to trigger an air-gust impulse to the plant. The valve is controlled by the same FPGA hardware that holds the stabilization control core. A pictorial representation of the experimental setup is presented in Figure 6.12. A host computer was interfaced with the FPGA through a Wi-Fi based virtual serial link for experiment control and data acquisition. After a new set-point angular position was

Table 6.1: Tuning Parameter for the controllers

Parameter	Value			Unit
	Inner PID	Outer PID	Outer ADRC	
k_P	20.0	15.0	-	-
k_I	0.0	0.008	-	-
k_D	50.0	450.0	-	-
ω_o	-	-	100.0	rad/s
ω_c	-	-	18.0	rad/s
b_0	-	-	20.0	-
T	0.001	0.002	0.002	s

initiated, the FPGA logged interesting controller variables at every 100 ms (1000 Hz). The logged data was sent to the host computer for storage and further analysis.

6.5.1 Performance Evaluation

The performance of the proposed cascaded controller was benchmarked against the widely used PID technology. Only the outer ADR controller was replaced with a PID controller during the comparative study while the inner PID loop remained unchanged. Experiments were conducted to validate the performance for a nominal plant and a perturbed plant. Both control methodologies (ADRC and PID) were first tuned well for 1-DOF motion (other motions are blocked as seen in Figure 6.12) wherein obtainment of asymptotic stability with the fastest possible convergence was the criterion of a well tuned controller. The tuning parameters of the PID controller were chosen empirically keeping the goal in mind, while the tuning of the ADRC relied on the performance of position and velocity tracking by the state observer (ESO). Table 6.1 shows the chosen tuning values. It needs to be noted, though the quadrotor is cabale of free-flight, but for simplicity and for sake of comparitive analysis this 1-DOF controlled experiment is preferred. In addition to that, the orientation controller actuates the motor using the drivers developed in Chapter 5. During the tuning exercise

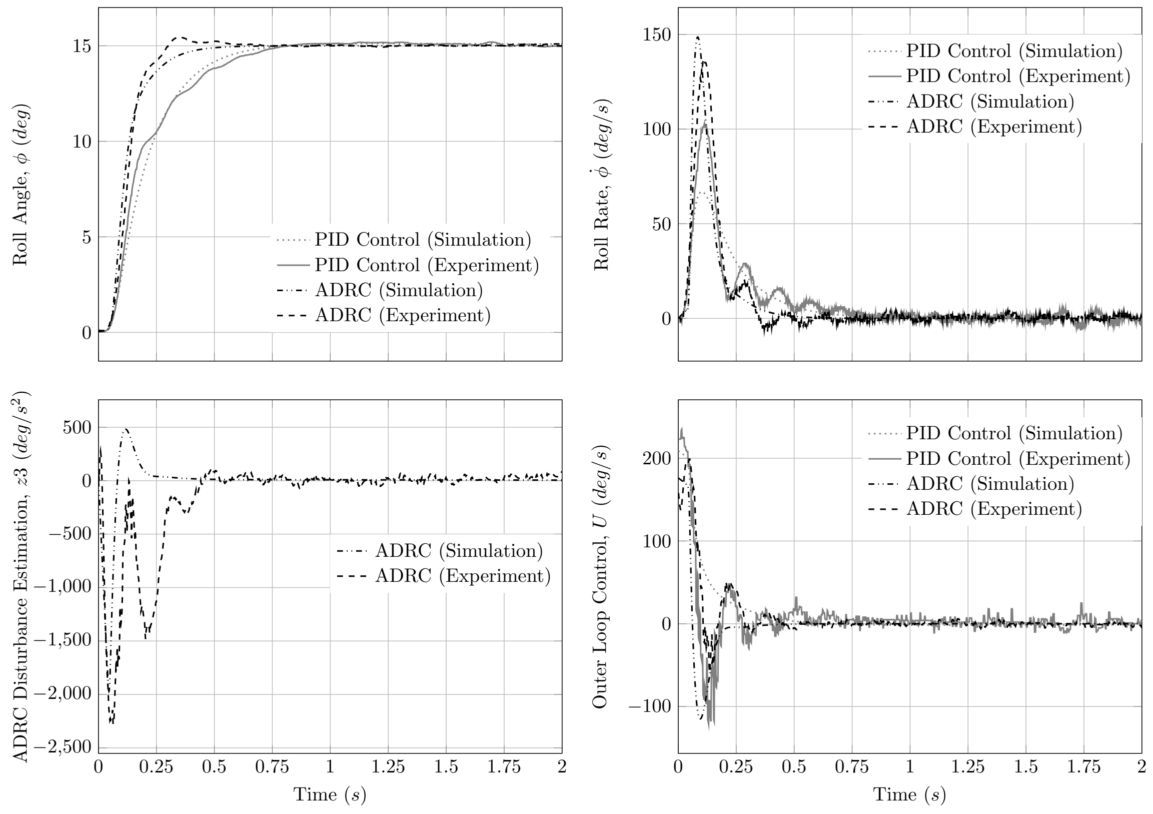


Figure 6.13: Comparison of the controller response for a reference step input of 15° in roll axis.

the control response for a defined step input of 15° starting from 0° was taken as the reference motion. Figure 6.13 shows a representative control response for both PID and ADRC. Though the controllers implemented here are model independent and a practical assumption was made at the beginning that the mathematical model of the controlled plant is unknown, for the sake of completeness the experimental result was verified with the simulation results obtained using dynamics model developed in [149]. Note that, though the simulation model presented earlier in Chapter 4 is comprehensive, it has not been used in this case due to the high frequency modes introduced by the parasitic elements of the model, which also increases run time of simulation experiment.

Experimental results presented in Figure 6.14 demonstrate that, a well tuned

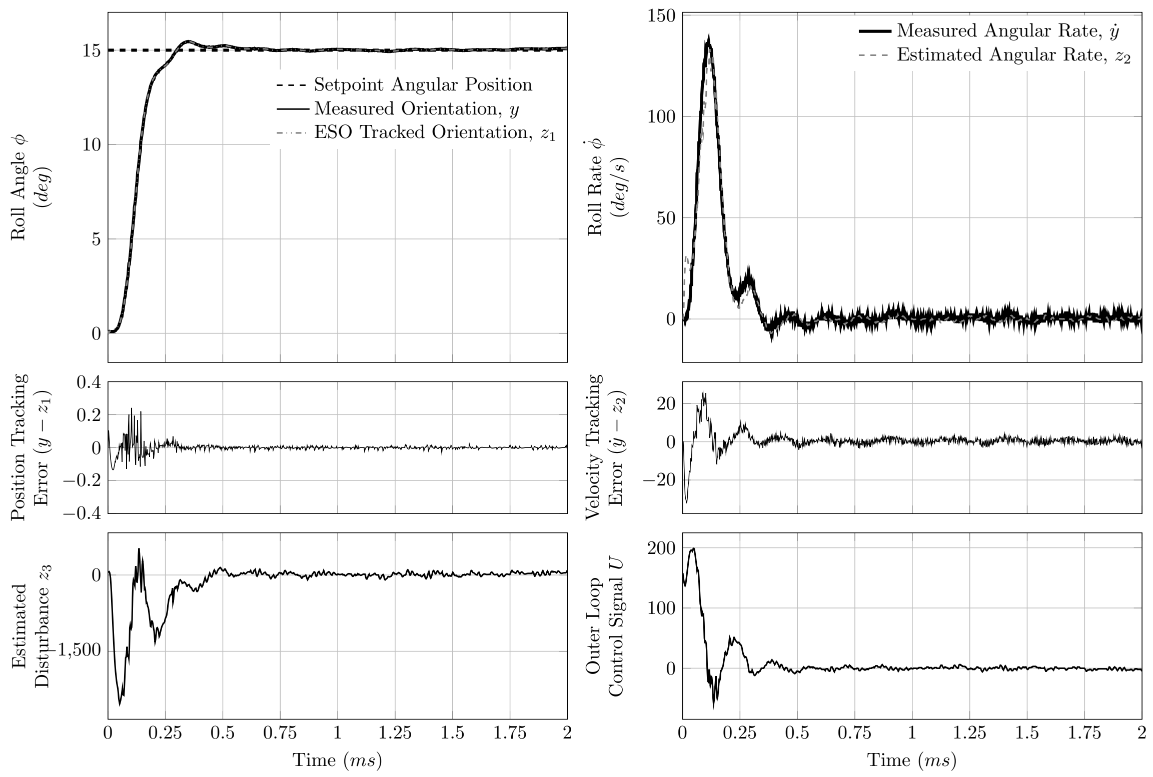


Figure 6.14: Tuning Performance of ADR controllers.

ADRC can reach a steady state faster compared with the well tuned PID. Both responses present a non-oscillatory response but the ADRC introduces an insignificant overshoot while reaching a steady state. This overshoot can be attributed to the tuning of the ESO, where velocity tracking during transitions degrades as seen in the figure. This can be resolved using a non-linear ESO, which is left for future improvement as this work only focuses on the FPGA implementation and comparative analysis of the ADRC and PID during quadrotor attitude stabilization.

In order to further evaluate the control performance an experiment consisting of 10 predefined step-input trials was conducted. The trial cases were limited to within an assumed practical operating range of 0° to 20° of the roll axis. Due to the limitation of experiment automation only 10 representative trials were considered, unlike in a Monte Carlo experiment with a large number of random trials. Settling times (t_s)

Table 6.2: Performance of the controllers

Statistics	Outer Loop ADRC		Outer Loop PID	
	t_s (ms)	e_{sse} (deg)	t_s (ms)	e_{sse} (deg)
Mean	426.48	0.053	1146.4	0.080
Max	512.76	0.109	1712.9	0.166
Min	367.01	0.011	695.1	0.046
σ	55.947	0.0299	391.8	0.040

and absolute steady state errors (e_{sse}) of these maneuvers quantified the performance of the controllers. The settling time is provided by the time the system response has entered and remained within ±2% of the desired step input value (desired orientation). A steady state was detected based on moving variance; when the moving variance of a window consisting of 100 samples was observed with less than the variance threshold of 0.1° the response was considered to have a steady state. The corresponding absolute steady state error was calculated by taking the mean of all absolute steady state errors for samples from the time when the system entered a steady state to the time when the trial ends. Each trial during this experiment was designed to last for 2000 ms, assuming that the system would reach a steady state within that period of time. Finally the experimental data were analyzed to determine the performance matrices presented in Table 6.2. The control responses featuring three representative trial maneuvers from the experiment are presented in Figure 6.15. It is evident from the statistics and from the response curves that for the given performance matrices the ADRC performs better than the PID controllers over the defined range of motion.

To this end, the controller was tuned for the nominal plant and evaluated for settling time and steady state error. Without retuning the control gain parameters, each control technology was evaluated for the perturbed plant. An experiment was designed where the quadrotor stability at nominal hover condition (0° orientation) was perturbed with an impulse of wind disturbance that was sustained for 1 second. Since

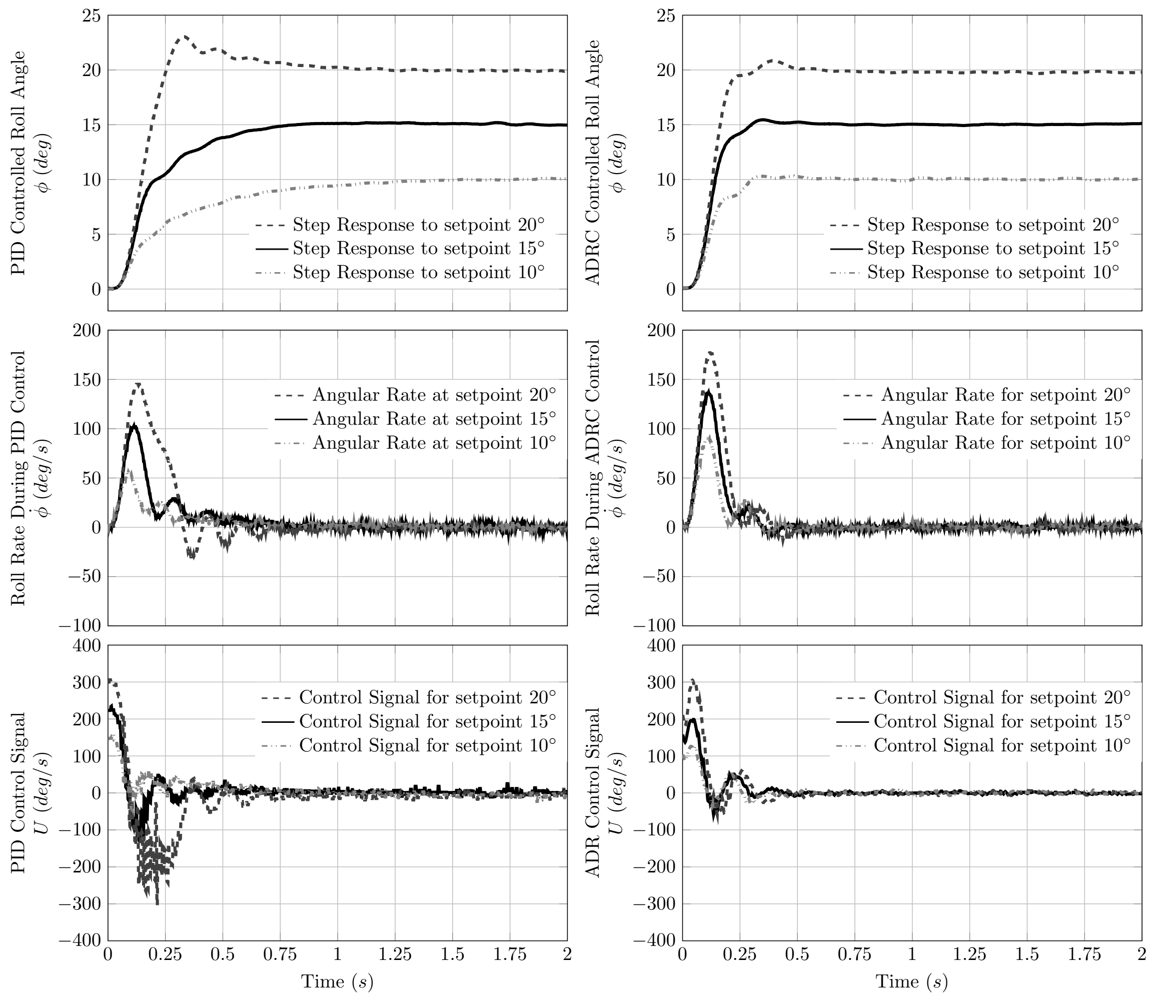


Figure 6.15: Experimental results of the Cascaded PID Controller during stabilization.

the magnitude of the wind force was unknown, an identical setup was used for both control technologies and the disturbance was triggered in a time synchronous fashion. As seen in the experimental setup in Figure 6.12, a 9 mm air nozzle was placed 8 inches below the propeller as the wind-gust generator, which was connected to an air supply unit pressurized with 500 kPa. First the plant was controlled for the nominal hover condition; once it was stable the experiment was initiated along with the data recording. After 1 second of the experiment the pneumatic solenoid was activated that triggered the disturbance. After 2 seconds the solenoid was deactivated to stop

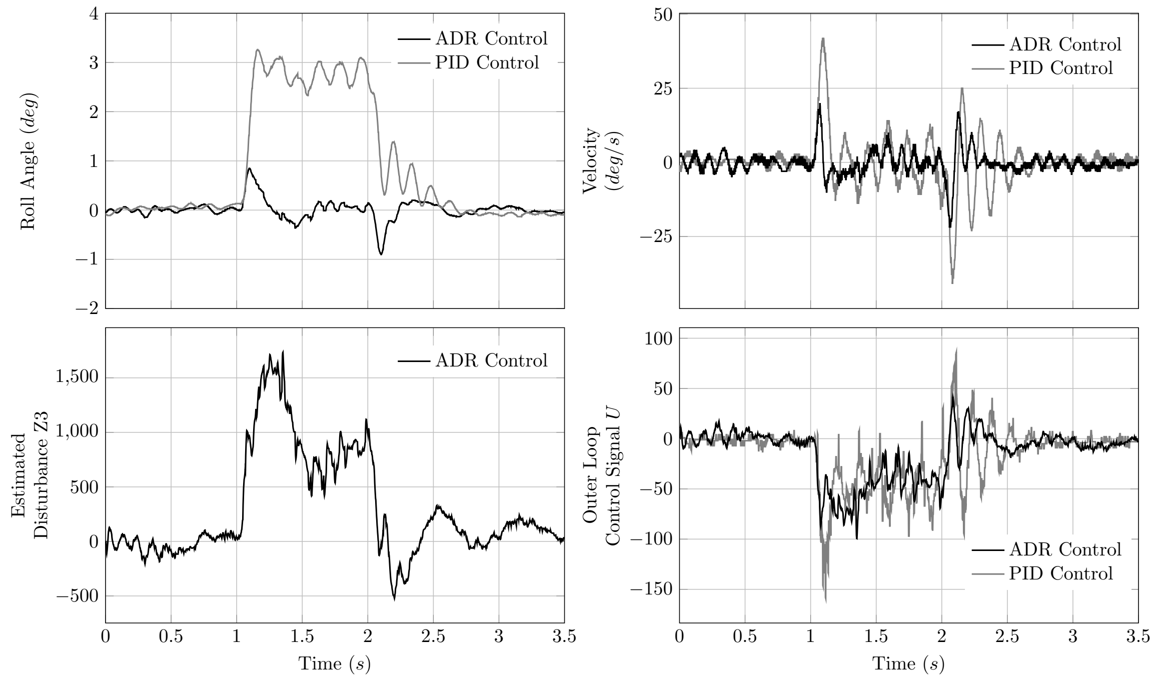


Figure 6.16: Experimental results of the Cascaded PID Controller during perturbed condition.

generating wind disturbance and the data was recorded for another full second. The system responses are presented in Figure 6.16.

It is obvious from the control response that the ADRC reacted and responded to the given disturbance better than the PID controller for an identical disturbance. The ESO in the ADRC estimated the disturbance as soon as it was triggered and accordingly the controller compensated for it in the control signal to maintain the hover condition, whereas the PID controlled response was affected by the disturbance, as seen in the experimental result.

6.6 Discussion & Conclusion

In this chapter a linear Active Disturbance Rejection Control was implemented for quadrotor attitude stabilization. A cascaded control architecture was proposed wherein the inner loop PID controller controls the high speed dynamics of the platform (angular rate) and the outer loop ADRC controls the stochastic unknown external disturbances with better dynamic range. The controller along with the supporting modules (e.g., data acquisition system and the actuation module) were implemented in FPGA hardware. The contribution of this work lies in both control law development and the hardware implementation of the control stabilization system in the FPGA hardware.

An experiment was conducted to validate the control performance, wherein the performance of ADRC was compared with a widely used PID control technology. It was noticed that once ADRC is set up and tuned for an arbitrary step input response, no further tuning is needed for the controller to perform well within a defined operating range of the quadrotor. Admittedly the data presented in Table 6.2 are not an absolute measure of the control performance as these performance matrices depend on the tuning parameter. However, it was confirmed through simulation that ADRC performs well for a non-linear, time-varying system. In addition, the experimental result for the perturbed plant presented in Figure 6.16 indicates the robustness in control performance when the plant is subjected to wind disturbance.

Disturbance rejection of the ADRC and PID control technology was only verified by operating in the time domain. More information could be obtained by using frequency domain characteristics. It could also supply better ADRC and PID tuning parameters and thus increase control bandwidth. Future work involves implementation of a nonlinear ESO to better mitigate the overshoot observed in the controlled response, and subsequently evaluate in-flight control performance when all three rotational DOFs are controlled using the proposed ADRC based cascaded control architecture for stabilization.

Chapter 7

Single Chip Monocular Visual Servoing Solution for Quadrotor

7.1 Introduction

Control of small-scale UAVs is exciting research with a wide range of practical applications. Quadrotor UAVs are of particular interest in control applications due to their small size, great maneuverability and hovering capability. Omni-directional flight capability also extends the range of possible applications. In addition, a small-scale quadrotor can offer the agility necessary to operate in a dynamically changing environment by responding quickly to abrupt changes (i.e., wind gusts).

In most cases, these kinds of autonomous systems have utilized a wide range of mechanical and non-visual sensors, including rate gyros, accelerometers, magnetometers, and a GPS to determine the attitude and the position of the vehicle. In addition, if the UAV requires tracking a moving object during flight, more sensors such as visual and sound sensors are needed to ensure successful control of the platform. However, to integrate a large number of sensors and measurement units demands for increased on-board real-estate as reported by the literature discussed in Section 2.5. In contrast, the goal of this chapter is to develop a visual servoing solution for a small-scale quadrotor using minimal hardware components, so that the agility of the platform can be harnessed while featuring vision based position control. Moreover, these platforms use GPS for localization, which is not suitable for indoor navigation or precise position control (e.g., hovering over charging station, landing on remote vessels etc.) as GPS receivers do not work in indoor operations and the position measurements are inaccurate by up to a few meters when operating outdoors. Hence, development of a visual servoing solution for a small-scale aerial platform involves a number of engineering challenges including selection of appropriate sensor and processing hardware and the development of an efficient visual servoing algorithm that can be deployed in the selected hardware. Thus, knowledge from multiple engineering disciplines is required, that can range from mechanical design or sensor integration to software and firmware development.

Among the existing visual servoing research as discussed in Section 2.5, the number of cameras used to develop the vision system varied from a minimum of one [84, 85] to a maximum of four [80]. Many researchers have implemented cameras both onboard and at the ground control station [85] while others have developed monocular pose estimation [80, 82, 92]. Subsequently, markers have been used to acquire visual reference feedback [83, 84, 88]. Adopting the ideas from the existing solution architectures and at the same time realizing the constraints of a small-scale platform, this study proposes a vision processing hardware solution using only one on board IR pass filtered camera that is looking downward towards a reference landmark back-lit with IR-LEDs (light emitting diode). Unfortunately, the choice of such limited sensory components increases the challenge for development of a vision processing algorithm. In order to complement the vision processing hardware with more sensory/state information, Kalman filter has been fused with vision processing algorithm.

The objective of the vision processing algorithm is to estimate the relative pose of the vehicle with respect to the landmark. Previous literature, has reported efficient pose estimation algorithms that can estimate three degrees of freedom to six degrees of freedom based on the number of sensors integrated with the system, including an optical flow sensor [81]. However, the complexity of the reported algorithms involves solving non-linear equations [82] through computationally heavy iterative solutions [79, 86], which demands superior computing power that is unavailable in a compact aerial platform. To strike a balance between the requirement of on-board computing power and the complexity of a pose estimation algorithm a vision processing technique is proposed in this study that estimates relative translation (3-DoF) of a platform from a single perspective view using a circular passive marker and single FPGA chip to prototype the servoing algorithm. As discussed earlier in Section 2.5, FPGA technology has made significant advances since its inception, and today can offer a single chip reconfigurable solution for integration of SWaP efficient parallel computing power to small-scale UAVs (i.e., quadrotors). Accordingly, a monocular visual servoing solution is developed for single chip FPGA implementation.

The chapter is organized as follows. Section 7.2 develops the mathematical framework of the proposed visual servoing technique, followed by the formulation of the control law presented in Section 7.3. The algorithm is then realized in FPGA hardware in Section 7.4. The performance achieved by the proposed design is experimentally evaluated in Section 7.5. Finally, concluding remarks are offered in Section 7.6.

7.2 Mathematical Framework

The proposed visual servoing technique is developed and executed in four consecutive steps. First, an image processing algorithm is developed that analyses the projection of the circular landmark in the image plane to extract the feature information (i.e., centroid and area) of the projected blob. Next, the vision processing algorithm is developed to estimate the relative position of the landmark based on the location of the centroid of the blob in the image plane and the intrinsic properties of the camera (i.e., focal length, camera offset etc.); wherein, the area of the centroid is used to evaluate the altitude of the vehicle from the target. Once the position information is obtained, motion of the platform is estimated through fusion of Kalman filter with the data from vision processor. Finally, the closed loop control law is developed to establish a feedback control system to control the motion of the platform over the target. To understand the technique and to conduct detailed analysis of the each process, a mathematical framework is developed in the following sections. For simplicity of the algorithm, the study assumes that the landmark/target is horizontal to the earth reference frame.

7.2.1 Image Processing Algorithm

The primary criteria for selection of a suitable image processing algorithm include the sensitivity of the algorithm to changing environmental conditions (i.e., lighting) and the ability to handle low quality images formed by the moving camera (mounted on the vehicle). Computational complexity of the algorithm is also critical to obtain information at a high update rate [99]. A fast and accurate feature extraction algorithm is adopted [150] for image processing that works in two steps; first, the acquired RGB (red-green-blue) image is converted into a binary image, then the required moment invariants (i.e., centroid and area) of the projected blob are calculated through contour tracking of the binarized image.

7.2.1.1 Binarization

Since the proposed method uses IR-LEDs with a wavelength matched with the wavelength of the infrared-pass filter inserted in the camera, the sensor in the camera potentially sees only an image of a bright circle when compared to the environment. Thus, a thresholding function is sufficient to detect the blob in the image and to binarize it at the same time:

$$\boldsymbol{I'}(u,v) = \begin{cases} \boldsymbol{I}(u,v), & \text{if } \boldsymbol{I}(u,v) > \text{threshold}, \\ 0 & \text{otherwise.} \end{cases} \tag{7.1}$$

Thresholding essentially reduces the number of channels in the image from three (i.e., RGB image) to one with each pixel having a value of either 1 or 0. However, the threshold parameter depends on the shutter speed of the camera settings. It is empirically determined that a large range of parameters works well (80 - 180) for an 8-bit data. For the simplicity of hardware implementation a threshold parameter is identified from the range of operation that worked best for the vision system.

7.2.1.2 Feature Extraction

In the next step, spatial moments of intensity distributions for the binary image are calculated to find coordinates of the centroid of the projected blob and the area of the binary pixel distribution [151]. This method of identifying the features of the target is less sensitive to the relative motion between the camera (mounted on the vehicle) and the landmark. The n^{th} moment about a point p in the image plane (u, v) is:

$$\mu_n = \int_{-\infty}^{\infty} (u_\mu - p)^n f(u_\mu) du_\mu. \tag{7.2}$$

Since an image has two dimensions, the relationship can be extended to:

$$\mu_{m,n} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (u_\mu - p_x)^m (v_\mu - p_y)^n f(u_\mu, v_\mu) du_\mu dv_\mu. \tag{7.3}$$

However, calculating the area of the binarized image is the same as calculating the zeroth moment of the image:

$$\mu_{0,0} = \sum_{u=0}^{w} \sum_{v=0}^{h} u_\mu^0 v_\mu^0 f(u_\mu, v_\mu). \tag{7.4}$$

For binary images the function $f(x,y)$ takes a value of 1 for pixels belonging to class *object* and '0' for class *background*. Since every binary pixel is either a 1 or a 0 due to binarization, equation (7.4) can be used to calculate the amount of area covered by white pixels, thereby calculating the area of the circular blob in the image. Therefore, to find the centroid of the blob we need to calculate the $\mu_{1,0}$ moment and $\mu_{0,1}$ moment:

$$\mu_{1,0} = \sum_{u=0}^{w}\sum_{v=0}^{h} u^1 v^0, \qquad \text{and} \qquad \mu_{0,1} = \sum_{u=0}^{w}\sum_{v=0}^{h} u^0 v^1. \tag{7.5}$$

The moment sums the u-coordinates for which the pixel value is 1 and also for the v- coordinates. The centroid is then calculated by dividing these sums by the area, which is essentially the average location in terms of u and v. The centroid is also called the first-order central moment:

$$(u_c, v_c) = \left(\frac{\mu_{1,0}}{\mu_{0,0}}, \frac{\mu_{0,1}}{\mu_{0,0}}\right). \tag{7.6}$$

Higher order moments can also be computed to provide additional data for pattern recognition of scale, translation, and rotation invariant. However, the higher the order of the moment, the more computation cycles are required to extract the moment. Higher order moments are not used in this study because algorithmic and computational simplicity are prioritized to ensure meeting current real-time deadlines and to maximize available resources for auto-flight controller extensions.

7.2.2 Vision Processing

To formulate the vision processing algorithm, a pinhole camera model is used and the necessary coordinate definition is illustrated in Figure 7.1. Here, $[{}^{w}X,\ {}^{w}Y,\ {}^{w}Z]^T$ are axes of the three-dimensional world coordinate frame $\mathbb{W}$, $[{}^{c}X,\ {}^{c}Y,\ {}^{c}Z]^T$ are the axes of the camera frame $\mathbb{C}$, which is attached to the camera lens center $\boldsymbol{O}$. $[U,\ V]^T$ are the axes of the image sensor plane, $\boldsymbol{O_S}$ denotes the point where the z axis of the camera

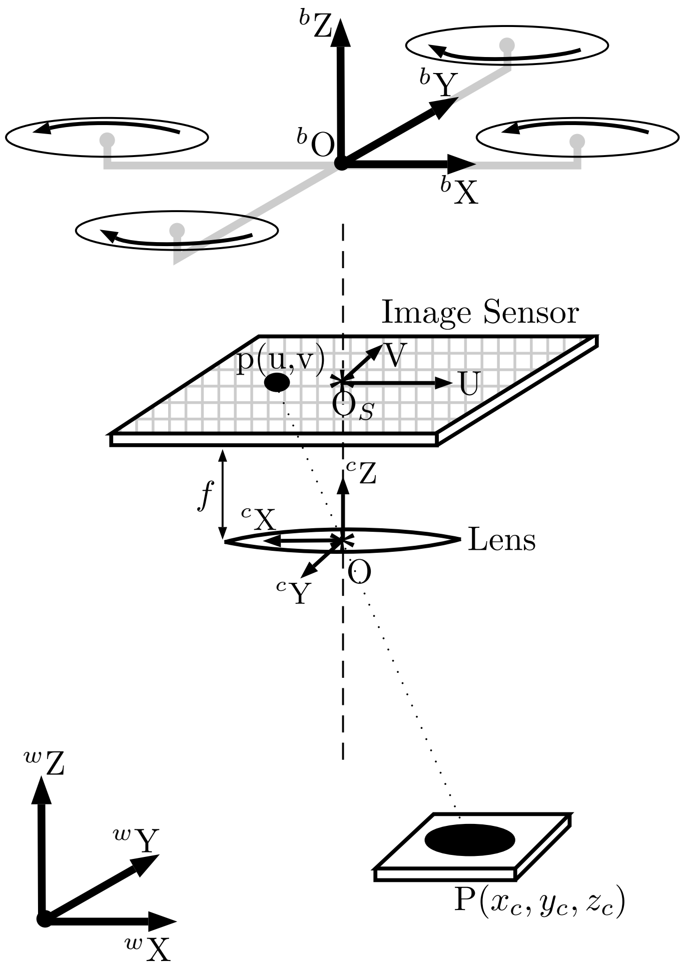
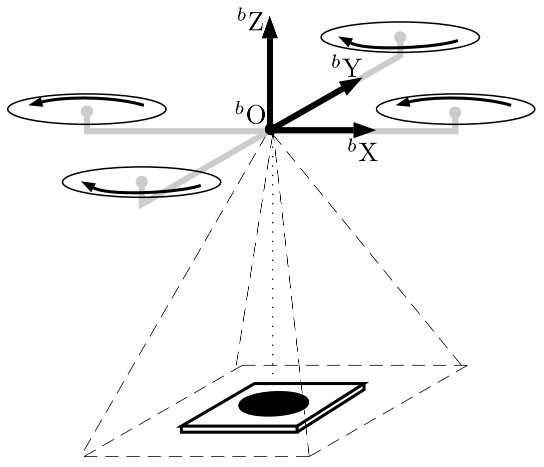


Figure 7.1: Geometry and coordinate frame of vision system.

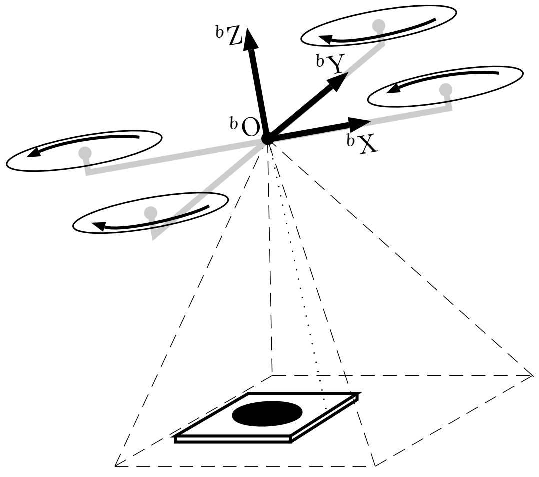
coordinate frame intersects the image plane, and f is the focal length of the camera. The mapping from a point ${}^c\boldsymbol{P} = [x_c,\, y_c,\, z_c]^T$ to a point p in the image plane can be written as:

$$\boldsymbol{p} = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{z_c} \begin{bmatrix} x_c \\ y_c \end{bmatrix}, \tag{7.7}$$

here, (u, v) denotes the location of the centroid of the projection of the circle in the landmark (target).



(a) Quadrotor with zero roll and pitch orientation. Dashed square indicate the image seen by the on-board camera.



(b) Quadrotor with non-zero orientation angles. Note that, the target is shifted from its original location whereas the UAV is holding its position.

Figure 7.2: Effect of body rotation on the image data.

7.2.2.1 Rotation Effect Compensation

While the quadrotor is flying, image displacements can occur not only with translation of the vehicle but also with its rotation as seen in Figure 7.2. In order to accurately sense the translation of the vehicle, rotation effects must be eliminated from the measured image displacement. The effect is more prominent in the case of quadrotor, as the translational motion of the vehicle is directly resulted by the change in attitude of the vehicle. To address this problem and to compensate for the rotational components in the image displacement, onboard AHRS data (orientation angles $[\phi,\ \theta]$) are used. A virtual coordinate frame $\mathbb{V}$ is defined with the position and yaw angles aligned to match the camera coordinate frame, and the z axis is aligned with the z axis of the world frame. If references are generated from an image in $\mathbb{V}$, the desired decoupling is achieved. To map the captured image into $\mathbb{V}$, first consider an imaginary camera frame which is the same as the true camera frame but rotated to zero roll and pitch.

The image features in this imaginary frame become:

$$\begin{aligned} {}^{r}\boldsymbol{P} &= \boldsymbol{R}(\theta, y)\ \boldsymbol{R}(\phi, x)^{c}\boldsymbol{P}, \\ \text{and, } {}^{r}\boldsymbol{p} &= \frac{f}{{}^{r}z_c}\begin{bmatrix} {}^{r}x_c \\ {}^{r}y_c \end{bmatrix} = f\begin{bmatrix} \frac{u\cos\theta + f\sin\theta}{-u\cos\phi\sin\theta + v\sin\phi + f\cos\phi\cos\theta} \\ \frac{u\sin\phi\sin\theta + v\cos\phi - f\sin\phi\cos\theta}{-u\cos\phi\sin\theta + v\sin\phi + f\cos\phi\cos\theta} \end{bmatrix} = \begin{bmatrix} {}^{r}u \\ {}^{r}v \end{bmatrix}, \end{aligned} \tag{7.8}$$

where

$$\boldsymbol{R}(\phi, x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix},\ \boldsymbol{R}(\theta, y) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}.$$

Here, ${}^{r}\boldsymbol{P}$ is the coordinate defined in the roll and pitch compensated camera frame and ${}^{r}\boldsymbol{p}$ is the corresponding image feature coordinate of ${}^{r}\boldsymbol{P}$. For nominal flight regimes, roll and pitch angles do not approach 90° so the denominators in equation (7.8) will be non-zero. Interestingly, to this end, no depth information is needed to perform the roll and pitch compensation. However, to measure the 3-D position of the quadrotor with respect to the camera the depth information is required.

7.2.2.2 Depth Estimation

Assuming that we know the nominal geometry of the target in the camera frame, we can now estimate the relative depth. The height in the camera frame z_c can be calculated if the area of the target object S, is known beforehand using the equation (7.9). s is the area of the blob in the image frame in $pixel \times pixel$, which can be converted to mm^2 using pixel dimension, as reported by the sensor manufacturer. Therefore, the depth from the camera to the image features can be calculated as [152]:

$$\frac{f}{z_c} = \sqrt{\frac{s}{S}}. \tag{7.9}$$

At this point, in theory, it is possible to reconstruct the translational motions of the target and perform image based control of the motions. However, this simple depth calculation is only a rough estimate and can be adopted in the case where computational efficiency is prioritized over measurement accuracy [153], and this would not produce an accurate enough reconstruction for altitude control algorithms. Fortunately, image based control (i.e., visual servoing) is less sensitive to this measurement so having only this rough estimate is sufficient and subsequently adopted in this chapter for further development.

7.2.2.3 Camera Offset Compensation

The last thing that is included in the numerical framework development is the geometric distortion of the camera due to an offset from the origin of the quadrotor body frame to the origin of the camera frame, which is denoted as $[\delta_x,\, \delta_y,\, \delta_z]^T$. To generalize the relationship, consider another imaginary camera frame $\mathbb{I}$, which is translated to compensate for the discrepancy caused by rotation of the offset. Assuming the feature location in this frame ${}^{\boldsymbol{I}}\boldsymbol{P}$, for which the following relationship can be developed:

$$ {}^{\boldsymbol{I}}\boldsymbol{P} = {}^{\boldsymbol{r}}\boldsymbol{P} + \boldsymbol{R}(\theta, y)\ \boldsymbol{R}(\phi, x)\ [\delta_x,\, \delta_y,\, \delta_z]^T. \tag{7.10} $$

Now, the only difference between the virtual coordinate frame $\mathbb{V}$, and the imaginary camera frame $\mathbb{I}$, is the offset position of the camera. Therefore, the target coordinate of center defined in the virtual frame is:

$$ \begin{aligned} {}^{\boldsymbol{v}}\boldsymbol{P} &= {}^{\boldsymbol{I}}\boldsymbol{P} - [\delta_x,\, \delta_y,\, \delta_z]^T, \\ \text{and, } {}^{\boldsymbol{v}}\boldsymbol{p} &= \frac{f}{{}^{v}z_c} \begin{bmatrix} {}^{v}x_c \\ {}^{v}y_c \end{bmatrix}. \end{aligned} \tag{7.11} $$

where ${}^{\boldsymbol{v}}\boldsymbol{p}$ is the image feature coordinate of ${}^{v}\boldsymbol{P}$. Considering the rotations of the body frame, taken from the AHRS module, the coordinates in the camera frame x_c, y_c, and

z_c are transformed to the world frame as:

$$
\begin{aligned}
{}^{w}\boldsymbol{P} &= \boldsymbol{R}(\psi, z)\ \boldsymbol{R}(\theta, y)\ \boldsymbol{R}(\phi, x)\ {}^{v}\boldsymbol{P} = {}^{w}\boldsymbol{R_b}\ {}^{v}\boldsymbol{P}, \\
\text{where, }\ {}^{w}\boldsymbol{R}_b &= \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}.
\end{aligned}
\tag{7.12}
$$

Here, ZYX Euler angle representation has been chosen for roll(ϕ), pitch(θ), and yaw(ψ).

7.2.3 Translational Velocity Estimation

The vision processing algorithm thus far provides an estimate of the relative position of the vehicle. However, robustness of the control law partially depends on feedback of velocity information. Accordingly, a motion estimation algorithm is developed in this section.

In general, translational velocities can be calculated using the first derivative of the position information obtained from vision processing in Section 7.2.2. In reality, the measurements of the relative position between the quadrotor and the moving target obtained are noisy (nature of image data), which can amplify if only the time derivative of the position data is considered for velocity estimation. In order to reliably (i.e., with low noise) compute the velocities as well as improve the position estimation, Kalman filter is used to estimate velocity based on vision-based position measurements only.

7.2.3.1 Velocity Estimation using Kalman Filter

For the development of the process model of the Kalman filter, it is assumed that the target is moving with piece-wise constant velocity and acceleration of the target is negligible. Accordingly, the state space model is formulated as:

$$\begin{aligned} \boldsymbol{X}_{k+1} &= \boldsymbol{A}\boldsymbol{X}_k + \boldsymbol{B}(\boldsymbol{U}_k + \boldsymbol{W}_k), \\ \boldsymbol{Y}_k &= \boldsymbol{C}_k\boldsymbol{X}_k + \boldsymbol{V}_k. \end{aligned} \tag{7.13}$$

To simplify the formulation of the model, the Kalman Filter is derived assuming motion along the x axis only. However, numerical models for motion along the remaining axes are identical to this formulation. Assuming there is no input $\boldsymbol{U}_k$ to the system:

$$\boldsymbol{A} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \boldsymbol{C} = [1 \quad 0],\ \boldsymbol{X_k} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{7.14}$$

Here, similar to process noise $\boldsymbol{W}_k$, in the measurement noise, $\boldsymbol{V}_k$ is assumed to have white Gaussian noise. The two state Kalman equation can then be written as:

$$\begin{aligned} &\hat{\boldsymbol{X}}\boldsymbol{1}_{k|k-1} = \boldsymbol{A}\hat{\boldsymbol{X}}_{k-1|k-1} + \boldsymbol{B}\boldsymbol{W}_{k-1}, \\ &\boldsymbol{P1}_{k|k-1} = \boldsymbol{A}\boldsymbol{P}_{k-1|k-1}\boldsymbol{A^T} + \boldsymbol{B}\boldsymbol{Q_k}\boldsymbol{B^T} \\ &\hat{Y}_k = Y_k - \boldsymbol{C}\hat{\boldsymbol{X}}\boldsymbol{1}_{k|k-1} \\ &\boldsymbol{G}_k = \boldsymbol{P1}_{k|k-1}\boldsymbol{C^T}[\boldsymbol{C}\boldsymbol{P1}_{k|k-1}\boldsymbol{C^T} + R_k]^{-1} \\ &\hat{\boldsymbol{X}}_{k|k} = \hat{\boldsymbol{X}}\boldsymbol{1}_{k|k-1} + \boldsymbol{G}_k\hat{Y}_k \\ &\boldsymbol{P}_{k|k} = \boldsymbol{P1}_{k|k-1} - \boldsymbol{G}_k\boldsymbol{C}\boldsymbol{P1}_{k|k-1} \end{aligned} \tag{7.15}$$

where

$$\boldsymbol{G}_k = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix},\ \boldsymbol{P}_{k|k} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix},\ \boldsymbol{P1}_{k|k-1} = \begin{bmatrix} P1_{11} & P1_{12} \\ P1_{21} & P1_{22} \end{bmatrix},\ \boldsymbol{Q_k} = \begin{bmatrix} Q_p & 0 \\ 0 & Q_v \end{bmatrix},$$

$$\hat{\boldsymbol{X}}\boldsymbol{1_{k|k}} = [X1_1 \quad X1_2]^T,\ \hat{\boldsymbol{X}}_{\boldsymbol{k|k}} = [X_1 \quad X_2]^T.$$

$\boldsymbol{P1}_{k|k-1}$ is the priori error covariance estimate, $\hat{\boldsymbol{X}}\boldsymbol{1_k}$ is the priori state estimate, $\hat{\boldsymbol{Y}}_{\boldsymbol{k}}$ is the output estimate, $\boldsymbol{G}_k$ is the Kalman gain, $\hat{\boldsymbol{X}}_{\boldsymbol{k|k}}$ is the posteriori state estimate, and $\boldsymbol{P}_{k|k}$ is the posteriori error covariance estimate. $\boldsymbol{Q_k}$ is the system noise covariance matrix and $\boldsymbol{R_k}$ is the position measurement noise covariance.

Following the computation in equation (7.15), we perform the Kalman filtering for every loop and obtain an optimal estimation of current position and velocity in $\hat{\boldsymbol{X}}_{k|k}$. Since the initial condition is not important to initialize the Kalman filtering process, the initial state variables are assigned to null when the filter first starts. Note that the process model in the velocity estimation is dependent on wind, as the airspeed directly affects the drag term. However, a wind estimation scheme is beyond the scope of this work, so it is deferred to future study of this thesis.

7.3 Control Law Design

The control philosophy adopted for visual servoing of the quadrotor consists of three stages: target acquisition, target tracking, and altitude control for autonomous climb or descent. The three stages are illustrated in Figure 7.3. First, the quadrotor flies towards the target as defined by the GPS location of the target. Then when the target/landmark is within the field of view of the vehicle, it regulates its GPS position to that of the target. After the target is detected, the feedback input for the position control algorithm changes from inertial frame to the body-planar frame and the target tracking stage begins. Relative position information is used at this stage of control. Both magnetometer and GPS location information are no longer used at this time due to their high degree of error. As soon as the position error falls below a certain condition, the controller of the quadrotor shifts from tracking to altitude control mode, where the vehicle climbs/descends on the target. However, the the vehicle still tracks the target during this phase, provided that the target is within field of view of the camera. Different flight controllers may be used at different stages of the flight; however, this study is limited to stage two and three (i.e., target tracking & altitude control) of the visual servoing process.

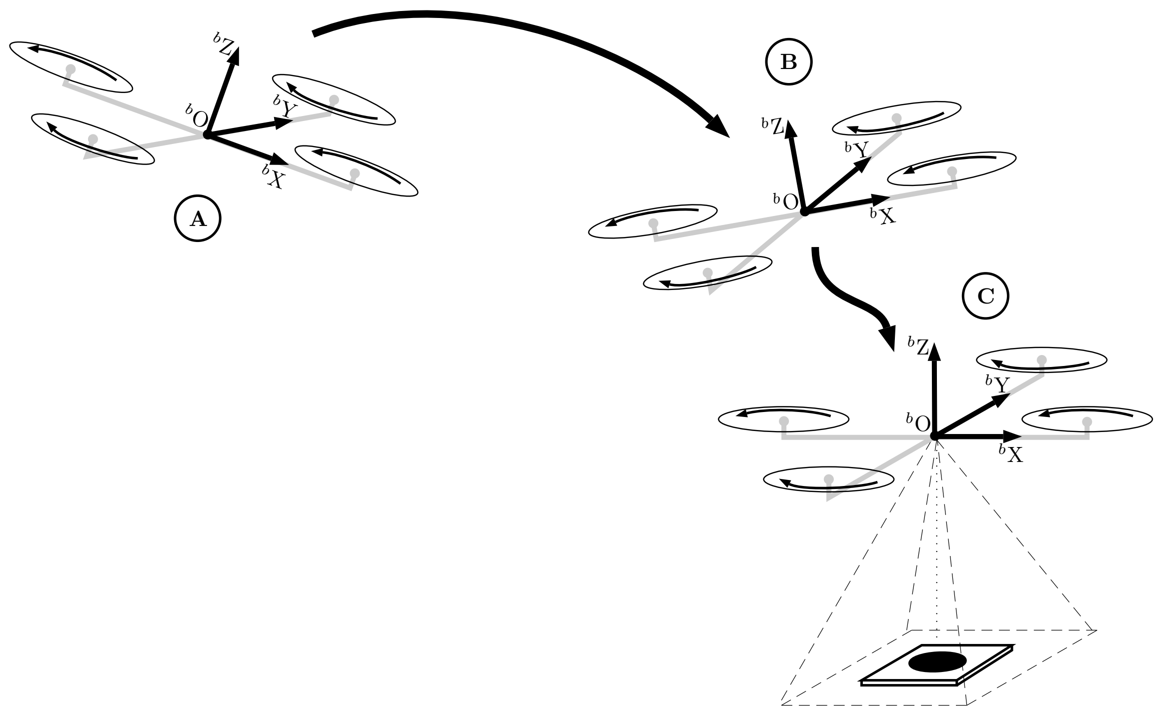


Figure 7.3: The three stages of flight control plan. A. Target acquisition B. target tracking C. altitude control while tracking.

Attitude stabilization and control are always performed at a lower level, which is developed in Chapter 6. The low-level attitude control should rely only on inertial measurement data from the AHRS. On the other hand, position controller manipulates the roll and pitch attitude of the vehicle to initiate translational motion. Yaw control is not used in visual servoing, hence the single circle based pose estimation approach is not designed to estimate yaw or any angular orientation. In general, the yaw is used to point an on-board camera of the vehicle to an area of interest. As the focus of this study is limited to downward looking camera and vision based climb/descent, the yaw is kept free for the end user. Moreover, by not restricting the motion on the yaw, a second camera can be integrated to the quadrotor, which can be used to maintain focus on an arbitrary subject during tracking and/or altitude change mode of the vehicle.

7.3.1 Relative Position Controls

The target tracking and altitude control are performed using relative positioning between the quadrotor and the target. Let the inertial position of the quadrotor denoted as ${}^{\boldsymbol{w}}\boldsymbol{P_q} = [x_q,\ y_q,\ z_q]^T$. In order to control relative motion of the target with respect to the body frame (as the actuation control signals are generated with respect to body frame), the position and velocity of the target need to be obtained in the body-planar frame $\mathbb{B}$. Since the $\mathbb{B}$ frame is centered at ${}^{\boldsymbol{w}}\boldsymbol{P_q}$ right before the tracking and altitude control takes place, the relative position is defined as:

$$ {}^{\boldsymbol{b}}\boldsymbol{P_r} = {}^{\boldsymbol{b}}\boldsymbol{R_w}\ ({}^{w}\boldsymbol{P} - {}^{\boldsymbol{w}}\boldsymbol{P_q}). \tag{7.16} $$

Since the focus of the work is to develop a control algorithm, irrespective of the dynamics/motion of the the target/landmark, no assumptions are made about the dynamics of the target. Instead, if the dynamics of the target was known, that could be used to complement the estimation of the relative position between the target and the vehicle. However, the knowledge on target dynamics is not always possible to obtain. Accordingly, only the relative position of the target and the vehicle is used to evaluate the visual servoing commands required to generate the motion of the quadrotor.

7.3.2 Cascaded Feedback Control

The desired position and velocity for the quadrotor during control of the translational motion are chosen to match the relative position estimate between the target and the quadrotor, such that error is defined as:

$$ \begin{aligned} \boldsymbol{e}_p &= \boldsymbol{P}^* - {}^{\boldsymbol{b}}\boldsymbol{P_r}, \\ \boldsymbol{e}_v &= \boldsymbol{V}^* - {}^{\boldsymbol{b}}\dot{\boldsymbol{P}}_{\boldsymbol{r}}, \end{aligned} \tag{7.17} $$

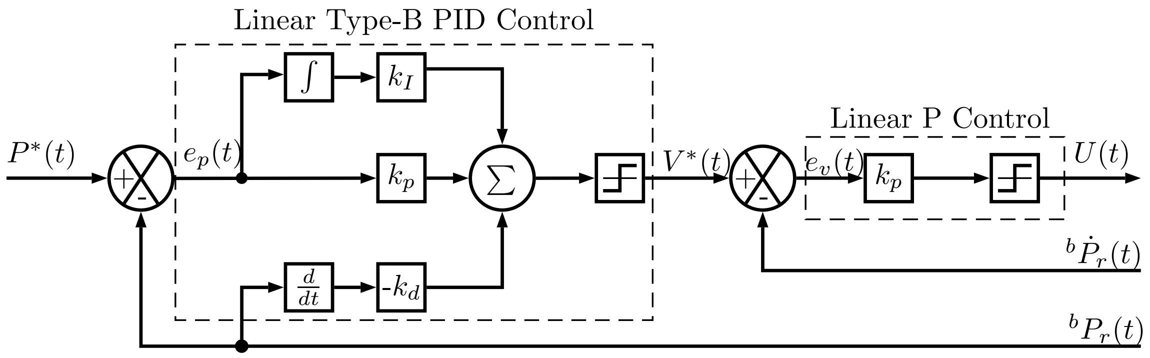


Figure 7.4: Proposed cascade PID-P control law for visual servoing application.

where $\boldsymbol{P}^*$ and $\boldsymbol{V}^*$ are the desired position and velocity respectively. For example, to maintain position hold right over the target and to track the target, the desired relative lateral position and velocity are zero, with a desired altitude. The position control algorithm used to develop the visual servoing is based on the same cascade control law proposed in Chapter 6. However, in this case the outer-loop is a PID control law that maintains the desired relative position while the inner-loop is a simple proportional control that tracks the desired relative velocity of the platform. The inner-loop velocity controller ensures the motion stability of the vehicle. It acts as a damper for the motion control law. The control law can be presented with the following equation:

$$\begin{aligned} \boldsymbol{V}^* &= k_{p,T}\,\boldsymbol{e}_P + k_{i,T}\int \boldsymbol{e}_P dt - k_{d,T}\,{}^{\boldsymbol{b}}\dot{\boldsymbol{P}}_{\boldsymbol{r}} \\ \boldsymbol{U} &= k_{p,v}\,\boldsymbol{e}_V \end{aligned} \tag{7.18}$$

Here, $k_{p,T}$, $k_{i,T}$, and $k_{d,T}$ are the proportional, integral, and derivative gains respectively for translational motion. $\boldsymbol{U}$ is the vector of control input for linear translation. As previously mentioned, the quadrotor is an under-actuated platform and the x and y state variables cannot be controlled directly from the inputs. Therefore, the horizontal motion control along the $-y$ and x axis is mapped to the desired roll and pitch in the body-planar frame respectively (see Section 4.2). An overview of the visual servoing control law is presented with a block diagram in Figure 7.4. Note that a type-B PID controller is employed in the outer loop of the proposed cascaded control architecture.

7.4 Hardware Implementation

Since real-time operation along with SWaP management are critical to this application, the guiding philosophy for hardware realization of the algorithms is to develop efficient digital circuits on single chip FPGA hardware. In addition to that, for the estimation of relative position to be useful in correcting the flight path of the quadrotor, it must be delayed as little as possible. Hence the system design employed here combines - on the same chip - all the application specific cores of the visual servoing process. Much of the effort associated with the implementation involves building, from the ground up: the visual servoing hardwares, the image processing core, the vision processing core, the velocity estimation core, and the flight control core as shown with a block diagram in Figure 7.5.

In order to conduct time efficient processing of the image information, the FPGA hardware is equipped with two SRAMs, which are connected directly to the FPGA bus line and are operated in parallel. While the image processing core is storing one frame (i.e., one image) of the landmark image into one of the SRAMs, the image features are extracted from the other image stored in the second SRAM and vice-versa. Next, the vision processing core computes the relative position of the vehicle to the target. This information is utilized by the velocity estimation core to generate feedback signals for the controller. Finally, the flight control core evaluates required orientation control signals (i.e., $\Delta\Omega_{lift}, \Delta\Omega_{\phi}, \Delta\Omega_{\theta}$) using the information from the velocity estimation core, and completes one cycle of the visual servoing process. For efficient data handling between each core, the visual servoing firmware is supported by a single FPGA bus as shown in Figure 7.5.

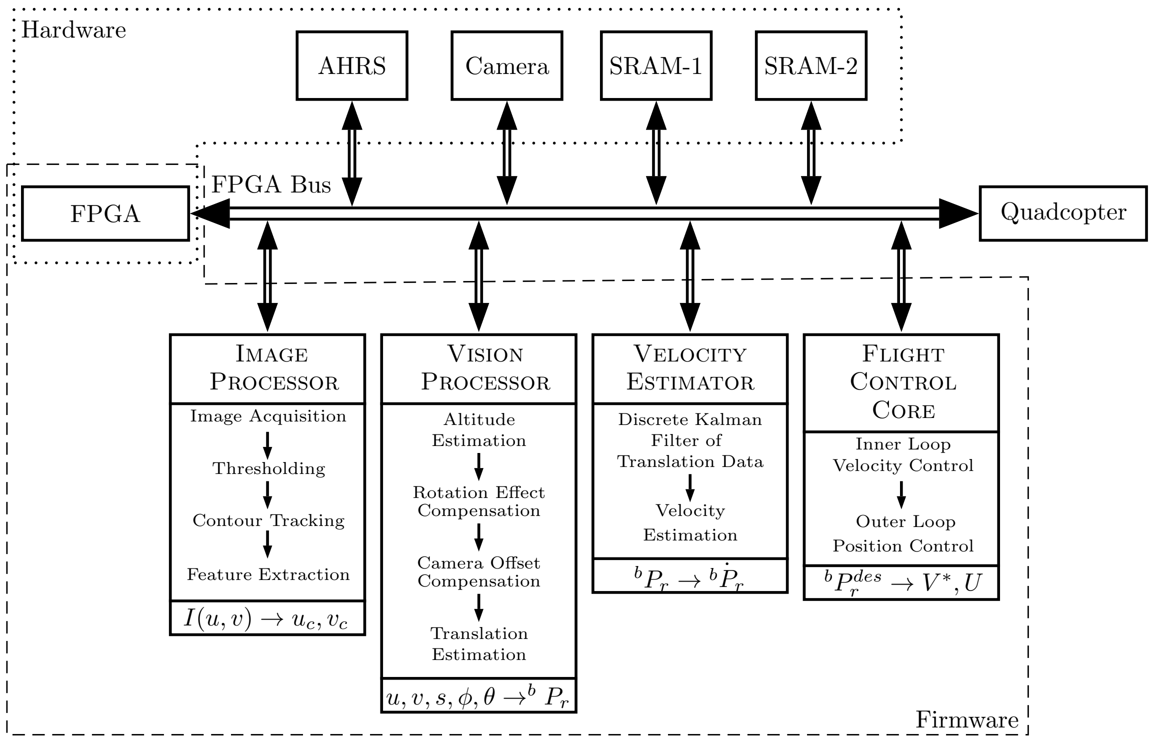

Hardware
AHRS
Camera
SRAM-1
SRAM-2
FPGA Bus
FPGA
Quadcopter
Image Processor
Image Acquisition
Thresholding
Contour Tracking
Feature Extraction
$I(u,v) \rightarrow u_c, v_c$
Vision Processor
Altitude Estimation
Rotation Effect Compensation
Camera Offset Compensation
Translation Estimation
$u, v, s, \phi, \theta \rightarrow^b P_r$
Velocity Estimator
Discrete Kalman Filter of Translation Data
Velocity Estimation
${}^bP_r \rightarrow {}^b\dot{P}_r$
Flight Control Core
Inner Loop Velocity Control
Outer Loop Position Control
${}^bP_r^{des} \rightarrow V^*, U$
Firmware

Figure 7.5: System architecture of Visual Servoing Core.

7.4.1 Implementation of the Image processing Core

The image processing core is implemented in FPGA hardware in conjunction with an image data acquisition circuit and image feature extraction circuit as discussed in the following sections.

7.4.1.1 Image Acquisition Circuit

The image acquisition circuit presented in Figure 7.6 serves as the FPGA interface for the digital image sensor, which includes a camera control circuit, SRAM control circuit, and image thresholding circuit, to capture, binarize, and store the image respectively. A 22-bit wide incremental counter is utilized to generate memory locations, which are then fed to both the camera controller and SRAM controller, to guarantee that the captured pixel information is stored in the RAM in proper sequence. Since raw pixel information is not required for this study, the image data are transferred

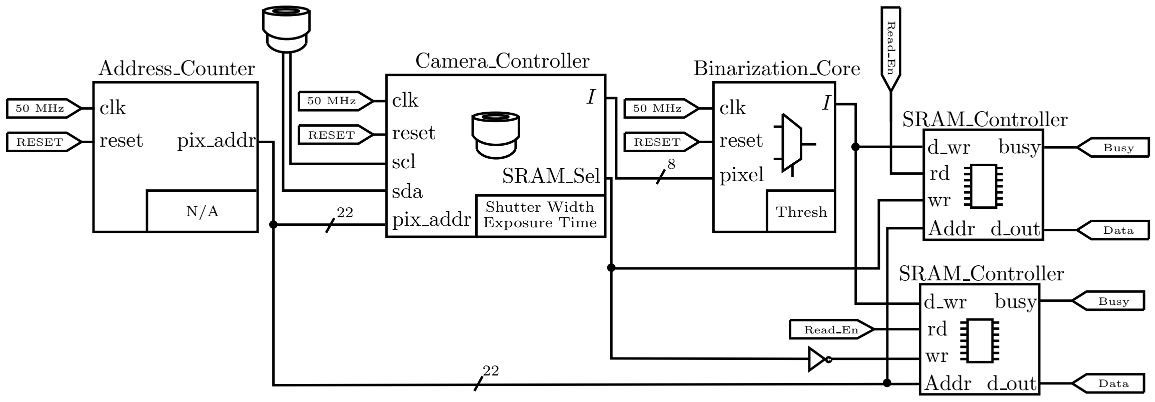


Figure 7.6: Block representation of image acquisition circuit.

from the camera to the binarization circuit, one pixel at a time, which is then compared with a given threshold value to rewrite the pixel information to either 1 or 0 as discussed in Section 7.2.1.1. At the end of the image acquisition process, SRAM only holds binary image information in its memory.

For the simplicity of the implementation, a fixed threshold value is identified and hard-coded into the conversion circuit. In this study, the threshold number is identified for a fixed IR intensity and fixed camera shutter width.

While the image acquisition circuit is storing a binary image into one SRAM, the image feature extraction circuit processes the image stored in the other SRAM module. The combination of this parallel and pipelined implementation allow processing the image with no latency introduced to other operations of the visual servoing core. The image size used in this study was 1024×768 pixels, and the frame rate of the camera was 37 Hz (dependent on exposure time).

7.4.1.2 Feature Extraction Circuit

In the binarized image captured and processed by the image acquisition circuit, the 1s are essentially the blob (i.e., projection of the circle) that represents the circle illuminated by the IR-LEDs based circular target. At this point, the only information

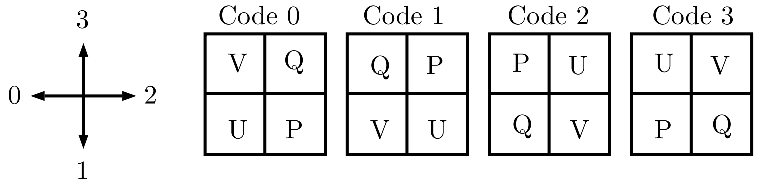


Figure 7.7: 4-Connectivity Crack codes.

Table 7.1: Table showing conditions of crack code during contour tracking.

U	**V**	**P'**	**Q'**	**Turn**	**Code**
$\times$	1	V	Q	Right	Code-1
1	0	U	V	None	Code
0	0	P	U	Left	Code+1

needed from each blob is the center of mass and the area of the blob. To accomplish this, a computationally efficient contour tracking/border following algorithm is implemented in the FPGA hardware as developed in [150]. Contour tracking segments the objects from the image and identifies the pixels that fall on the boundaries. A standard code definition based on a 4-connectivity code (crack code) is adopted in this study.

The approach is to scan the image until a pixel P that belongs to the class *objects* and a neighbouring pixel Q that belongs to the class *background* is encountered (0 in the binary image defines background and 1 in the image means it is a blob). Depending on the relative position of Q relative to P, pixels U and V are identified as shown in Figure 7.7. Pixels U and V are used to determine the next "move" (i.e., the next element of crack code) as summarized in the Truth Table 7.1. From this point, the direction to the next edge pixel is added to the crack code and tracking continues until the whole contour of the blob is defined. An example of the first three states of the crack code is illustrated in Figure 7.8.

In order to find the the shape features/generalized moments (i.e., area and centroid), the double (i.e., surface) integral is evaluated over the blob. However, imple-

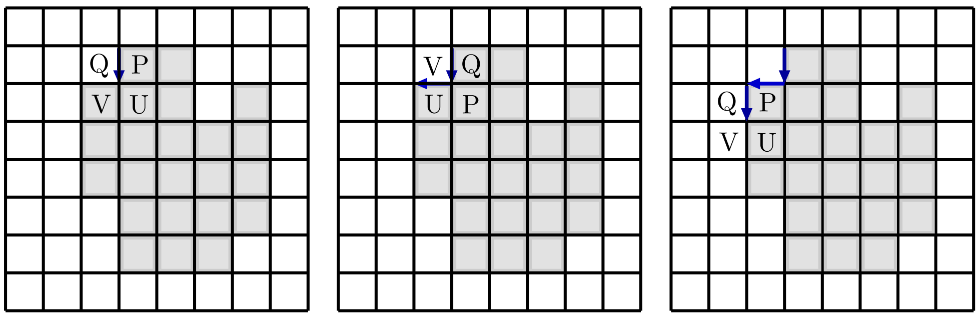
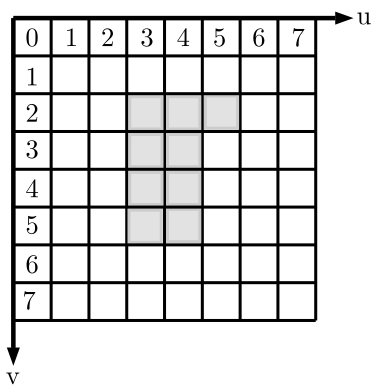


Figure 7.8: The crack code of the following blob is {10111211222322333300103300}



i	j	M_{ij}
0	0	9
0	1	33
1	0	30

Figure 7.9: Example showing the process of moment invariants calculation during contour tracking.

mentation of the surface integral is neither time efficient nor resource efficient when FPGA hardware implementation is considered. Therefore, applying Green's Theorem [154] the surface integral formulation is converted into a line integral for discrete hardware implementation, which can then be evaluated during contour tracking using the moment estimation algorithm developed in [155]. An example in Figure 7.9 illustrates the calculation of the moment invariants using equation (7.4) and (7.5), which can be evaluated in real-time as the contour tracking of the image is performed in hardware.

The contour tracking algorithm passes through the boundary/edge pixels only; hence, it requires random access to memory locations (i.e., pixel value), which is made possible by a memory controller circuit. Such an implementation is instrumental as it

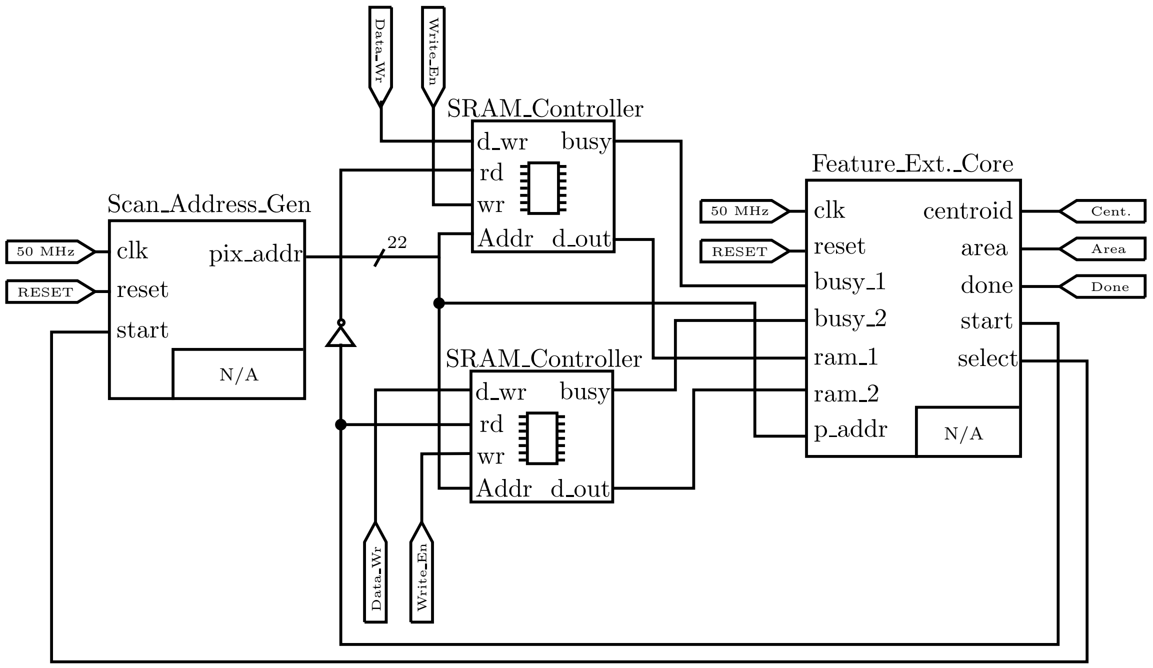


Figure 7.10: Hardware circuit of the Image Processing Core.

needs only a single pass through the image to calculate both centroid and area of the circle in the image plane. Also the single pass image processing guarantees that the feature extraction will be completed before the completion of the new image capture because the blob will always be a subset of the full image. A complete FPGA hardware architecture of the image processing core is presented in Figure 7.10. After the feature parameters are extracted, the information is then fed to the vision processing circuit for further processing.

7.4.2 Implementation of the Vision Processing Circuit

Using feature information of the blob, given the focal length of the camera lens and the trigonometric formulations established in equations (7.7) - (7.11), the relative position of the target is estimated. To simplify the trigonometric formulations to algebraic equations, small angle approximation [156] of the trigonometric functions is assumed ($\sin\theta \to \theta$ and $\cos\theta \to 1$). Subsequently, the equations are hard-coded in

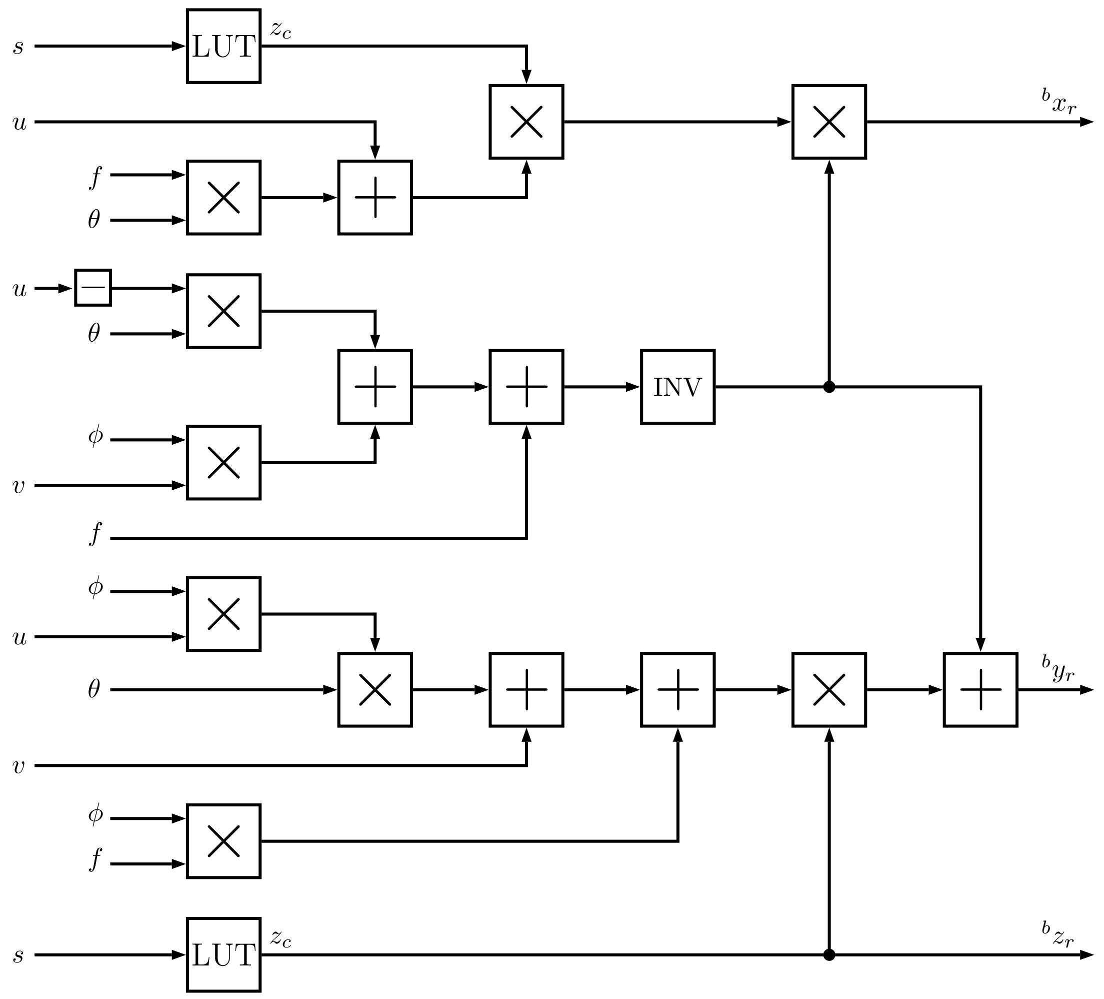


Figure 7.11: Hardware implementation of the pipelined vision processing circuit.

FPGA using simple adders and multipliers as shown in Figure 7.11. The pipelined implementation takes only six FPGA clock cycles to calculate the estimation relative position data. Because the frame rate of the camera (37 Hz) is much slower when compared to the FPGA clock (50 MHz), the translation estimation core computes data at the frame rate of the camera.

Since altitude estimation involves a square root function of the ratio of area (see equation (7.9)), it is resource inefficient to implement it in FPGA hardware. Therefore, the relation between altitude and the area of the blob was identified through altitude calibration, which was then hard-coded into the FPGA hardware using a look up table (LUT). Six adders and eight multipliers were required to obtain the proposed vision processing algorithm. For precision and accuracy 48 bit fixed point (Q32.16) signed integers were used, wherein a 16 bit LSB was assigned to carry fraction information of the data. To obtain the vision algorithm using a fixed data format, a fixed point library [132] was extensively used in this development.

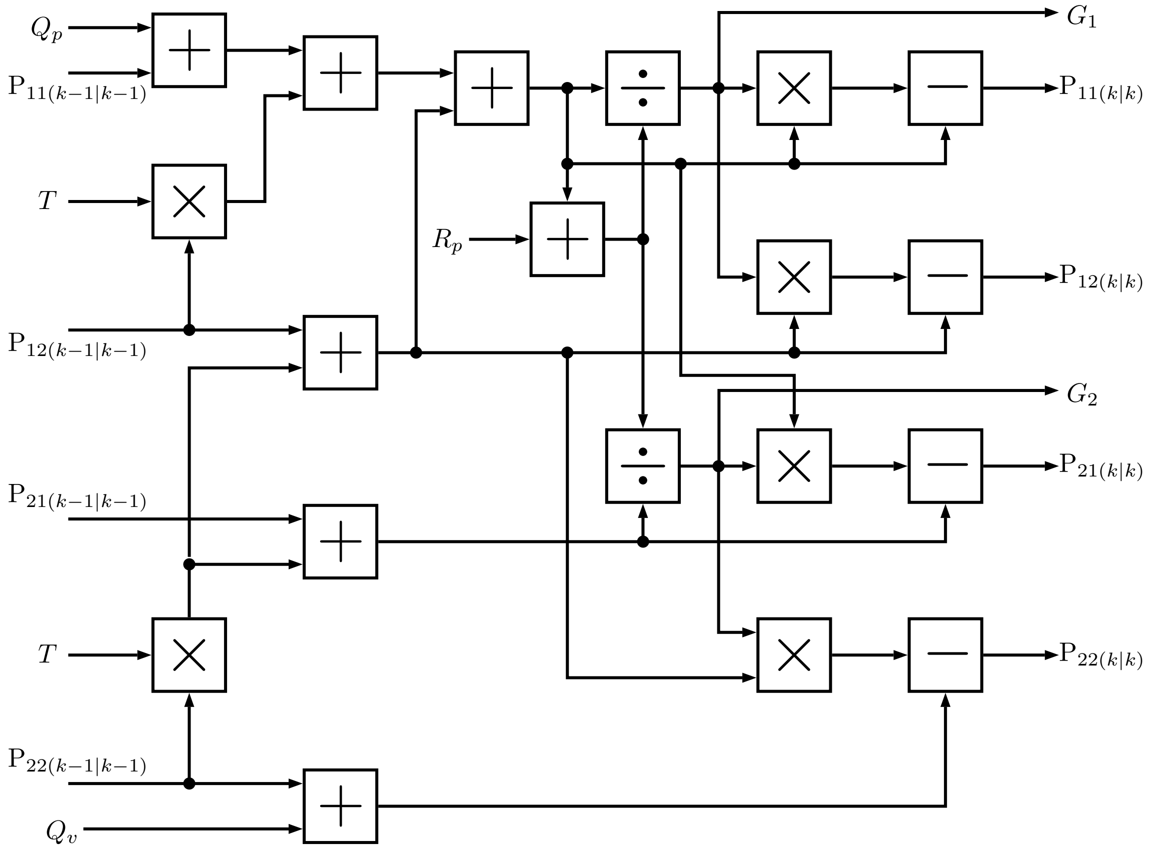
7.4.3 Implementation of the Velocity Estimation Circuit

After the relative position is estimated, the velocity is computed using the discrete Kalman filter formulation (7.15) established in section 7.2.3. However, the hardware implementation of the algorithm is only possible if the numerical relationships are broken down into scalar form, which will allow the FPGA digital circuit development process to use simple arithmetic adders, subtracters, multipliers, and divisions. Accordingly, equation (7.19) is formulated.

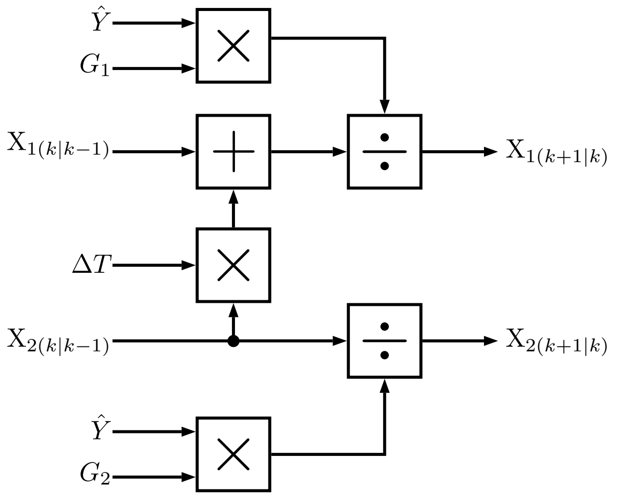
$$\begin{aligned}
X1_1 &= X_1 + \Delta T X_2, \\
X1_2 &= X_2, \\
P1_{11} &= P_{11} + \Delta T P_{12} + \Delta T P1_{21} + Q_p, \\
P1_{12} &= P_{12} + \Delta T P_{22}, \\
P1_{21} &= P_{21} + \Delta T P_{22}, \\
P1_{22} &= P_{22} + Q_v, \\
G_1 &= P1_{11}/(P1_{11} + R_p), \\
G_2 &= P1_{21}/(P1_{11} + R_p), \\
X_1 &= X1_1/G_1\hat{Y}, \\
X_2 &= X1_2/G_2\hat{Y}, \\
P_{11} &= P1_{11} - P1_{11}G_1, \\
P_{12} &= P1_{12} - P1_{12}G_1, \\
P_{21} &= P1_{21} - P1_{11}G_2, \\
P_{22} &= P1_{22} - P1_{12}G_2.
\end{aligned} \tag{7.19}$$

The hardware circuit of the discrete formulation is developed in two major parts. In the first part $P_{11}, P_{12}, P_{21}, P_{22}$, and G_1, G_2 are evaluated by arranging simple arithmetic circuits as shown in Figure 7.12(a). Similarly, in the second part X_1 and X_2 are evaluated as shown in Figure 7.12(b).

The circuit consists of 12 adders and 13 multipliers that take only 9 clock cycles to evaluate the results. Similar to a vision processing circuit, the Kalman filter circuit is operated at 50 MHz (FPGA clock speed); therefore, the filter is able to estimate velocity information before new image processing data are available to the circuit.



(a) First part of FPGA based Kalman Filter.



(b) Second part of FPGA based Kalman Filter.

Figure 7.12: FPGA Circuit of a Two State Kalman Filter.

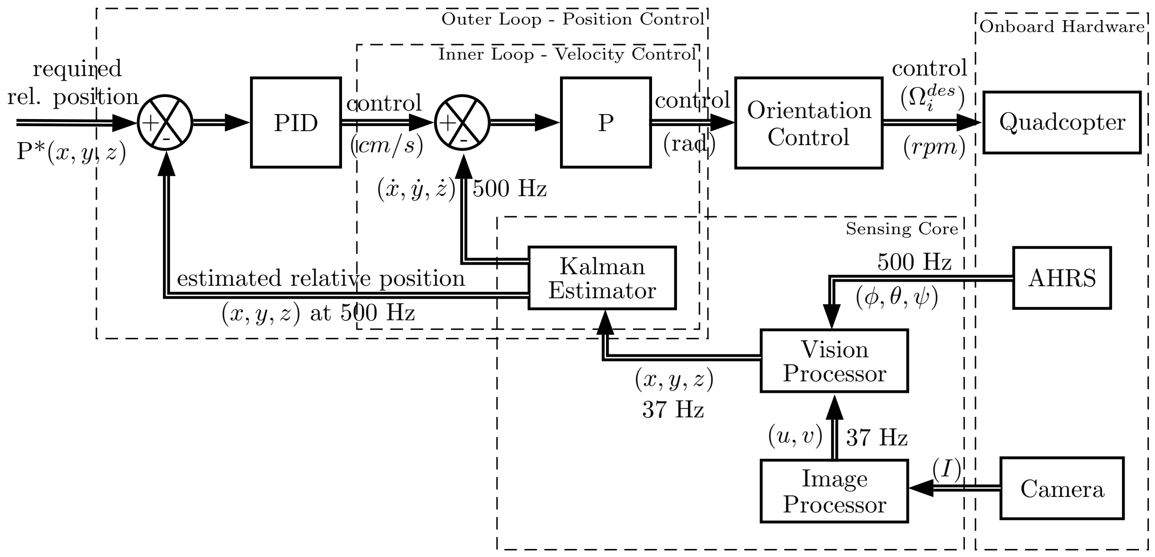


Figure 7.13: Overview of the FPGA Hardware of the Visual Servoing System.

7.4.4 Implementation of the Visual Servoing Control Law

The implementation of the control law requires discretization of the cascaded control formulation presented in equation (7.18). Since the outer control loop is a type-B PID control law, which was already developed in Section 6.3.5.1, the circuit is reused in this development. However, a minimal modification is done as the inner loop of the servoing control law is based on a proportional controller instead of a PID controller.

An overview of the FPGA hardware implementation of the complete servoing control technique is presented in Figure 7.13. Using the visual feedback from the camera (37 Hz) and the body orientation information from the AHRS (500 Hz), the position information is estimated by the vision processing unit at 37 Hz, which is equivalent to the camera frame rate. The Kalman filter then estimates the velocity information at 500Hz. The inner loop controller operates at 200Hz and the outer loop operates at 100 Hz. Since, motion on all three axes are independent of the each other, three single input - single output (SISO) cascaded PID are employed for complete implementation of the visual servoing controller. Because translational motions of a quadrotor are achieved through change in orientation, once the control signal is generated for motion on each axis, the orientation controller calculates the required orientation to move along the desired axis of motion. Finally, the control decoupler (introduced in chapter 6) calculates required speeds for all four motors to achieve the orientation setpoints. To track the motion of the landing target, desired relative motion between the vehicle and the target is set to zero. Similarly desired relative altitude is varied to obtain climb/descent from a certain relative altitude.

7.5 Performance Evaluation of the Visual Servoing System

In order to evaluate the proposed visual servoing technique, real time experiments were carried out in an indoor environment, and also include the calibration of sensory modules. The quadrotor developed in Chapter 3 was used as the platform for functionality and performance validation of the controller. The following sections present the hardware used to deploy the visual servoing system in the quadrotor. A test bed was designed to calibrate the sensor and evaluate its functionality and performance. Finally, each digital circuits were evaluated and the performance result of the overall system was analyzed.

7.5.1 Hardware Details

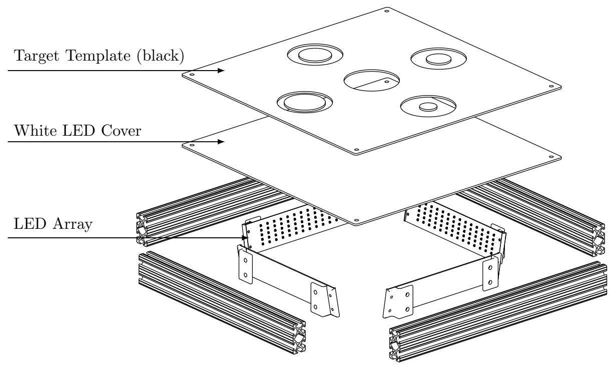
Vision cores were deployed on the same FPGA hardware that holds the low level controller of the quadrotor. A detailed construction of the FPGA hardware is presented in Section 3.3.5. As previously mentioned, the Cyclone III (40F484C6) FPGA chip holds 39,600 logic gates and the chip is operated at 50MHz. The AHRS unit and two 16MB SRAMs are hardwired to the FPGA chip to support image processing and vision processing. Two external hardwares are designed and constructed to support the visual servoing system: a camera equipped with an IR-Filter and an IR-LED back-lit circular landmark/target.

7.5.1.1 The Custom Lens-Camera Assembly

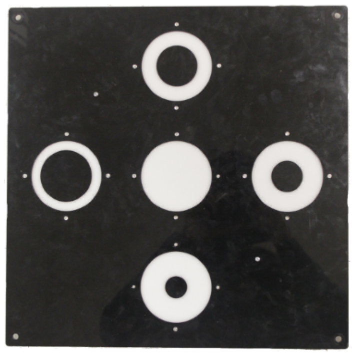
A custom designed lens-camera assembly was constructed from an off-the-shelf 2-Mega Pixels image sensor to have full control over the incoming pixel information and to exploit its full capacity (i.e., update rate, exposure time, shutter width etc.). The image sensor was wired directly to the FPGA chip. In order to decrease computational load associated with camera control and consequently increase the data output rate, only one quarter of the image sensor (1024×768 pixels) was used instead of the full frame. In addition to that, the sensor was configured to capture only grayscale images as opposed to full colour images, which require higher data acquisition time. For the noise suppression at a mechanical level, an infra-ray pass filter was mounted on the lens of the camera. The wavelength of the IR-pass filter (950 nm) was chosen to match the wavelength of the incoming IR-light from the landmark, restricting it to see only the circle of the landmark. This is the same camera that was designed and constructed in Section 3.3.6. However, for this application only the IR-pass filter was inserted into the lens mount of the camera.

7.5.1.2 Design & Construction of the Landing Target

A generic landing marker was designed that holds five 3 inch diameter circles laser-cut from an acrylic plastic. This five circle based target was designed and constructed to accommodate future development of the vision processing algorithms. However, in this study, only the center circle is used as the algorithm is designed to work with one circle only. Similar to the approach reported in [88], the target was back lit with four IR-LED arrays that have a wavelength equivalent to 950 nm. An illustration of the target and the camera is presented in Figure 7.14. In order to have a precise



(a) Mechanical Construction of the target.



(b) Target after development.

Figure 7.14: Landing target developed to aid vision algorithm.

manufacturing tolerance, the circular template was laser cut from a black acrylic plastic. To diffuse the IR light, the LEDs were covered with a white acrylic plate that has enough transparency to allow IR light to pass through.

7.5.2 The Test & Calibration Bench

In order to support the development process through testing and calibration of the sensors, a bench was designed using off-the-shelf aluminum framing structures. The frame has already been introduced previously in Section 6.5. Unlike the previous setup, the quadrotor here is equipped with the custom designed camera as shown in Figure 7.15. A VGA monitor was connected to the FPGA hardware to collect image information for post processing and data analysis. A PC was connected to the FPGA to capture real-time data of the avionics module and to configure the camera parameters for testing. The target, the quadrotor avionics and processors and the quadrotor motors are powered with their respective power supplies. The quadrotor along with the camera are mounted on the test bed to restrict the motion to only one rotational degree of freedom (roll/pitch) and one translational degree of freedom.

In the representative arrangement of the test bench setup presented in Figure

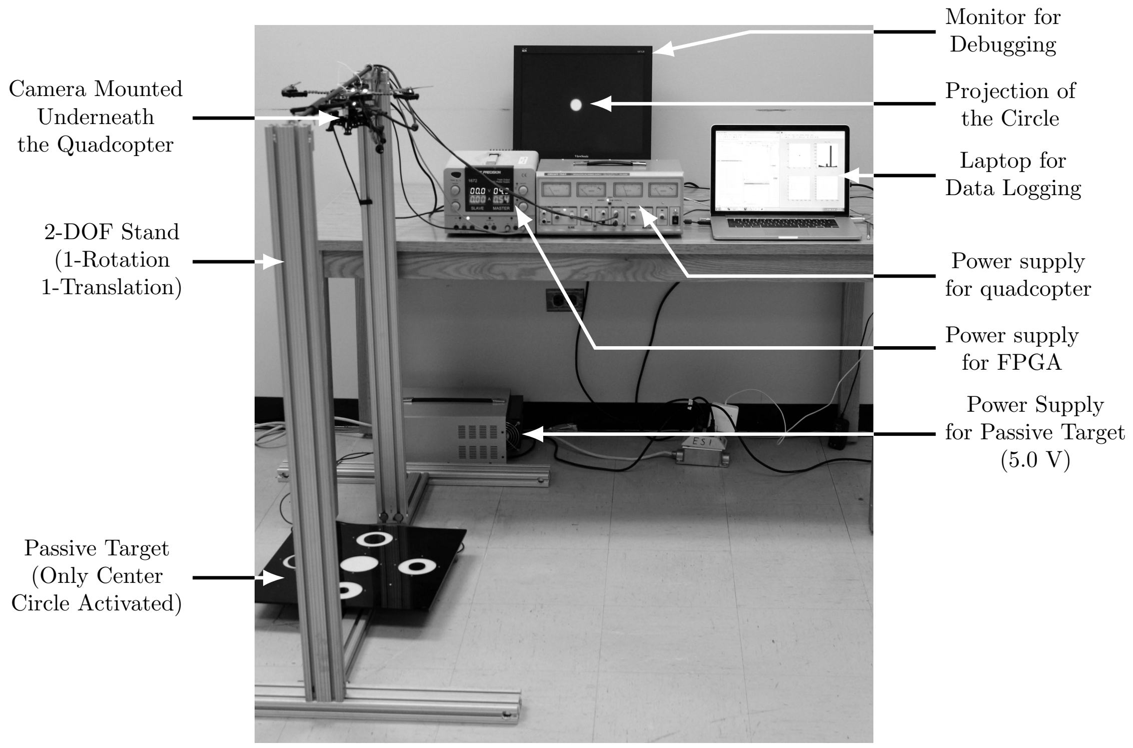

Monitor for Debugging
Camera Mounted Underneath the Quadcopter
Projection of the Circle
Laptop for Data Logging
2-DOF Stand (1-Rotation 1-Translation)
Power supply for quadcopter
Power supply for FPGA
Power Supply for Passive Target (5.0 V)
Passive Target (Only Center Circle Activated)

Figure 7.15: Test setup during evaluation of vision processing

7.15 the lens-camera assembly mounted underneath the quadrotor is directly looking towards the target. It is evident from the camera output in the monitor that only the middle circle of the target is active, a requirement of the proposed visual servoing method.

7.5.3 Functional Validation of the Image Processor

First, the functionality of the contour tracking is validated with a simple test, wherein a processed image was captured when the center of the quadrotor was aligned with the center of the target (validated mechanically) and the quadrotor was horizontal (validated through AHRS information). The estimated centroid value was $(0, 0)$ as expected from the image processing circuit. Since the image processing core directly relies on the image acquisition circuit, thresholding circuit and the feature extraction

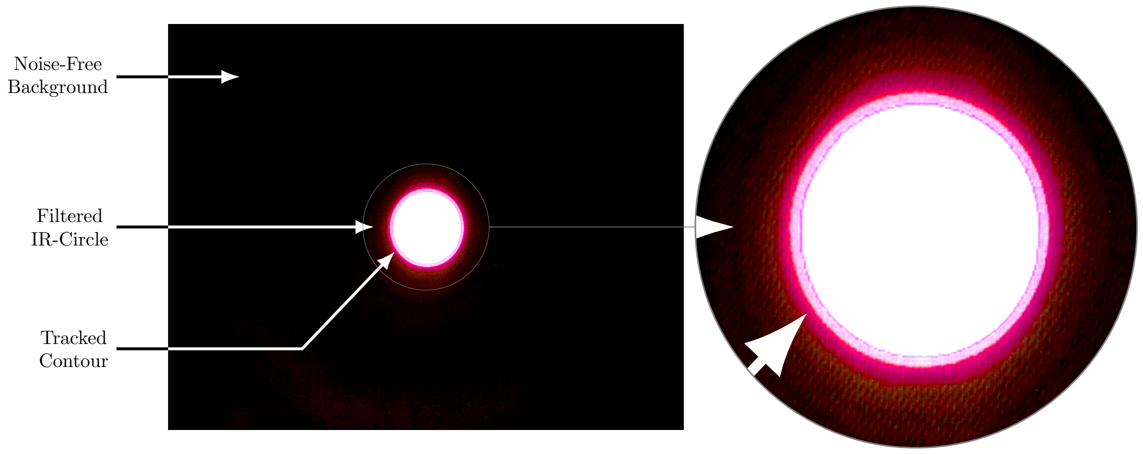


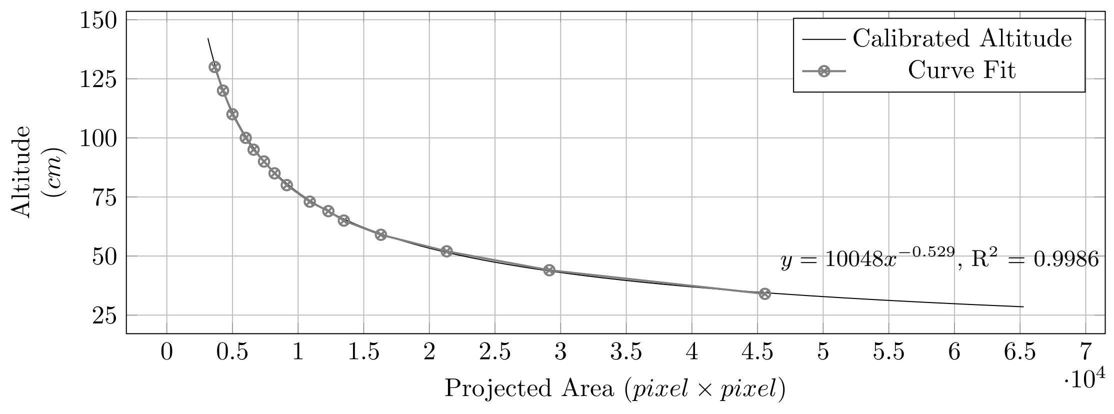
Figure 7.16: Contour of the circle is identified through image processing and displayed on a VGA monitor.

circuit, it is safe to claim that each digital circuit is as functional as the complete vision processing unit. The output of the image processor is presented in Figure 7.16. The representative image was taken from the VGA output and only shows the input and output of the image processing core. The white circular blob in the image is the projection of the illuminated circle of the target and the the red contour overlaid on the blob confirms the contour tracking operation of the image processing core.

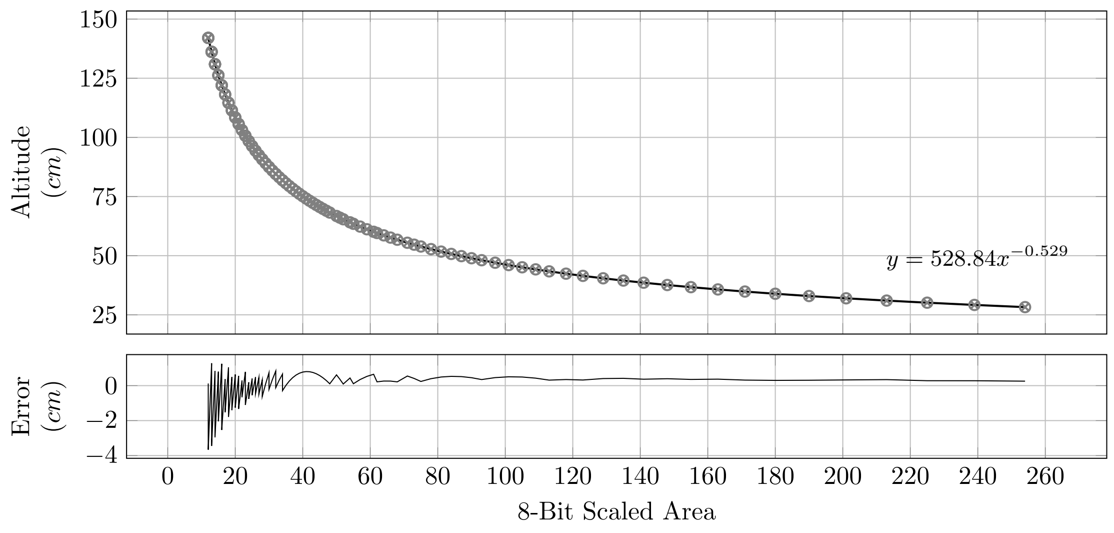
7.5.4 Altitude Calibration

As established earlier, the vision processor requires altitude information for accurate pose estimation and the implementation of the altitude formulation (equation (7.9)) is not suitable (involves square-root function) for FPGA hardware. Therefore, an identification of the relation between altitude and the area of the projected circle was carried out through calibration. Accordingly, the area of the projected blob in the image plane was recorded at different altitudes of the quadrotor. The calibration result presented in Figure. 7.17(a) suggests that the altitude is inverse square-root of the area of the blob.

However, the area information of the blob is too large to define a (LUT) in



(a) Function identification from calibration.



(b) Function scaled for FPGA LUT implementation.

Figure 7.17: Altitude calibration graphs.

the FPGA hardware. In order to solve this problem, the area is scaled to 8-bit data and the associated function is synthesized into the FPGA hardware during the compiling time using an 8-bit LUT. The scaling is achieved through normalizing the area information for 8-bit integer data. As a by-product of the scaling approach, a maximum of 2% error was introduced at the higher altitude regime as shown in Figure 7.17(b). However, the error can be reduced if a higher order data bit is used for scaling instead of an 8-bit number, which can offer more resolution.

7.5.5 Experimental Observations of the Vision Processor

After the parameters for altitude estimation circuit were integrated into the vision processor, the performance of the vision processing algorithm was evaluated for two conditions; namely, for pure rotation of the quadrotor and for pure translation of the quadrotor with respect to the landmark. The experiment was repeated at two different heights (at 70 cm and 100 cm), to evaluate the dynamic performance of the vision processing circuit.

7.5.5.1 Pure Body Rotation

The aim of a pure body rotation test is to demonstrate the effectiveness of the rotation effect compensation circuit as well as the vision processing circuit. First, a test was conducted by keeping the quadrotor at 70 cm height (0, 0, 70)cm from the target while inducing rotation (by hand) to the quadrotor body that holds the camera. Next the same test was performed for relative positions at (0, -50, 70)cm and at (0, 0, 100)cm.

Figure 7.18 presents the estimation results for the relative distance of (0, 0, 70)cm. It is evident that only the position measurement on the x axis is affected due to the induced rotation about the y axis (pitch) as seen in the top left of the figure. The recorded data for error in the x-measurement confirm that the rotation compensation circuit has corrected the measurement error of ± 40 cm down to ± 6 cm. However, some altitude error was observed which could be attributed to the induced body rotation. This was expected, because the basis for altitude measurement adopted here was just the projected area of the target, which could be altered either by changing height or through change in body rotation. In addition, the calibration was conducted only for change in height. Hence, the observed altitude error can be reduced further by incorporating calibration information for both change in body rotation and change in height.

To further evaluate the effectiveness of the rotation effect compensation in presence of translation along with induced body rotation, the target was placed at an offset of -50 cm; however, the altitude remained the same at 70 cm. Unlike the previous test, roll motion was induced in this case instead of pitch motion; therefore, an error was observed in the y motion estimation instead of the x motion as seen in Figure 7.19(top left). Without the rotation effect compensation, the estimated y motion varied from 40 cm to -60 cm though the target was stationary at -50 cm. After the rotation effect is compensated for, using AHRS input, the measurement error is reduced to $\pm$ 15 cm. However, due to the added offset in the relative target position, the error in the altitude measurement is shifted by 10 cm as compared to the previous test results.

In the next experiment, the quadrotor was lifted by 30 cm, moving to a new relative position of (0, -50, 100) cm. The objective is to evaluate the exclusive effect of altitude change while the system is excited with roll motion. The performance results are presented in Figure 7.20. It is evident from the center-left plot in the figure that the y position measurement varies from 80 cm to -100 cm while the target is at -50 cm, thereby creating an error band of 180 cm (-50 cm to 130 cm). When compared with the previous experiment, the error is increased by almost 200%. However, the performance of the rotation effect compensation in measuring translational position is similar in both cases. In both cases the error band after rotation effect compensation is around 30 cm ($\pm$15 cm).

These static tests conclude that the performance of the vision processing system, in particular the rotation effect compensation result, is more reliable as the relative position gets close to the zero, irrespective of the relative altitude between the camera and the target. However, as the altitude increases the error in the altitude measure-

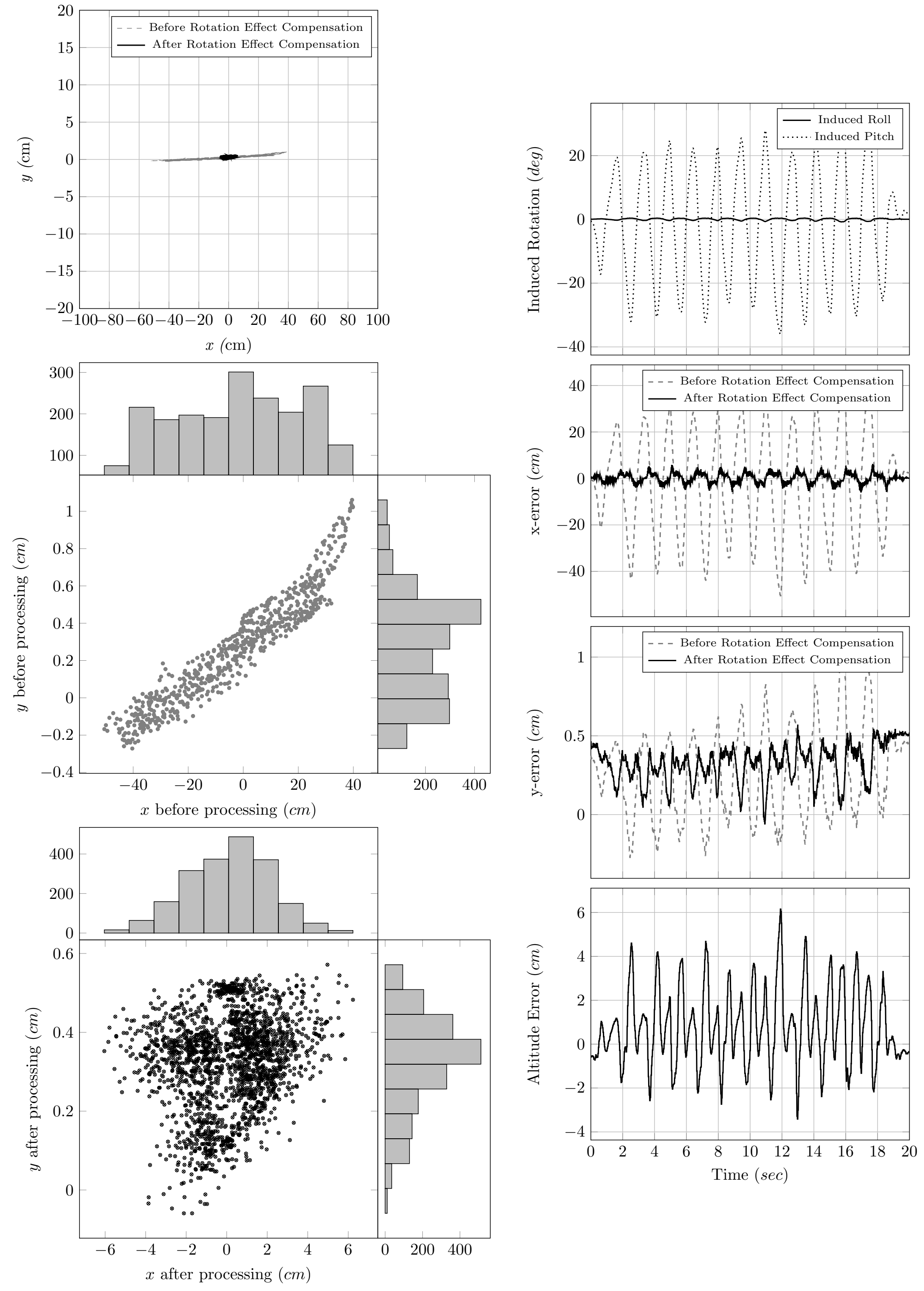


Figure 7.18: Results during pure body rotation. Relative distance $(0, 0, 70)$.

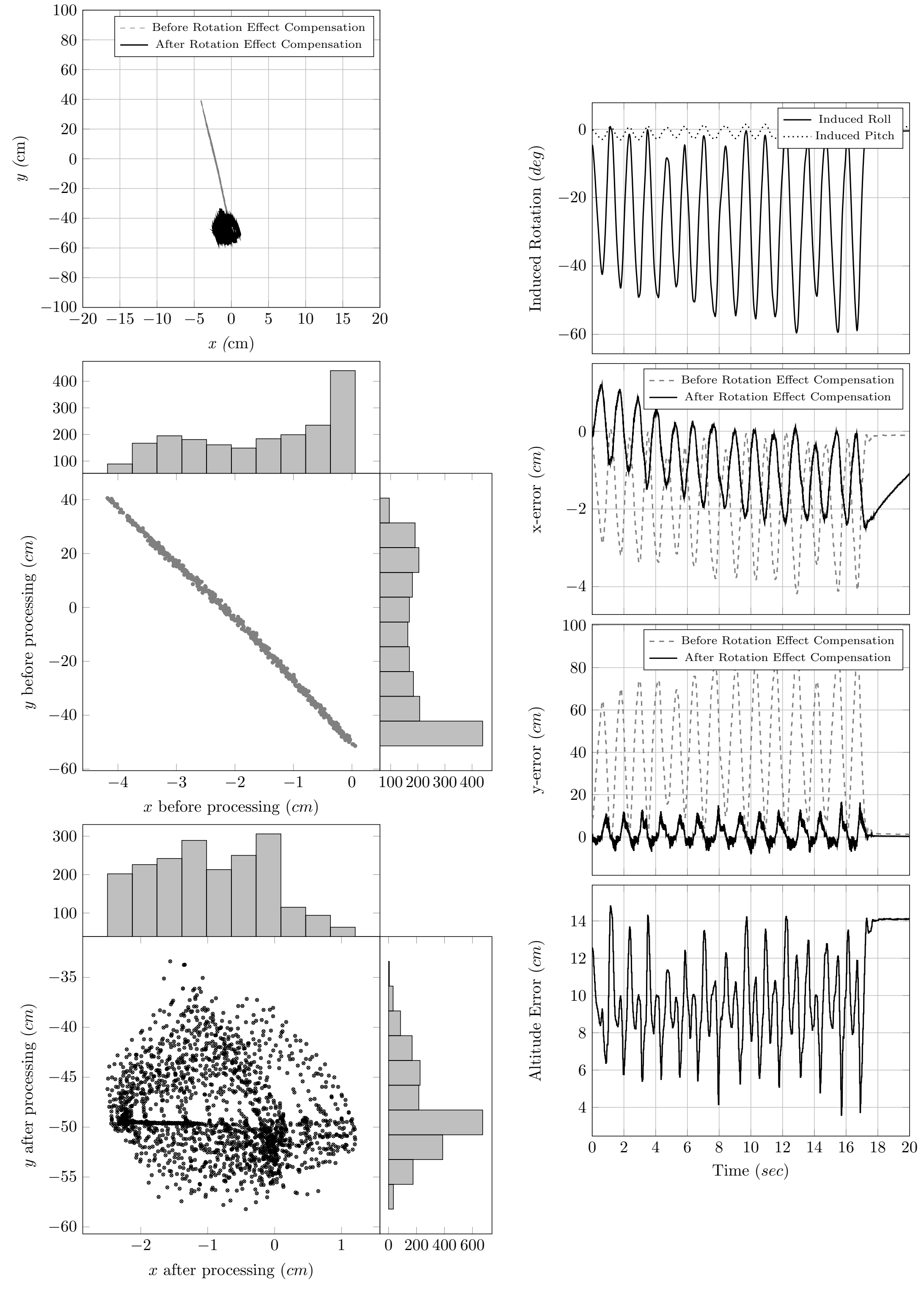


Figure 7.19: Results during pure body rotation. Relative distance $(0, -50, 70)$.

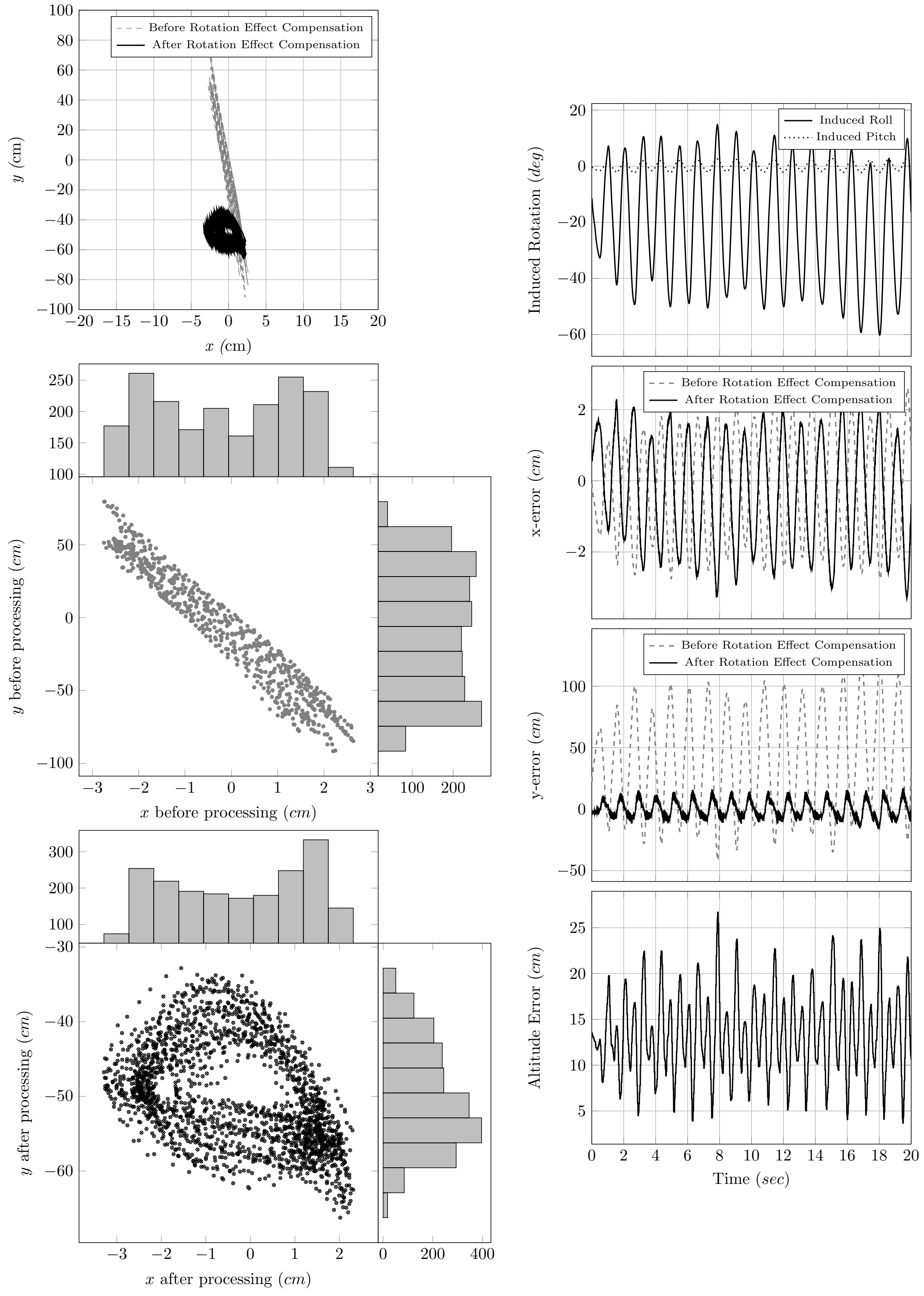


Figure 7.20: Results during pure body rotation. Relative distance $(0, -50, 100)$.

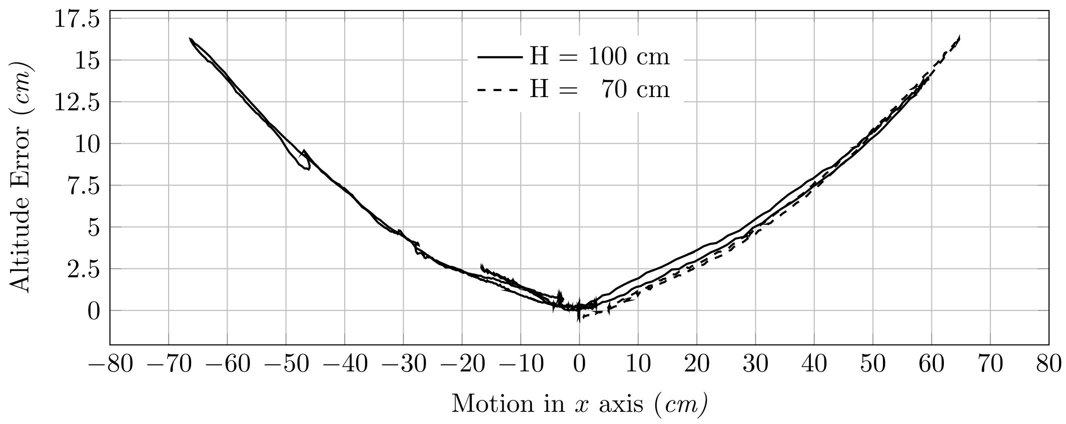


Figure 7.21: Estimation results when the quadrotor is translating on one axis.

ment increases provided that the camera is away (not aligned) from the target in horizontal directions. To further test the effect of translation on the altitude error, an experiment was conducted based purely on translation of the target.

7.5.5.2 Pure Translation

In this experiment, the quadrotor was kept static at a 100 cm height in a flat orientation (0° roll and 0° pitch) while the target was moved on the x axis. The aim is to observe the impact of altitude measurement due to pure translation. The experiment was repeated for a relative height of 70 cm to verify the effect altitude change for the same horizontal motion. The results are presented in Figure 7.21.

The results suggest that error in altitude measurement increases almost linearly as the target moves away from the vehicle and the phenomenon is consistent at different altitudes. This is obvious as the altitude is solely dependent on the area of the projected circle. In addition, the identification of the function between altitude and the area of the projected blob was conducted for the no relative translation condition (aligning the z axis of the camera with the vertical axis of the target passing through the center of the circle in the target). Therefore, as the target moves away from the

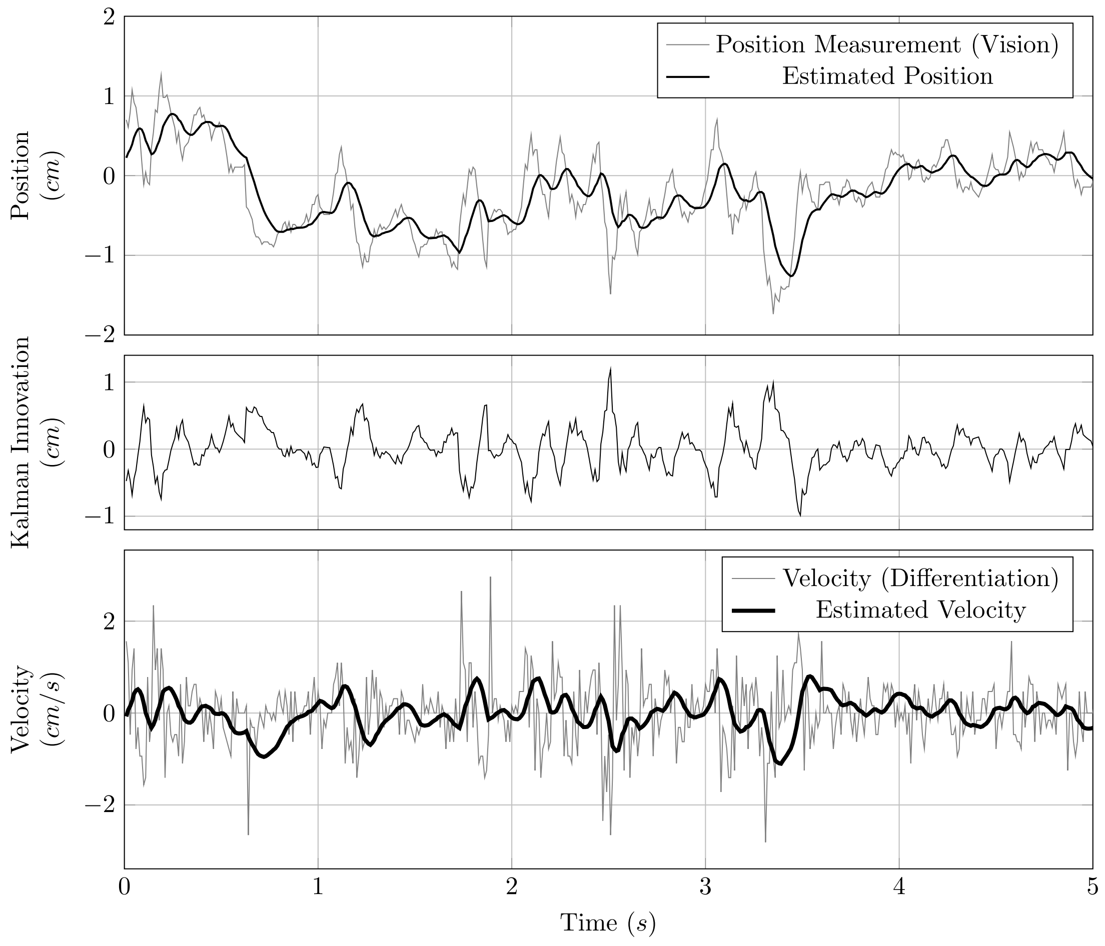


Figure 7.22: Representative responses of Kalman states for x-axis. Data recorded during position hold.

field of view of the camera, the projected area of the blob becomes more oval than circular, leading to a reduced projected area on the image plane. Subsequently, the calibration result evaluates the altitude information for the corrupted area and returns a higher altitude value leading to erroneous information. However, as there is a consistent relation between translational motion of the target and the altitude error, this can be compensated in the vision processing circuit in future revisions.

7.5.6 Performance Validation of the Velocity Estimator

The functionality and performance of the FPGA implemented Kalman Filter were verified by comparing the estimated velocity with numerically differentiated position measurements. A representative set of Kalman states for motion along the x-axis are presented in Figure 7.22. The data were recorded for a steady position hold over the target. The position and velocity result confirm the successful implementation of the the discrete Kalman Filter in FPGA hardware. It is also evident from the responses that, the filter not only estimates the velocity but also reduces noise when compared to the vision based position measurements. However, the noise suppression feature introduced lag to the estimated data. Since an error based control law is highly sensitive to noisy measurements, a feedback signal containing minimum lag is preferred to a noisy data. In addition, noisy feedback signals inhibits from using higher order gain values in the control law, which in turn reduces robust control operation. Thus, the estimated result from the Kalman Filter is fed to the control law for further development and performance evaluation.

7.5.7 Experimental Results of Visual Servoing

The vision based position control law was validated through three different experimental maneuvers; namely, climb, hold position, and descend. The motion was prescribed through a set of step setpoints: first climb from 60 cm to 100 cm while holding its relative horizontal translational motion to zero, then remain at that position, and finally drop the altitude back to 60 cm. Since, the vision processing system cannot estimate relative position information when the quadrotor is either too close to the target (<30 cm) or too far from the target due to the fixed focus lens, the experimental maneuvers are limited within the aforementioned boundary. It needs to be noted that, visual servoing is conducted with free-flight tests with the quadrotor developed

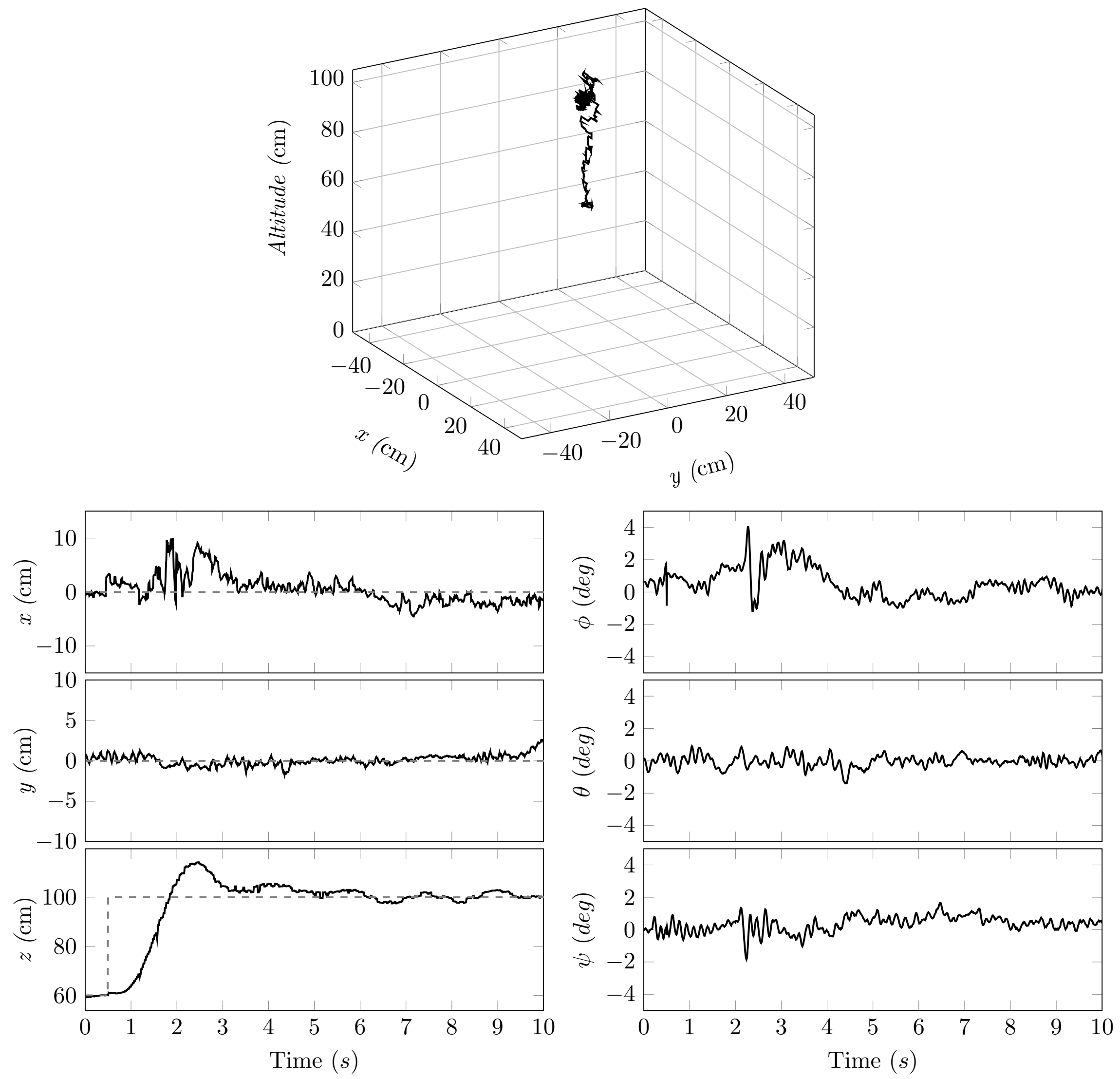


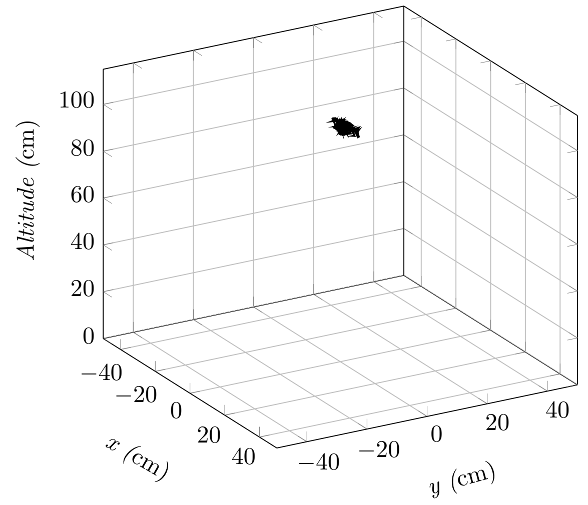
Figure 7.23: Time history of position and attitude information while the quadrotor is climbing from 60 cm to 100 cm relative to the visual target.

in Chapter 3; wherein, the motor driver that actuates the motors of the platform is the one developed in Chapter 5 and the orientation control employed in the vehicle is the cascaded ADRC-PID control law developed in Chapter 6. Essentially, this experiment demonstrate results, where all components developed in thesis are working together.

For ease of data analysis and clear visual representation of the recorded response each maneuver is presented with separate response curves. Figure 7.23 shows the history of position and attitude variables of the quadrotor during a climb. The dashed line in the figure presents relative set-point positions during the test. The desired horizontal position and translational velocities are prescribed as zero in this case. It can be observed from the response that the altitude control response settled at about 3 sec with an overshoot of 20 cm at 2 sec. During the overshoot a drift was also noticed on the x-axis, which was settled as soon as motion along the z-axis was stabilized. The tracking error was found to be within ± 10 cm on the x axis and ± 5 cm on the y axis. Figure 7.24 shows the time history of position and attitude information during the position hold maneuver. Once the aerial platform reaches its stability over the target, it can hover with minimal adjustment in the orientation control as seen in the attitude data (ϕ and θ). As previously mentioned, yaw information is not critical to the position hold, therefore it was not affected by the visual servoing controller. Finally during descent the controller performed better than for the climb, as seen in Figure 7.24. This could be attributed to the setting of gain parameters, which were tuned for a descent maneuver. The setpoint was changed at 0.5 sec and the altitude response asymptotically reached its desired altitude in less than 1.5 sec. Unlike the climb controller, the descent controller did not overshoot. In all experiments the desired setpoint for the yaw controller was set to zero.

7.6 Conclusion

The chapter presents a practical design of a single chip, monocular visual servoing solution for a small-scale quadrotor that relies on an IR-illuminated passive landmark. The components developed to achieve the goal include image processing algorithms, a vision processing technique with a rotation compensation algorithm, a velocity estimation method, and a position controller to control the relative position over the



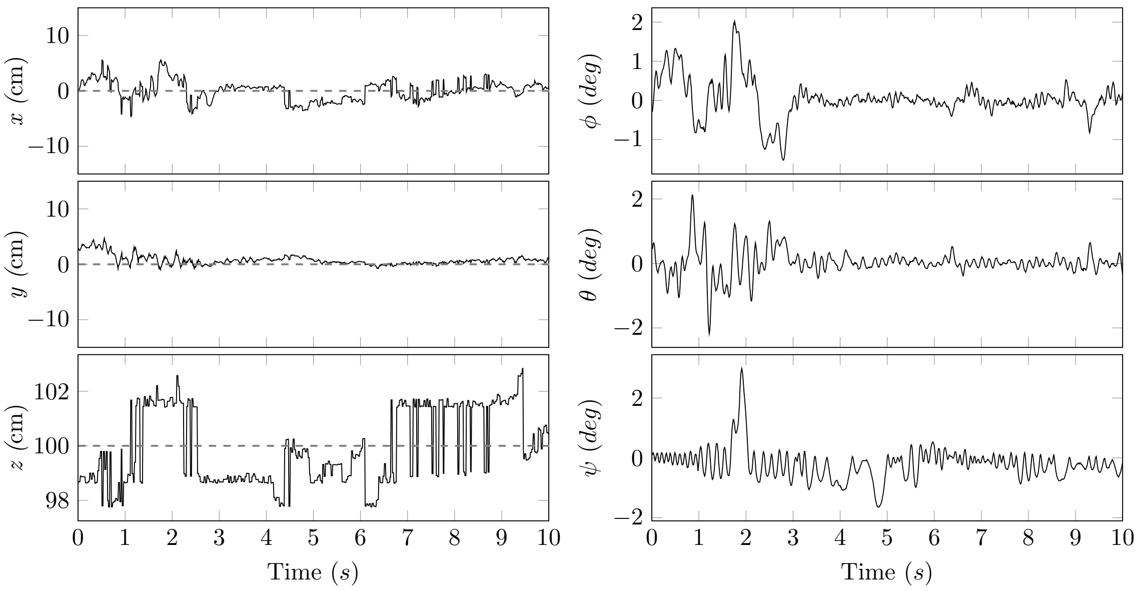


Figure 7.24: Time history of position and attitude information while the quadrotor is holding its position at 100 cm altitude relative to the visual target.

target. The image processing algorithm captures images and extracts feature information at the camera frame rate, while the vision processing algorithm estimates the relative position between the vehicle and the landmark with the help of the feature data. To increase the accuracy of the position estimation, the vision processing also compensates for the rotation effect of the vehicle that holds the camera. A cascaded PID control architecture is developed to track the target using feedback from the

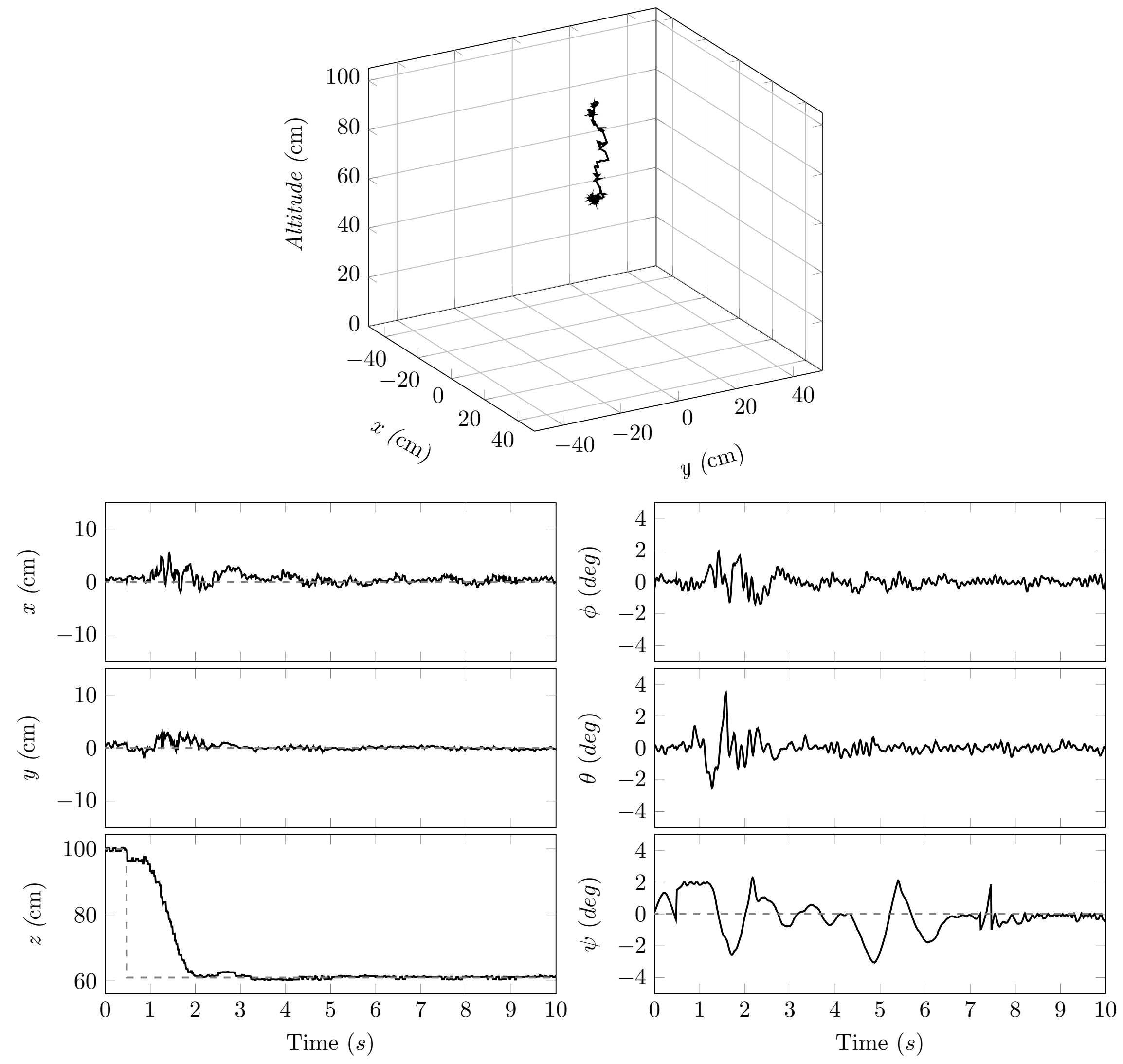


Figure 7.25: Time history of position and attitude information while the quadrotor descends from 100 cm to 60 cm relative to the visual target.

motion estimator. To support the control architecture with velocity feedback information, a Kalman filter based velocity estimation is adopted, which uses the vision based position information from the camera to estimate the velocity in real time.

To validate the visual servoing technique using an FPGA based quadrotor vehicle, all processing units are deployed exclusively on the same single chip FPGA hardware that operates the vehicle. Through the development of discrete time domain formulation digital circuits are designed and implemented in the FPGA hardware. Experimental results confirmed the functionality and performance of each monocular vision processing unit in compensating for vehicle rotation while reliably estimating the relative position between the target and the vehicle. The results also demonstrate the performance of the Kalman filter in estimating the relative motion. Finally, through experiments, the overall visual servoing action of the quadrotor is validated for three different maneuvers: climb to an altitude, hold position at that altitude with no translation, and return to the former altitude while not drifting in horizontal directions. The results confirmed the successful development and implementation of the proposed computationally inexpensive visual servoing technique suitable for a small-scale FPGA based quadrotor.

The simplicity of the technique leads to faster vision processing, almost at the camera frame rate (here, at 37 fps) and the FPGA implementation guarantees parallel implementation of the vision processor and other visual servoing cores without introducing latency to time critical flight control cores of the quadrotor (e.g., stabilization core). In addition, due to modularity and reconfigurability of the FPGA hardware, the visual servoing method is easily expandable to accommodate future revisions. Since the proposed algorithm is limited to translation estimation only, this version of the visual servoing method cannot be employed in a scenario where all six degrees of freedom information are required for autonomous maneuver control (i.e, hover over a floating vessel). A solution to overcome the limitation is to formulate a vision processing algorithm for an increased number of feature points in the landmark. This strategy is pursued as ongoing research work. This would allow the proposed monocular vision based control to mature as a sound method to serve the important function of providing motion estimation from a single perspective view required for fully autonomous control tasks of a multi-rotor aerial vehicle.

Chapter 8

Conclusion

The focus of this research was to explore design and control synthesis methodologies to automate an agile quadrotor UAV using FPGA technology. After careful review of the related literature, it was concluded that a ground-up design philosophy is better suited to realize the specified goals as opposed to constructing a prototype through utilization of existing (off-the-shelf) general purpose components. Evidently, the chances of success of this approach increase due to a deeper concentration of design goals required at every step of the development cycle. However, the ground-up approach is not free from pitfalls, as it may happen that existing technologies and design solutions that are proven to be technically sound and practically robust might get overlooked. Accordingly, this thesis was watchful not to employ established design methodologies wherever they were applicable, but at the same time, explored all possible avenues to extend existing knowledge. In addition, every effort was made to create the bridge between scientific curiosity and practicality of the implementation through evaluation of design solutions in a simulation environment and subsequently implementing them in the physical domain. The empirical evidence confirmed the functionality and effectiveness of the proposed ground-up design concept.

8.1 Review of the Contributions

The contributions of this thesis, as claimed in Section 1.3, focus on the design synthesis and the hardware implementation of the key components of an agile FPGA based quadrotor, which includes its sensors, actuators, low level stabilization control and high level visual servoing. Each of these components is discussed in the following order to qualitatively review the relevance of the contributions to address the challenges faced in the development of a small-scale agile quadrotor.

As the first step towards achieving the goals of the thesis, a quadrotor platform was designed and constructed, as reported in Chapter 3 that addresses the need for an agile vehicle to support robust stabilization control of the vehicle by reacting quickly to the control command. The thesis performed an extensive study of the available design ideas to increase the agility of the platform. The outcome of the review indicated that a quadrotor with small characteristic length can offer agile performance. However, small platforms come with limited real-estate, limited payload capability and flight endurance. Accordingly, a systematic dimension estimation approach is developed to determine an optimum size of the small-scale quadrotor that can offer designed payload loading and an optimum flight endurance. In order to conform with the design specifications, a monolithic platform construction approach is adopted in this research that employed a single PCB as the vehicle frame and single chip FPGA technology based processing hardware resulting in a relatively compact profile with the capacity to execute sensing, actuating, and control tasks, all in parallel.

In the second step of the thesis, multi-body dynamics and control of the quadrotor was developed to conduct simulation studies and to identify suitable control laws. Due to the interactions of components from multiple engineering domains, a bond graph graphical multi-domain modeling approach was chosen in this research, wherein the graphical nature and explicit power flow paths inherent in the bond graph formalism facilitated model construction and troubleshooting. The mainframe of the vehicle was assumed as one body, and all motor-propeller assemblies were considered as separate bodies connected to the mainframe via four revolute joints. Newton-Euler formalism with body fixed coordinates was used to model the dynamics of each rigid body. A DC motor model was used to incorporate the actuator dynamics. Rotor drag torque was assumed to be proportional to the thrust of the rotor. Unlike a single rigid body model, which ignores the effect of multi-body interactions that occur in a real system, the proposed comprehensive multi-body model reproduced actual dynamic behaviour of the system. The model results for different maneuvers generated through combinations of motor speed agreed with existing theoretical results. Open loop simulation study showed increasing roll and pitch angular speed due to a combination of two gyroscopic effects; one from the rigid body rotation and the other due to change in the orientation of the motor-propeller axis of rotation. The closed loop controlled response demonstrated the stabilization of the system from an initial roll, pitch, yaw and altitude to the desired steady state configuration.

The lack of performance in the off-the-shelf BLDC motor controllers in terms of update rate, bidirectional operation capability, power efficiency, and absence of closed loop control reported in the literature inspired the task undertaken in the next step of this research study. A BLDC motor driver was developed from the ground up using discrete MOSFET switches, which were commutated by the FPGA hardware. Three hall sensors were mounted around the periphery of the motors to identify the right quadrant of the rotor and the trapezoidal back-EMF based commutation sequence was generated by the firmware deployed in the FPGA hardware. The ground-up approach allowed the commutation to be executed at a significantly higher update rate (at 20kHz) when compared to the off-the-shelf motor driver (typically 50 Hz to 400 Hz, maximum 1kHz). The feature to write a custom commutation algorithm also allowed the deployment of a power efficient commutation algorithm previously established in reported literature. The result was a maximum of 4% power efficient motor driver (which varies with the operating speed) as opposed to classical commutation techniques employed in general purpose motor drivers. To implement the closed loop control law, various hall sensor based speed measurement techniques were studied in this research. However, the poor resolution observed in existing techniques when implemented in FPGA hardware led to implementation of an observer based speed measurement algorithm. Though the observer based technique consumed more FPGA resources, superior resolution led to the adopted speed observer for the development of the motor controller. The complete FPGA hardware implementation along with the PID based speed controller were experimentally evaluated, which provided good validation of energy efficient commutation, accuracy of the speed measurement technique and the speed control feature.

When a quadrotor is deployed in an outdoor environment, the robustness to the external disturbance and internal noise like wind gust and sensor noise must be satisfied. Many researchers have proposed stabilization control algorithms to meet the criteria. However, the numerical complexity associated with the algorithms and the dependency of the algorithms on accurate dynamics formulations makes it impractical for deployment in small size, low power consumption, high speed FPGA hardware. In order to address this tension between practical implementation and robust performance of the controller, an innovative cascaded attitude stabilization control law is developed in this thesis. The corresponding experimental evaluations, performed both in nominal hover and hover under wind-gust, confirmed the robust control performance for time varying non-linear systems such as the quadrotor. The proposed cascaded control law based on the ADRC and PID algorithm demonstrated a consistent stabilization performance throughout the operating range, wherein the inner loop PID controller stabilized the high acceleration of the agile vehicle and the outer loop ADRC algorithm efficiently maintained the attitude of the vehicle. Beside the robustness, the control law also allowed the harnessing of the agility of the vehicle, as concluded by the faster settling time (average 426.48 ms) to a step input.

The next study of this research was to integrate high level autonomy to the single chip FPGA based quadrotor by employing a visual servoing system along with the aforementioned features of the platform. Due to the size and weight constraints, a 3 mm lens camera equipped with Infra-red (IR) pass filter was installed underneath the quadrotor platform to collect visual cues from a target/landmark. The target was designed from a circle with a known diameter and was back-lit with IR LEDs.

An image processing circuit, a vision processing circuit, a velocity estimation circuit, and a target tracking control circuit were designed and deployed exclusively on the same FPGA hardware that holds the rest of the processing modules. The image processing circuit collected image information (projection of the circular target onto the image sensor), converted it to a grayscale image, performed thresholding to filter noise, and finally implemented contour tracking to extract feature information (centroid and area of the projected circle). To perform time efficient image processing, the FPGA hardware was supported with two SRAMs. While an image was captured, and stored into one SRAM, the FPGA performed contour tracking on the previously stored image in the second SRAM. The roles of the SRAMs are swapped after their respective tasks are completed. Using the feature information from the image processing core, the vision processing circuit estimated relative position/location of the quadrotor with respect to the center of the target circle. A vision processing circuit was formulated based on a pinhole camera model that compensated vehicle angular motion using real-time orientation information from the AHRS. The result was a real-time estimate of relative position. In order to implement a tracking control law, the relative velocity between the body and the target was estimated through a Kalman filter that takes the relative position as the input for state estimation. The tracking control law was based on cascade control architecture of two PID controllers, wherein the inner loop controller controls linear velocity of the vehicle and the outer loop controls the relative position of the vehicle. The outcome was a visual servoing system that uses a single perspective view from flying robots.

As part of the thesis, all the aforementioned modules were validated using experimental results. Since software tools were utilized to formulate and subsequently evaluate designs in a virtual environment, modern engineering products are being developed more efficiently than was ever possible. Admittedly, it needs to be acknowledged that it is difficult to exhaustively model real world constraints using these software tools. Hence, the practicality of a design can only be convincing when its physical prototype can achieve all the design goals. Accordingly, the experimental results conducted with the developed prototype quadrotor achieved the design goals as specified in this research study.

8.2 Limitations and Future Research Directions

In any organized research study, a set of boundaries is predefined that reflects the research goals; being too ambitious can result in unmanageable complexities of the available resources while being too conservative may defeat the aim of generating new knowledge. Hence, a trade-off is made during the selection of the boundaries so that likelihood of success is maximized. However, even carefully defined boundaries can result in limitations to the conducted research. Unlike any other related research, this study has identified many avenues for improvement that can be addressed through future research.

The proposed monolithic design of the quadrotor platform presented in Chapter

3 was achieved using three separate PCB boards (mainframe board, FPGA board, and camera board) instead of just one (mainframe PCB) that had enough real-state to populate all components. Essentially the rationale was to increase the capability of debugging at the development stage as the thesis adopted a ground-up development philosophy for constructing the prototype. This may have hindered the possibility of having more flight time from the platform than what is currently reported in this thesis. In order to both fully exploit the performance of the platform and improve flight time by reducing size and weight even further, all components can be populated on the mainframe PCB along with the appropriate power circuitry in future revisions. In addition, a detailed vibration analysis of the platform may help in reducing components associated with vibration damping of the platform, which has been empirically designed in this first iteration of the vehicle design.

In the current form of multi-body model developed in Chapter 4, the simulation study is limited to stabilization control performance only. However, considering the modularity of the bond-graph model, a future model extension may include integration of position control beside the existing stabilization and altitude controller, to help study its horizontal motion in 3-D spaces. In addition, higher-fidelity aerodynamics contributed by the propellers' blades can be included to study the aerodynamics effect more accurately in future research.

Inspired by the successful implementation of the FPGA technology based BLDC motor driver (motor commutation) presented in Chapter 5, a PID controller was implemented to control the speed of the motor. However, the PID controller is susceptible to integral windup, which may have a negative impact on the overall robustness of the quadrotor. Since the thesis has identified a robust controller (ADRC) while exploring stabilization control algorithms for the quadrotor, improving the motor speed controller with ADRC technology can result in an integral wind-up free controller as well as improve its robustness even further. Therefore, this work would highly benefit from further research on ESO algorithm designs such as 2-state ESO algorithms [70] to estimate disturbance from motor velocity input. Hence, implementation of the ADRC technology to control motor speed would further enhance the speed control performance of the motor driver/controller.

In this thesis, the robustness of the cascaded attitude stabilization controller under wind disturbance was proven through experimental evaluation assuming a SISO (Single Input Single Output) structure. However, the presented approach is limited to comparative study only, and extending it to absolute performance evaluation with MIMO experiment was deferred for future work. Although the existing literature provides numerical methods for estimating wind disturbance, eminent researchers [157] have criticized them for being inconsistent and ill-equipped to numerically represent its qualitative definition. Nonetheless, research with potential impact can be conducted if an accurate wind disturbance estimation model and measurement technique are developed, which would complement the accuracy of the simulation study as well.

The FPGA based visual servoing technique designed in this thesis employs a static brightness thresholding algorithm to search an IR-LED based passive target, which works well indoors and in a low light condition. However, operating outdoors in direct sunlight can result in an overexposed image [158] leading to compromised performance by the visual servoing system. Possible future research can explore this question by incorporating color into the passive target and filtering out the ambient sunlight [159]. In addition, increasing the number of feature points in the passive target would allow the vision processor to estimate 6-DOF motion with increased accuracy and increased resolution [82]. Admittedly, a higher number of feature points may generate a numerically complex algorithm, making it challenging for hardware implementation. Hence, it is expected that the findings of this research would provide the basis for the development of a simple yet robust FPGA implementable visual servoing system for future research.

8.3 Final Remarks

The dynamic capacity, robust stabilization control, and the autonomous visual servoing capabilities of the agile quadrotor designed and synthesized in this thesis were demonstrated by the individual abilities of the performance optimized platform design, efficient actuators, stabilization controller, and the monocular IR-LED based position controller respectively. The PCB based vehicle frame and the single FPGA based processing platform provided the base for achieving agile motion while featuring high payload capacity and longer flight endurance. In addition, the FPGA hardware allowed a computing platform for implementing high speed digital circuits (firmware) required to harness agility of the vehicle. The FPGA based motor driver/controller was shown to provide a maximum power efficiency of 4%, with the capability of operating in both clockwise and counterclockwise directions and updating command at 16 kHz, making it an efficient motor speed controller. The proposed ADRC based attitude stabilization controller demonstrated globally robust performance with an agile response proven by achieving asymptotic stability within 427 ms of the step commands. The controller also demonstrated superior disturbance rejection capability compared to a standard PID based stabilization controller when subjected to a disturbance in the form of wind gust. The single chip FPGA based visual servoing confirmed the autonomous position control and tracking of a passive target using a single perspective view from the quadrotor. In view of these achievements, it is claimed that this thesis has fulfilled its original goals of devising designs for an FPGA based quadrotor that can deliver the speed and robustness required to develop a SWaP constrained agile autonomous aerial platform.

Bibliography

[1] B. Argrow, D. Lawrence, and E. Rasmussen, "Uav systems for sensor dispersal, telemetry, and visualization in hazardous environments," in *43rd Aerospace Sciences Meeting and Exhibit*, vol. 4, 2005.

[2] Amazon Prime Air. URL: `http://www.amazon.com/b?node=8037720011`.

[3] C. Bolkcom, "Homeland security: Unmanned aerial vehicles and border surveillance," DTIC Document, 2004.

[4] V. Baiocchi, D. Dominici, and M. Mormile, "Uav application in post-seismic environment," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XL-1 W*, vol. 2, pp. 21–25, 2013.

[5] A. Rango, A. Laliberte, C. Steele, J. E. Herrick, B. Bestelmeyer, T. Schmugge, A. Roanhorse, and V. Jenkins, "Using unmanned aerial vehicles for rangelands: current applications and future potentials," *Environmental Practice*, vol. 8, no. 03, pp. 159–168, 2006.

[6] Y. Mulgaonkar, M. Whitzer, B. Morgan, C. M. Kroninger, A. M. Harrington, and V. Kumar, "Power and weight considerations in small, agile quadrotors," in *SPIE Defense plus Security*, pp. 90831Q–90831Q, International Society for Optics and Photonics, 2014.

[7] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[8] S. Piróg, M. Baszyński, and T. Siostrzonek, "The high-speed flywheel energy storage system," *Energy Storage, SCIYO*, 2010.

[9] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics & amp amp Automation Magazine*, no. 19, pp. 20–32, 2012.

[10] J. Han, "From pid to active disturbance rejection control," *Industrial Electronics, IEEE transactions on*, vol. 56, no. 3, pp. 900–906, 2009.

[11] R. Inovan, A. Ataka, H. Tnunay, M. Q. Abdurrahman, A. Cahyadi, and Y. Yamamoto, "A cascade controller for linearized quadrotor model," in *Advanced Robotics and Intelligent Systems (ARIS), 2014 International Conference on*, pp. 161–164, IEEE, 2014.

[12] M. R. Hossain, D. G. Rideout, and D. N. Krouglicof, "Bond graph dynamic modeling and stabilization of a quad-rotor helicopter," in *Proceedings of the 2010 Spring Simulation Multiconference*, p. 215, Society for Computer Simulation International, 2010.

[13] M. R. Hossain and N. Krouglicof, "Propeller dynamometer for small unmanned aerial vehicle," in *Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on*, pp. 1–5, IEEE, 2010.

[14] M. R. Hossain, T. Rahman, and N. Krouglicof, "An agile single board quadrotor providing "eye in the sky" capabilities for marine environments," in *Oceans-San Diego, 2013*, pp. 1–5, IEEE, 2013.

[15] M. R. Hossain, T. Rahman, and N. Krouglicof, “FPGA based active disturbance controlled quadrotor uav for marine environments,” in *Unmanned Systems Canada, 2013*, UVS, 2013.

[16] G. C. Bakx and J. M. Nyce, “UAS in the (inter) national airspace: Approaching the debate from an ethnicity perspective,” in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pp. 189–192, IEEE, 2013.

[17] DJI Products. URL: `http://www.dji.com/products`.

[18] S. H. Rhee and Y. S. Lee, “Unmanned flying vehicle made with pcb,” Nov. 29 2010. US Patent App. 12/954,975.

[19] C. H. Wolowicz, J. S. Bowman, and W. P. Gilbert, “Similitude requirements and scaling relationships as applied to model testing,” *NASA technical Paper*, vol. 1435, 1979.

[20] J. Roberts, “Enabling collective operation of indoor flying robots,” 2011.

[21] J. G. Leishman, *Principles of Helicopter Aerodynamics with CD Extra.* Cambridge University Press, 2006.

[22] A. J. Robinson, “A small scale quadcopter platform for robotic swarm development,” 2013.

[23] S. G. Fowers, “Stabilization and control of a quad-rotor micro-uav using vision sensors,” 2008.

[24] P. S. King, *System Design Methodology and Implementation of Micro Aerial Vehicles.* PhD thesis, 2014.

[25] C. Lehnert and P. Corke, “μAV-design and implementation of an open source micro quadrotor,” *AC on Robotics and Automation, Eds*, 2013.

[26] C. Lebres, V. Santos, N. F. Ferreira, and J. T. Machado, “Application of fractional controllers for quad rotor,” in *Nonlinear Science and Complexity*, pp. 303–309, Springer, 2011.

[27] J. M. Seddon and S. Newman, *Basic helicopter aerodynamics*, vol. 40. John Wiley & Sons, 2011.

[28] V. Mistler, A. Benallegue, and N. M’sirdi, “Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback,” in *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pp. 586–593, IEEE, 2001.

[29] P. McKerrow, “Modelling the draganflyer four-rotor helicopter,” in *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, vol. 4, pp. 3596–3601, IEEE, 2004.

[30] P. Pounds, R. Mahony, and P. Corke, “Modelling and control of a quad-rotor robot,” in *Proceedings Australasian Conference on Robotics and Automation 2006*, Australian Robotics and Automation Association Inc., 2006.

[31] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, “Quadrotor helicopter flight dynamics and control: Theory and experiment,” in *Proc. of the AIAA Guidance, Navigation, and Control Conference*, vol. 2, 2007.

[32] J. Kim, M.-S. Kang, and S. Park, “Accurate modeling and robust hovering control for a quad-rotor vtol aircraft,” in *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*, pp. 9–26, Springer, 2010.

[33] M. Bangura, H. Lim, H. J. Kim, and R. Mahony, “Aerodynamic power control for multirotor aerial vehicles,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 529–536, IEEE, 2014.

[34] A. Rodić and G. Mester, “The modeling and simulation of an autonomous quadrotor microcopter in a virtual outdoor scenario,” *Acta Polytechnica Hungarica*, vol. 8, no. 4, pp. 107–122, 2011.

[35] S. Bouabdallah, P. Murrieri, and R. Siegwart, “Towards autonomous indoor micro vtol,” *Autonomous Robots*, vol. 18, no. 2, pp. 171–183, 2005.

[36] T. Ersal, J. L. Stein, and L. S. Louca, “A bond graph based modular modeling approach towards an automated modeling environment for reconfigurable machine tools,” *Proceedings of IMAACA*, vol. 4, 2004.

[37] D. Karnopp, “Understanding multibody dynamics using bond graph representations,” *Journal of the Franklin Institute*, vol. 334, no. 4, pp. 631–642, 1997.

[38] W. Favre and S. Scavarda, “Bond graph representation of multibody systems with kinematic loops,” *Journal of the Franklin Institute*, vol. 335, no. 4, pp. 643–660, 1998.

[39] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System dynamics: modeling, simulation, and control of mechatronic systems.* John Wiley & Sons, 2012.

[40] A. Keemink, *Design, Realization and Analysis of a Manipulation system for UAVs.* PhD thesis, Master’s thesis, University of Twente, 2012.

[41] J.-M. Seo, J.-H. Kim, I.-S. Jung, and H.-K. Jung, “Design and analysis of slotless brushless dc motor,” *IEEE transactions on Industry applications*, vol. 2, no. 47, pp. 730–735, 2011.

[42] S. Chaithongsuk, B. Nahid-Mobarakeh, J.-P. Caron, N. Takorabet, and F. Meibody-Tabar, “Optimal design of permanent magnet motors to improve field-weakening performances in variable speed drives,” *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 6, pp. 2484–2494, 2012.

[43] D. Kos, M. Curkovic, and K. Jezernik, “Fpga based bldc motor current control with spectral analysis,” in *2006 12th International Power Electronics and Motion Control Conference*, 2006.

[44] A. Sathyan, N. Milivojevic, Y.-J. Lee, M. Krishnamurthy, and A. Emadi, “An fpga-based novel digital pwm control scheme for bldc motor drives,” *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 8, pp. 3040–3049, 2009.

[45] J.-Y. Chai and C.-M. Liaw, “Development of a switched-reluctance motor drive with pfc front end,” *Energy Conversion, IEEE Transactions on*, vol. 24, no. 1, pp. 30–42, 2009.

[46] P. Pillay and R. Krishnan, “Modeling, simulation, and analysis of permanent-magnet motor drives. ii. the brushless dc motor drive,” *Industry Applications, IEEE Transactions on*, vol. 25, no. 2, pp. 274–279, 1989.

[47] Y.-S. Lai, F.-S. Shyu, and Y.-H. Chang, “Novel loss reduction pulsewidth modulation technique for brushless dc motor drives fed by mosfet inverter,” *Power Electronics, IEEE Transactions on*, vol. 19, no. 6, pp. 1646–1652, 2004.

[48] M. Tomita, T. Senjyu, S. Doki, and S. Okuma, “New sensorless control for brushless dc motors using disturbance observers and adaptive velocity estimations,” *Industrial Electronics, IEEE Transactions on*, vol. 45, no. 2, pp. 274–282, 1998.

[49] J.-W. Choi and S.-C. Lee, "Antiwindup strategy for pi-type speed controller," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 6, pp. 2039–2046, 2009.

[50] T. Ohmae, T. Matsuda, K. Kamiyama, and M. Tachikawa, "A microprocessor-controlled high-accuracy wide-range speed regulator for motor drives," *IEEE Transactions on Industrial Electronics*, vol. 3, no. IE-29, pp. 207–211, 1982.

[51] M. Baszynski and S. Pirog, "The high speed drive control system for the low power induction machine," *Przegląd Elektrotechniczny*, vol. 87, no. 2, pp. 151–156, 2011.

[52] C.-W. Hung, J.-H. Chen, and K.-C. Huang, "A correction circuit of hall-sensor-signal-based speed measurement for bldc motors," *Artificial Life and Robotics*, vol. 17, no. 1, pp. 80–85, 2012.

[53] A. Tilli and M. Montanari, "A low-noise estimator of angular speed and acceleration from shaft encoder measurement," *Automatika*, vol. 42, no. 3/4, pp. 169–176, 2001.

[54] A. Yoo, S.-K. Sul, D.-C. Lee, and C.-S. Jun, "Novel speed and rotor position estimation strategy using a dual observer for low-resolution position sensors," *Power Electronics, IEEE Transactions on*, vol. 24, no. 12, pp. 2897–2906, 2009.

[55] G. P. Tournier, M. Valenti, J. P. How, and E. Feron, "Estimation and control of a quadrotor vehicle using monocular vision and moire patterns," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, pp. 21–24, 2006.

[56] G. B. Kim, T. K. Nguyen, A. Budiyono, J. K. Park, K. J. Yoon, and J. Shin, "Design and development of a class of rotorcraft-based uav," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[57] H. Bolandi, M. Rezaei, R. Mohsenipour, H. Nemati, and S. M. Smailzadeh, "Attitude control of a quadrotor with optimized pid controller," 2013.

[58] Y. Bai, H. Liu, Z. Shi, and Y. Zhong, "Robust control of quadrotor unmanned air vehicles," in *Control Conference (CCC), 2012 31st Chinese*, pp. 4462–4467, IEEE, 2012.

[59] A. Das, F. Lewis, and K. Subbarao, "Backstepping approach for controlling a quadrotor using lagrange form dynamics," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1-2, pp. 127–151, 2009.

[60] D. Lee, C. Ha, and Z. Zuo, "Backstepping control of quadrotor-type uavs and its application to teleoperation over the internet," in *Intelligent Autonomous Systems 12*, pp. 217–225, Springer, 2013.

[61] T. Taniguchi, L. Eciolaza, and M. Sugeno, "Quadrotor control using dynamic feedback linearization based on piecewise bilinear models," in *Computational Intelligence in Control and Automation (CICA), 2014 IEEE Symposium on*, pp. 1–7, IEEE, 2014.

[62] O. Fritsch, P. De Monte, M. Buhl, and B. Lohmann, "Quasi-static feedback linearization for the translational dynamics of a quadrotor helicopter," in *American Control Conference (ACC), 2012*, pp. 125–130, IEEE, 2012.

[63] M. D. Hua, *Contributions to the automatic control of aerial vehicles*. PhD thesis, Université Nice Sophia Antipolis, 2009.

[64] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2451–2456, IEEE, 2004.

[65] A. Ataka, H. Tnunay, R. Inovan, M. Abdurrohman, H. Preastianto, A. I. Cahyadi, and Y. Yamamoto, "Controllability and observability analysis of the gain scheduling based linearization for uav quadrotor," in *Robotics, Biomimetics, and Intelligent Computational Systems (ROBIONETICS), 2013 IEEE International Conference on*, pp. 212–218, IEEE, 2013.

[66] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An integral predictive/nonlinear h∞ control structure for a quadrotor helicopter," *Automatica*, vol. 46, no. 1, pp. 29–39, 2010.

[67] M.-D. Hua, T. Hamel, P. Morin, and C. Samson, "A control approach for thrust-propelled underactuated vehicles and its application to vtol drones," *Automatic Control, IEEE Transactions on*, vol. 54, no. 8, pp. 1837–1853, 2009.

[68] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, pp. 1–14, 2008.

[69] A. Radke and Z. Gao, "A survey of state and disturbance observers for practitioners," in *American Control Conference, 2006*, pp. 6–pp, IEEE, 2006.

[70] Q. Zheng, L. Dong, D. H. Lee, and Z. Gao, "Active disturbance rejection control for mems gyroscopes," in *American Control Conference, 2008*, pp. 4425–4430, IEEE, 2008.

[71] H. Xiong, F.-s. Jing, J.-q. Yi, and G.-l. Fan, "Automatic takeoff of unmanned aerial vehicle based on active disturbance rejection control," in *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pp. 2474–2479, IEEE, 2009.

[72] H. Xiong, J.-q. Yi, G.-l. Fan, F.-s. Jing, and R.-y. Yuan, “Autolanding of unmanned aerial vehicles based on active disturbance rejection control,” in *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, vol. 2, pp. 772–776, IEEE, 2009.

[73] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, “Visually guided landing of an unmanned aerial vehicle,” *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 3, pp. 371–380, 2003.

[74] S. Hrabar and G. S. Sukhatme, “Omnidirectional vision for an autonomous helicopter,” in *Robotics and Automation, 2003. Proceedings. ICRA’03. IEEE International Conference on*, vol. 1, pp. 558–563, IEEE, 2003.

[75] S. Saripalli, G. S. Sukhatme, L. O. Mejías, and P. C. Cervera, “Detection and tracking of external features in an urban environment using an autonomous helicopter,” in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 3972–3977, IEEE, 2005.

[76] L. Mejias, S. Saripalli, P. Campoy, and G. S. Sukhatme, “Visual servoing of an autonomous helicopter in urban areas using feature tracking,” *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 185–199, 2006.

[77] Z. Yu, K. Nonami, J. Shin, and D. Celestino, “3d vision based landing control of a small scale autonomous helicopter,” *International Journal of Advanced Robotic Systems*, vol. 4, no. 1, pp. 51–56, 2007.

[78] R. Lozano, *Unmanned aerial vehicles: Embedded control.* John Wiley & Sons, 2013.

[79] P. N. Smith, B. Sridhar, and B. Hussien, “Vision-based range estimation using helicopter flight data,” in *Computer Vision and Pattern Recognition, 1992. Pro-*

ceedings CVPR'92., 1992 IEEE Computer Society Conference on, pp. 202–208, IEEE, 1992.

[80] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," in *Robotics and automation (ICRA), 2011 IEEE international conference on*, pp. 2992–2997, IEEE, 2011.

[81] B. Herisse, F.-X. Russotto, T. Hamel, and R. Mahony, "Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 801–806, IEEE, 2008.

[82] N. Krouglicof, "Rigid-body pose measurement from a single perspective view," in *Intelligent Autonomous Systems, IAS–3: An International Conference, Pittsburgh, Pennsylvania, February 15-18, 1993*, p. 368, IOS Press, 1993.

[83] S. Yang, S. A. Scherer, and A. Zell, "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 499–515, 2013.

[84] S. Lange, N. Sünderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor uav in gps-denied environments," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pp. 1–6, IEEE, 2009.

[85] L. D. Minh and C. Ha, "Modeling and control of quadrotor MAV using vision-based measurement," in *Strategic Technology (IFOST), 2010 International Forum on*, pp. 70–75, IEEE, 2010.

[86] F. Kendoul, K. Nonami, I. Fantoni, and R. Lozano, “An adaptive vision-based autopilot for mini flying machines guidance, navigation and control,” *Autonomous Robots*, vol. 27, no. 3, pp. 165–188, 2009.

[87] S. Saripalli, “Vision-based autonomous landing of a helicopter on a moving target,” in *Proceedings of AIAA Guidance, Navigation, and Control Conference, Chicago, USA*, 2009.

[88] K. E. Wenzel, A. Masselli, and A. Zell, “Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle,” *Journal of intelligent & robotic systems*, vol. 61, no. 1-4, pp. 221–238, 2011.

[89] J. Chen and A. Pinz, *Structure and motion by fusion of inertial and vision-based tracking.* na, 2004.

[90] S. G. Chroust and M. Vincze, “Fusion of vision and inertial data for motion and structure estimation,” *Journal of Robotic Systems*, vol. 21, no. 2, pp. 73–83, 2004.

[91] G. Chatterji, P. Menon, and B. Sridhar, “Gps/machine vision navigation system for aircraft,” *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 33, no. 3, pp. 1012–1025, 1997.

[92] J. Wang, M. Garratt, A. Lambert, J. J. Wang, S. Han, and D. Sinclair, “Integration of gps/ins/vision sensors to navigate unmanned aerial vehicles,” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, pp. 963–970, 2008.

[93] D. Hubbard, B. Morse, C. Theodore, M. Tischler, and T. McLain, “Performance evaluation of vision-based navigation and landing on a rotorcraft unmanned

aerial vehicle," in *Applications of Computer Vision, 2007. WACV'07. IEEE Workshop on*, pp. 5–6, IEEE, 2007.

[94] P. Bertin, D. Roncin, and J. Vuillemin, "Introduction to programmable active memories," 1989.

[95] J. U. Cho, S. H. Jin, X. D. Pham, J. W. Jeon, J. E. Byun, and H. Kang, "A real-time object tracking system using a particle filter," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 2822–2827, IEEE, 2006.

[96] D. K. Masrani and W. J. MacLean, "A real-time large disparity range stereo-system using fpgas," in *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*, pp. 13–13, IEEE, 2006.

[97] J. Woodfill and B. Von Herzen, "Real-time stereo vision on the parts reconfigurable computer," in *Field-Programmable Custom Computing Machines, 1997. Proceedings., The 5th Annual IEEE Symposium on*, pp. 201–210, IEEE, 1997.

[98] Y. Jia, X. Zhang, M. Li, and L. An, "A miniature stereo vision machine (msvm-iii) for dense disparity mapping," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 1, pp. 728–731, IEEE, 2004.

[99] O. Amidi, *An autonomous vision-guided helicopter.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1996.

[100] Y. Mulgaonkar and V. Kumar, "Towards open-source, printable pico-quadrotors,"

[101] D. Simonis, *Inventors and inventions.* Marshall Cavendish, 2007.

[102] T. Wilson and P. Trickey, “Dc machine with solid-state commutation,” *Electrical Engineering*, vol. 81, no. 11, pp. 879–884, 1962.

[103] APC Prop. URL: `http://www.apcprop.com/v/html/rpm_limits.html`.

[104] F. Handbook, “8083-21,” *Rotorcraft Flying Handbook*, 2000.

[105] Swiss Composites. URL: `http://www.swiss-composite.ch`.

[106] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, “Vision based mav navigation in unknown and unstructured environments,” in *Robotics and automation (ICRA), 2010 IEEE international conference on*, pp. 21–28, IEEE, 2010.

[107] J. Engel, J. Sturm, and D. Cremers, “Camera-based navigation of a low-cost quadrocopter,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2815–2821, IEEE, 2012.

[108] Allegro Micro-Systems. URL: `http://www.allegromicro.com/`.

[109] Arduino CC, “Arduino Pro mini specification, datasheet, schematic and PCB design board.” URL: `http://www.arduino.cc/en/Main/ArduinoBoardProMini`.

[110] Vicon Motion Systems, Inc. URL: `http://www.vicon.com`.

[111] T. P. Crompton, *Battery reference book*. Newnes, 2000.

[112] J.-M. Tarascon and M. Armand, “Issues and challenges facing rechargeable lithium batteries,” *Nature*, vol. 414, no. 6861, pp. 359–367, 2001.

[113] D. Karnopp and R. C. Rosenberg, “Analysis and simulation of multiport systems,” 1968.

[114] W. Borutzky, "Multibody systems," *Bond Graph Methodology: Development and Analysis of Multidisciplinary Dynamic System Models*, pp. 353–389, 2010.

[115] A. M. Bos, "Modelling multibody systems in terms of multibond graphs: With application to a motorcycle," 1986.

[116] A. Zeid and C.-H. Chung, "Bond graph modeling of multibody systems: a library of three-dimensional joints," *Journal of the Franklin Institute*, vol. 329, no. 4, pp. 605–636, 1992.

[117] R. W. Prouty, *Helicopter performance, stability, and control.* 1995.

[118] G. D. Padfield, *Helicopter flight dynamics.* John Wiley & Sons, 2008.

[119] C. Ice and B. Akin, "Designing high-performance and power-efficient motor control systems," *White paper, lit. num. SPRT528, Texas Instruments*, 2009.

[120] R. Carlson, M. Lajoie-Mazenc, and J. C. d. S. Fagundes, "Analysis of torque ripple due to phase commutation in brushless dc machines," *Industry Applications, IEEE Transactions on*, vol. 28, no. 3, pp. 632–638, 1992.

[121] A. Basak, *Permanent-magnet DC linear motors*, vol. 40. Oxford University Press, 1996.

[122] W. Hart Daniel, "Introduction to power electronics," 1997.

[123] Y. Jeon, H. Mok, G. Choe, D. Kim, and J. Ryu, "A new simulation model of bldc motor with real back emf waveform," in *Computers in Power Electronics, 2000. COMPEL 2000. The 7th Workshop on*, pp. 217–220, IEEE, 2000.

[124] S. Baldursson, "Bldc motor modelling and control-a matlab®/simulink® implementation," 2005.

[125] J. Chen and P.-C. Tang, “A sliding mode current control scheme for pwm brushless dc motor drives,” *Power Electronics, IEEE Transactions on*, vol. 14, no. 3, pp. 541–551, 1999.

[126] G. Pfaff, A. Weschta, and A. F. Wick, “Design and experimental results of a brushless ac servo drive,” *Industry Applications, IEEE Transactions on*, no. 4, pp. 814–821, 1984.

[127] Y. Wu, Z. Deng, X. Wang, X. Ling, and X. Cao, “Position sensorless control based on coordinate transformation for brushless dc motor drives,” *Power Electronics, IEEE Transactions on*, vol. 25, no. 9, pp. 2365–2371, 2010.

[128] R. Petrella, M. Tursini, L. Peretti, and M. Zigliotto, “Speed measurement algorithms for low-resolution incremental encoder equipped drives: a comparative analysis,” in *Electrical Machines and Power Electronics, 2007. ACEMP’07. International Aegean Conference on*, pp. 780–787, IEEE, 2007.

[129] Analog Devices, “Speed estimation and control using the admc401 encoder interface unit (eiu).”

[130] R. Isermann, *Digital control systems*. Springer Science & Business Media, 2013.

[131] W. Zhao, B. H. Kim, A. C. Larson, and R. M. Voyles, “Fpga implementation of closed-loop control system for small-scale robot,” in *Advanced Robotics, 2005. ICAR’05. Proceedings., 12th International Conference on*, pp. 70–77, IEEE, 2005.

[132] D. Bishop, “Fixed point package user’s guide,” *Packages and bodies for the IEEE*, pp. 1076–2008, 2006.

[133] CTC, "Measuring Motor Parameters." URL: `http://support.ctc-control.com/customer/elearning/younkin/motorParameters.pdf`, 2013.

[134] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital control of dynamic systems*, vol. 3. Addison-wesley Menlo Park, 1998.

[135] A. Kongmunvattana and P. Chongstivatana, "A fpga-based behavioral control system for a mobile robot," in *Circuits and Systems, 1998. IEEE APCCAS 1998. The 1998 IEEE Asia-Pacific Conference on*, pp. 759–762, IEEE, 1998.

[136] G. De Michell and R. K. Gupta, "Hardware/software co-design," *Proceedings of the IEEE*, vol. 85, no. 3, pp. 349–365, 1997.

[137] H. L. Wade, *Basic and Advanced Regulatory Control: System Design and Application.* ISA, 2004.

[138] A. Visioli, *Practical PID control.* Springer Science & Business Media, 2006.

[139] R. Czyba and G. Szafrański, "Control structure impact on the flying performance of the multi-rotor vtol platform-design, analysis and experimental validation," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[140] Y. X. Su, B. Y. Duan, C. H. Zheng, Y. Zhang, G. Chen, and J. Mi, "Disturbance-rejection high-precision motion control of a stewart platform," *Control Systems Technology, IEEE Transactions on*, vol. 12, no. 3, pp. 364–374, 2004.

[141] V. Sklyarov, "Reconfigurable models of finite state machines and their implementation in fpgas," *Journal of Systems Architecture*, vol. 47, no. 14, pp. 1043–1064, 2002.

[142] A. K. Oudjida, M. L. Berrandjia, R. Tiar, A. Liacha, and K. Tahraoui, "Fpga implementation of i 2 c & spi protocols: A comparative study," in *Electronics,*

Circuits, and Systems, 2009. ICECS 2009. 16th IEEE International Conference on, pp. 507–510, IEEE, 2009.

[143] P. P. Chu, *FPGA prototyping by Verilog examples: Xilinx Spartan-3 version.* John Wiley & Sons, 2011.

[144] Z. Ping and Z. Gao, "An fpga-based digital control and communication module for space power management and distribution systems," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 4941–4946, IEEE, 2005.

[145] Z. Gao, "Scaling and bandwidth-parameterization based controller tuning," in *Proceedings of the American Control Conference*, vol. 6, pp. 4989–4996, 2006.

[146] R. Miklosovic, A. Radke, and Z. Gao, "Discrete implementation and generalization of the extended state observer," in *American Control Conference, 2006*, pp. 6–pp, IEEE, 2006.

[147] C. Xing, L. Donghai, G. Zhiqiang, and W. Chuanfeng, "Tuning method for second-order active disturbance rejection control," in *Control Conference (CCC), 2011 30th Chinese*, pp. 6322–6327, IEEE, 2011.

[148] Z. Qing and G. Zhiqiang, "On practical applications of active disturbance rejection control," in *Control Conference (CCC), 2010 29th Chinese*, pp. 6095–6100, IEEE, 2010.

[149] S. Bouabdallah, *Design and control of quadrotors with application to autonomous flying.* PhD thesis, École Polytechnique federale de Lausanne, 2007.

[150] A. Rosenfeld and A. C. Kak, *Digital Picture Processing: Vol.: 1.* Academic Press, Incorporated, 1982.

[151] M.-K. Hu, "Visual pattern recognition by moment invariants," *Information Theory, IRE Transactions on*, vol. 8, no. 2, pp. 179–187, 1962.

[152] S. Azrad, F. Kendoul, and K. Nonami, "Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm," *Journal of System Design and Dynamics*, vol. 4, no. 2, pp. 255–268, 2010.

[153] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a vtol uav on a moving platform using image-based visual servoing," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 971–976, IEEE, 2012.

[154] E. Kreyszig, *Advanced Engineering Mathematics.* Wiley, 1962.

[155] W. Philips, "A new fast algorithm for moment computation," *Pattern Recognition*, vol. 26, no. 11, pp. 1619–1621, 1993.

[156] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, pp. 15–16, 2006.

[157] S. L. Waslander and C. Wang, "Wind disturbance estimation and rejection for quadrotor position control," in *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference, Seattle, WA*, 2009.

[158] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza, "A monocular pose estimation system based on infrared leds," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 907–913, IEEE, 2014.

[159] J.-E. Gomez-Balderas, G. Flores, L. G. Carrillo, and R. Lozano, "Tracking a ground moving target with a quadrotor using switching control," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 65–78, 2013.