DESIGN, DEVELOPMENT AND APPLICATIONS OF A FRAMEWORK FOR AUTONOMOUS VEHICLE OPERATIONS

by

Martin Diz
June, 2015

A dissertation submitted to the
Faculty of the Graduate School of
The University at Buffalo, State University of New York
in partial fulfillment of the requirements for the
degree of

Doctor of Philosophy

Department of Mechanical and Aerospace Engineering

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

DEDICATION

To my family, and everyone, that one way or another made this fun ride possible.

In loving memory of Marina Diaz, Matias Roldan, Segundo Diz and Solène Pouget

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to his advisor, Dr. Manoranjan Majji, whose encouragement, tutelage, motivation, guidance and support made this dissertation possible. Having taken constant inspiration from his advisor, the author is highly indebted to Dr. Majji. It is indeed a great honor to have shared some subspace of research activity with him and the author cannot find words to express his inordinate appreciation.

Dr. Puneet Singla has not been only a major influence on the author, providing him great advice, support, guidance and insight on several joint research topics, but also encouraged the author to apply to UB.

The author would like to express his gratitude to Dr. Juang for all the motivation, guidance, interaction, insight, training and collaboration he continues to provide the author, with exceptional humility.

The author further extends his thanks to Dr. John Crassidis for teaching him various elements of Dynamical Systems and System Estimation in three different classes.

The author would like to express his gratitude towards Dr. Marcus Bursik for his encouragement during the toughest moments.

Thanks to the past and present residents of AR Laboratories who make it a

wonderful place for research with a lot of interactions and activities that kept everyone going days and nights throughout the years.

The author is in an eternal gratitude with the State University of New York, University at Buffalo for its continuous, uninterrupted and astounding support.

This dissertation was funded partially by the National Aeronautics and Space Administration (NASA) under grant no. NNX12AM12G. The opinions and findings presented in the dissertation are of the author. They do not reflect the views of NASA.

The author of this dissertation would like to express his gratitude towards Advance Motion Controls (AMC) for the contribution of 4 motor drivers and a mounting board towards the development of the prototype Canfield joint.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The main objective of the current dissertation is to develop a "Plug and Play" autopilot. We present a systematic approach to decouple controller and filter design from hardware selection. This approach also addresses the performance decay due to hardware limitations. The outcome of this work is an integrated environment to develop, validate, and test algorithms to be used on flight controllers.

The dynamic model of an off-the-shelf radio controlled airplane is derived. A controller is developed for the aircraft and it is implemented within the proposed framework. The framework and the controller developed are validated using hardware-in-the-loop simulation. A physical experiment with an autonomous ground vehicle is performed to test the autopilot and algorithms before test flights are performed. Finally a test flight is performed using a Great Planes Funster.

A novel microsatellite attitude controller is presented. This controller is also developed and implemented using the approach presented in this dissertation. A satellite attitude hybrid simulator is presented and its design is discussed. An experimental apparatus to test the controller in a microgravity environment is constructed and discussed.

Finally a set of experiments that demonstrate how the framework can be integrated into commercially available autopilots and vehicles is presented. First

an off-the-shelf quad-rotor that integrates a look-down camera is used to perform visual navigation and landing. Second, a rover and a quad-rotor are used on a cooperative schema. The rover follows a route and the quad-rotor escorts it. The third experiment presents a popular autopilot on a surveillance mission. For this mission the airplane is equipped not only with the autopilot and radio link, but also with a video system. The video system consists of a camera and a radio link. These experimental results demonstrate the utility of the proposed framework in enhancing the capabilities of off-the-shelf autopilots and vehicles while simultaneously simplifying mission preparation and execution.

Chapter 01

INTRODUCTION

Wayfarer, there is no way.

Make your way as you go.

Antonio Machado

Control theory provides the methods required to achieve the desired performance of a dynamical system by utilizing mathematical models to facilitate an appropriate choice of actions. Dynamical system typically refer to a set of differential or equations modes for systems, such as vehicles, circuits, robotic arms, chemical reactors among others. The origins of automatic control can be traced back to James Watt's centrifugal governor for the speed control of a steam motor during the eighteenth century.[1]

Until relatively recently in human history, any change in the performance requirements for a mechanical or electrical device resulted in an entire system redesign and construction. The same is true for electrical system design and realizations. To overcome this limitation and give flexibility to system design, Bush introduced analog computers in the 1930's.[2] Hazen introduced the concept of servo-mechanism[3]

in 1934 which opened a whole new world of possibilities for the control of mechanical and electrical systems. Introduction of Bush's new differentiator computer[4] in 1936 ushered in an era of computational controls in dynamical systems as we now know it.[5] Analog computers allowed designers to achieve a variety of performance indices without the need of modifying the plant. However, these controllers were designed ad-hoc and any change in the design mandated a new controller to be built.[6] By 1960, hybrid analog-digital computers were introduced[7] giving birth to modern control techniques and the foundations of these platforms are still used today.

Hybrid analog-digital techniques and computers to control dynamical systems are now commonly referred to as digital control. Digital computers are used to implement controllers that achieve desired performance by utilizing complex mathematical models. However, the interface with actuators and sensors is performed with the use of analog circuits, hence the hybrid analog-digital design.

The control problem can usually be divided in three layers: Physical, Analog, and Digital, as shown in Figure 1.1. Sensors, Actuators and the plant define the physical layer. Different kinds of actuators such as linear servos, and motors control the plant. Strategically placed sensors then measure physical quantities such as pressure, temperature, or velocity. At low level these sensors are are transducers that translate the physical quantity into electrical signals; usually a voltage or current. Transducers, also provide the inverse conversion; converting electrical signals to physical excitation, a property useful for actuation. Electrical signals define the analog layer where prefilters are used to process analogical signals to reduce noise.

The analog layer acts as an intermediate link between the digital layer, where the controller is implemented, and the mathematical model of the plant exists. Digital to Analog Converters (DAC) and Analog to Digital Converters (ADC) are used to interface between the digital and analog layers.[8] The use of digital systems allows the designer to change control laws without the need of modifying the plant or data acquisition layers. With these new methods, performance can be updated without the need to redesign the control system. An example of a design where the three layers are clearly identified is the Apollo flight computer.[9] Although the computational power, in digital hardware was limited, a proper design allowed the engineers to achieve the desired performance of the famous space mission. This template continues to serve as reference model for autopilots to these days.

When implementing controllers on a physical system one of the major challenges is the selection of appropriate hardware. The list of hardware specifications to consider includes memory capacity for storing programs; memory volatility for computations; data acquisition layer architecture, which is dictated by the set of inputs and outputs; and computational capacity. In addition to hardware considerations, software compatibility must be carefully observed. Several kinds of programming languages exist, such as ADA, C, C++ are used to implement flight controllers. Each of them have they characteristic features. The designer needs to select a platform that can solve the required algorithms in the allocated time with the supported tools such as language, compilers, Application Program Interface (API). The interaction with the plant is limited to the set of inputs and outputs of the computer.

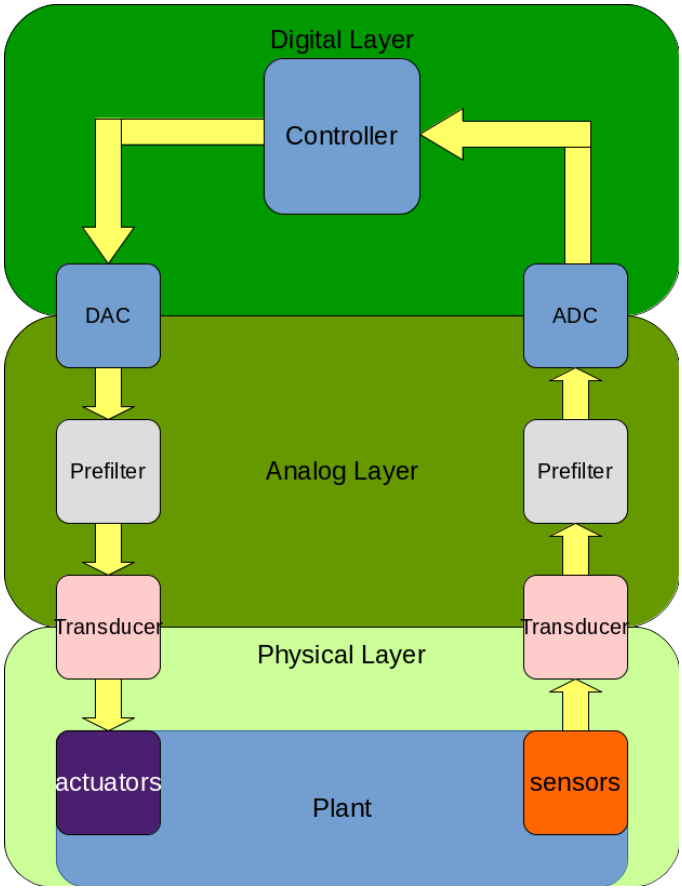


Figure 1.1: Layers involved in plant controller

The selected device for implementation must satisfy all the design constraints while offering reliability and stability. The equipment should perform in a variety of operating conditions, since the vehicle can be subject to different external perturbations such as vibrations, high temperatures, dust and humidity; all of which affect the microelectronics and controller performance. When developing flight controllers for Unmanned Air Vehicles (UAV) the search for hardware is a special challenge owing to the size and weight restrictions dictated by vehicle capabilities.

The proliferation of inexpensive microelectronics and availability of open soft-

ware has lead to the development of several autopilots such as Paparazzi,[10] Arduplane[11] among others.[12] These off-the-shelf solution have gained popularity due to their low cost and ease of use. However, these platforms suffer from limitations that sometimes limit their applicability for a problem of interest. The principal constraint is given by the tight-coupling that exists between the software and hardware; most of the code used is an ad-hoc design for a specific assembly of hardware. Software upgrades are limited by the extremely limited computational power of these embedded platforms. Since the code is developed ad-hoc, the reusability and portability of the software elements is extremely limited. The embedded electronics used in the popular solutions provided by Paparazzi and Arduplane have a limited set of inputs and outputs, which poses a challenge when trying to integrate new sensors or actuators; or in modifying Paparazzi system for more advance applications.

This dissertation presents a novel approach to address these issues and limitation in a unified way. A robust, portable, scalable multi-hardware, multi-platform autopilot has been developed (smartAI). The ability to easily integrate new controllers and estimators based on complex logic and algorithms are the main features of the autopilot developed in this research. The novel aspect of the design is the separation on the digital layer between the Hardware Abstraction Layer (HAL) and the control algorithms as shown in Figure 1.2. The HAL performs all the tasks related to the data acquisition. With this separation replacing the hardware requires only to add specific modules to the HAL; leaving the control algorithms unchanged. In Table 1.1 we present a comparison between the autopilot designed

in this dissertation and commercially available autopilots. For this comparison a Beagle Bone Black is considered as implementation hardware.

Autopilot	software board	Paparazzi[10] Lisa/L	Arduplane[11] apm2.6	Micropilot[13] MP2128	SmartAI Beagle Bone Black
physical	Size [mm] Weight [g]	90x50x10 30	70x45x12 28	100x40x15 24	86x53x20 37
CPU	Power [w] Clock	2 72 [MHz]	1 16 [MHz]	1.24 NA	1.5 1 [GHz]
sensors	barometer IMU (6 DOF) Magnetometer airspeed temperature GPS	yes ext ext yes yes ext	yes yes 3DOF ext yes ext	yes yes compass yes yes yes	ext ext ext ext ext ext
Interface	IN OUT	1 6	8 8	8 8(x3)	 8
Expansion	BUS Devices Misc	SP(3), I2C(2), SPI(2), USB(1) AI(2) CPU	SP(3), I2C, SPI, AI/DO(8) NA	SP(8) DI/DO/PWM (8) board	SP(4.5), I2C(2), SPI(2),USB(1) DIO(65), PWM(8), AI(7) ETH, HDMI, CAPE
Control	Loops [Hz] PID Plugin HW	100,50 PID (8) module 512KB, 64KB (RAM)	100 PID (6) NA 256KB, 8KB (RAM)	30,5,180 PID (12) GS (5) init, event (sys, pid) 64KB, 64KB (RAM)	1000x6 module,freq, event,init 4GB, 512MB (RAM)
Filter	AHRS INS	Complementary, DCM Altitude	DCM complementary	Kalman 12 states	Kalman[14], EXT smoother, EXT
Misc	Communication Log	ad-hoc none	MAVLink subset 16 Mb (4M),GPS, IMU,Loops	proprietary 1.5 Mb (0.3M) 47F	MAVLink 4GB, SD, Signals, FDR

Table 1.1: Autopilot comparison

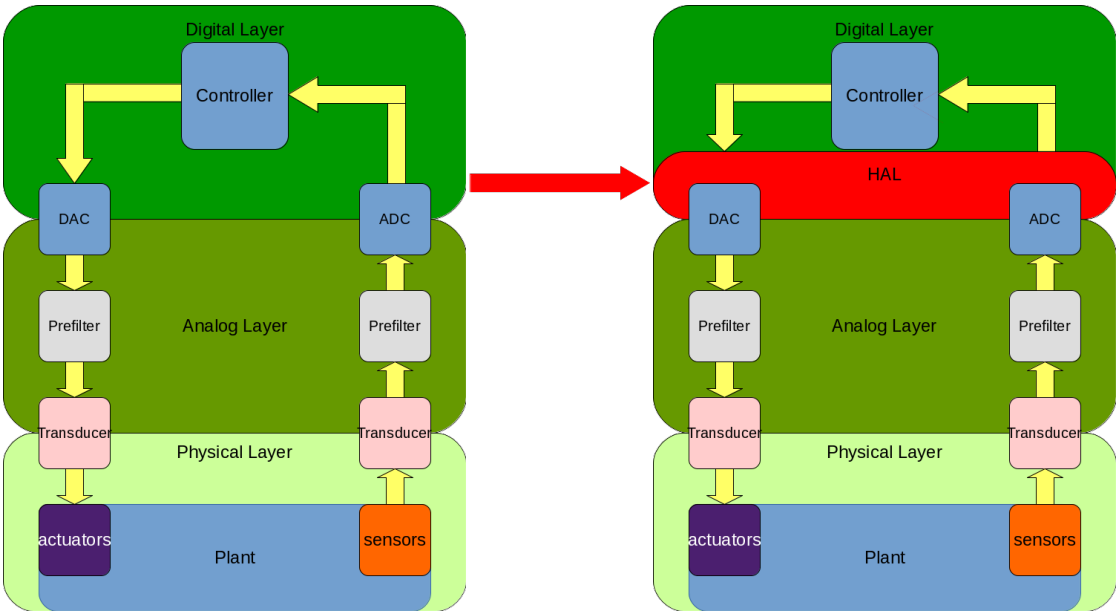


Figure 1.2: Proposed digital layer design

Autonomous vehicles cannot operate without a Ground Control Station (GCS). The GCS is required for three main reasons: First, it acts as an interface for the operator. Although the autopilot should safely operate autonomously, user supervision is vital to handle unmatched perturbations such as, mechanical or electrical failure. The second use for the GCS is to provide extra computational power to aid the autopilot in solving complex algorithms, such as those required by path planners. The third use is to complement the mission; for example, if an elevation map is being generated from a video stream of the landscape the required image processing techniques can be delegated to the GCS.

Ground Control Stations are being developed by research institutions[15–17] and several industrial entities.[18–20] Open source alternatives are also available.[21;22] Most

GCS software such as the one presented by Jun et. al.[23] merely relay information from the vehicle to the operator, but in no way provides the capability to augment or enhance vehicle performance. Others approaches, like the ones developed by the USAF[16;17] have the main objective of providing an immersive visualization environment for the remote operator. Other simpler alternatives like the open source GCS packages[21;22] forward waypoints and receive GPS data from an UAV. None of the mentioned GCS implementations accommodate the multifunctional and reconfigurable autopilot we propose; for this reason we introduce SAI-GCS.

In our design, the GCS is intended to enhance and extend the vehicle capabilities in such a fashion that the vehicle-GCS integration results in a system fully capable of accomplishing a variety of missions allowing variations levels of intervention from an external operator. Our system design allows autonomous and semi-autonomous modes of operation by accommodating flight operations by pilots with a wide variety of skill-sets. The key difference between our GCS and those previously presented is that the user intervention in our design is non-mandatory.

We base our system design on a version of the ground control station originally developed by NASA as a remote controller for the HiMAT platforms.[24] Although this design is obsolete the guidelines and technical framework has led to an effective GCS implementation, notably its excellent capability to accommodate system failures.[25] A complex system such as the GCS, requires an efficient and reliable way to test and validate each of the components. Dryden Research Center has developed a methodology just to validate these designs.[26]

Shim and Kim,[27] proposed a system to control an Unmanned Aerial Vehicle

(UAV) developed by the University of California at Berkeley, which shares much of the capabilities proposed in the current design. However, the system requires an autonomous vehicle to properly work. The minimum requirement to consider the vehicle autonomous is that it should be able to track waypoints received over a radio modem. On the other hand, for our design we relax the the requirement for independent waypoint tracking; this allows the system to be used with vehicles that lack this capability, such as the popular ArDone Parrot. Many inexpensive vehicles are remotely commanded with the use of a remote signal; although they cannot autonomously track position, they can be used for several types of missions if waypoint tracking can be implemented on the GCS.

The main goals of the of the SAI-GCS is to provide a simple, yet flexible configuration that can be scaled to match a variety of standard missions.This is made possible by utilizing state of the art software engineering techniques such as modular design. Alexander[28] defined patterns as "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solutions a million times over, without ever doing it the same way twice". Its clear that to increase reusability in SAI-GCS the use of patterns provides a simpler solution as opposed to conventional programming practices. The use of patterns in Object Oriented Programing[29] is not new to aeronautics, its already being used in flight simulators[30] with great outcomes.

We now discuss the contributions, scope, and limitations of this dissertation.

1.1 THESIS SCOPE

The main objective of the current dissertation is to provide a new abstraction layer between the data acquisition layer and the implemented algorithms. We present a systematic approach to decouple controller and filter design from hardware selection. This approach also, addresses the performance decay due to hardware limitations. The outcome of this work is an integrated environment to develop, validate and test algorithms where there is a direct match between the physical model and the software model. Such a system enables the development and implementation of model based control systems.

The issues to overcome when implementing flight controllers are numerous and by no means the present work addresses all of them comprehensively. However, the critical problems required for a safe flight have been addressed, and the non-critical ones have been taken into account and a mechanism has been set in place to develop solutions as future work. The next section presents the outline

1.2 THESIS OUTLINE

The organization of this dissertation is as follows: The remainder of chapter I is left to introduce the system layout. The software design is presented first; then advantages, and disadvantages, and limitations are discussed. Chapter II introduces a novel approach to deal with hardware constraints on controllers. Implementations are discussed. The mathematical models of the RC aircraft used in the thesis is presented. A controller is developed and numerical examples are given. Chapter III

presents the simulation capabilities of the proposed framework. The requirements for a hardware-in-the-loop (HIL) simulation and a set of simulations are presented. In Chapter IV a physical experiment with an autonomous ground vehicle is performed to test the autopilot and algorithms before test flights are performed. A test flight is performed using a Great Planes Funster RC plane, retrofitted with an automatic flight control system developed in this dissertation. Chapter V covers the design of an experimental setup and its integration into a spaceship for a microgravity experiment using a version of the autopilot system. Chapter VI presents several other uses and how the GCS can complement off-the-shelf autopilots and vehicles. Conclusion and future directions are presented in Chapter VII.

1.3 UNIFIED MODELING LANGUAGE

Before proceeding with the system description, we give a summary of the conventions used for the class diagrams and define some common terms. For a more comprehensive discussion of the Unified Modeling Language (UML) the reader can refer to Fowler's UML guide.[31] For a complete description of the object oriented and patterns design please refer to Gamma.[29]

A Class is represented with a rectangle with two horizontal lines giving three place holders. The top section is used for the class name. The middle section is reserved for the Class properties which are, the internal parameters. The bottom section is reserved for the class methods also referred as functions. In Figure 1.3 we show an example.

An *interface* is a description of the actions an object can do. In Object-Oriented

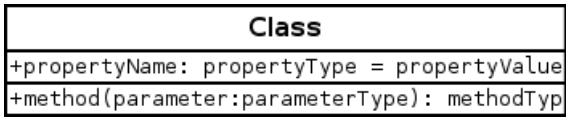


Figure 1.3: UML Class representation

Programming (OOP), an interface is a description of all the functions a Class must have to be of a certain type such as Filter, Controller, etc. More specifically an interface is a programming structure/syntax that allows the computer to enforce certain properties on a class. For example, we have a KalmanFilter class and a ParticleFilter class. Each of these two classes should have an "update" action. How the filter is updated is left to each particular class, but the fact that they must have a update method is a function of the interface. The interface description is an abstract class and it is not intended to be used. On the other hand, the classes to be used are the ones inheriting from the parent class that must be concrete classes.

In OOP, inheritance enables new objects to take on the properties of existing objects. A class that is used as the basis for inheritance is called a parent class. A class that inherits from a parent class is called a child class. A child inherits visible properties and methods from its parent while adding additional properties and methods of its own. This way, the child extend the capabilities of the parent. For instance a KalmanFilter will inherit from a parent class Filter. The inheritance diagram is shown in Figure 1.4, were a empty triangle indicates the parent and a line connects to the child. If the child is an abstract class, i.e another interface, the line is solid. If the child is a concrete class the line is dashed.

There are two terms that usually generate confusion when dealing with Object Oriented Design (OOD): *composition* and *aggregation*. These terms refers to the

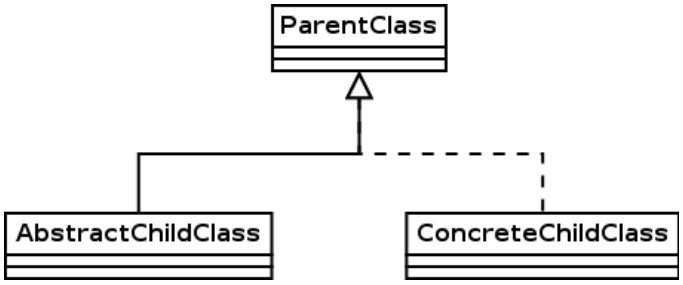


Figure 1.4: UML Class inheritance representation

objects an object has. For example, the Control class has a set of controllers. The only difference between composition and aggregation is the ownership of the object. For a composition the object composed with other objects own these objects. Control owns the controllers, when the control class is destroyed also all controllers are destroyed. On the other hand, on aggregation the object does not own the objects that it has. When the parent class is destroyed all children classes will remain alive. For example, all objects are connected using Signals, however when one of these objects is destroyed the signals will remain active. Composition is shown in Figure 1.5 which its indicated with a diamond in the parent and a arrow on the child, sometimes a number is added to indicate how many instances are owned.

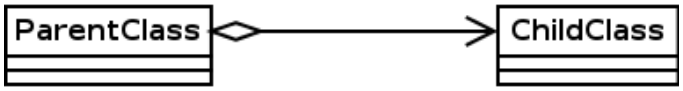


Figure 1.5: UML Class composition representation

The last term we need to define is *dependency*. This term indicates when a class uses another class. For example a factory class uses a class containing properties of the object to be created. This type of relationship is a more relaxed than composition or aggregation since the parent class does not own the child class. This

association is marked as a dashed line with an open arrow as shown in Figure 1.6.

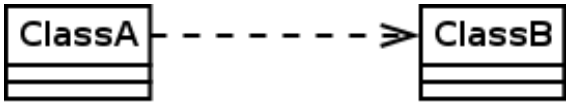


Figure 1.6: UML Class dependencies representation

1.4 SMART AI

AUTOPILOT MODULAR DESIGN

SmartAI (SAI) is a flexible, modular automatic flight control system. The main objective of the SAI is to control a vehicle (airplane, quadrotor, boat) to make it follow a path and perform actions based on predefined rules. The simplest case is to track waypoints and loiter around the final point. Filters, sensors, controllers and estimators are required to achieve this objective. A flexible autopilot allows an easy way to implement new features, modify existing ones and remove the ones that have become obsolete. With a modular design each feature is independent of the rest. In this section, we present the modular design used by the smartAI.

The components that form the autopilot can be address in three subsystems: Guidance, Navigation, and Control. Each of these subsystems can hold up to 100 components without appreciable change in performance. Each subsystem component is implementable as a simple block in a signal flow graph. We have blocks corresponding to numerical integration of nonlinear differential equations, signal summation, signal low pass filter, digital filtering and other basic signal processing

elements. The connections between the blocks needs to be flexible enough to allow block reconnection, re-usage and relocation. Due to its nature, the object oriented design is perfectly suited for the SmartAI implementations.

Since we do not want any component to be designed ad-hoc but rather to be reusable we specify the interface for Controller, Filter and PathPlanner. This implementation practice ensures that the behavior that each component inherits from the interface specified in a compatible manner. Since the basic signal processing elements such as summation nodes or gains can be used as controllers and filters, we add a BaseElement which act as the common interface for Filter and Controller. The design is shown in Figure 1.7.

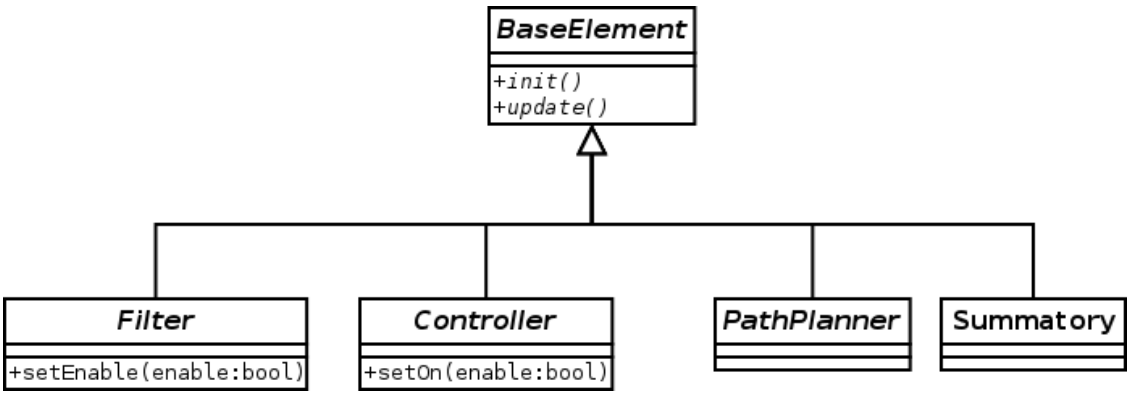


Figure 1.7: Interface design for smartAI components

Since the entire system is built from interconnected components, a configuration and connection system is required. The configuration and connection is implemented in run-time and is independent of any particular implementation. This distinction between the configuration and implementation is a requirement to achieve redesign without the need of recompile the source code. To address this

requirement we use the Builder pattern, which allows separation of the construction from the representation; hence the build method is used to facilitate multiple instantiations. The Builder is used to create and configure the components; using the builder to parse the configuration file will limit its applicability. To address this issue, a popular solution known as Configuration Parser is used. With the parser, we decouple the creation of the object from the interpretation of the configuration file. This separation allows us to easily change the configuration types from a standard format such as XML to a custom binary files, or an EEPROM memory reader without the need to modify the Builder. The entire design is shown in Figure 1.8. Upon the first initialization, the system searches for the configuration file supplied by the user, if the file is missing or no file is supplied the system fallback to a default configuration. This default configuration is no different from other files except that is protected so it can not be modified by the user.

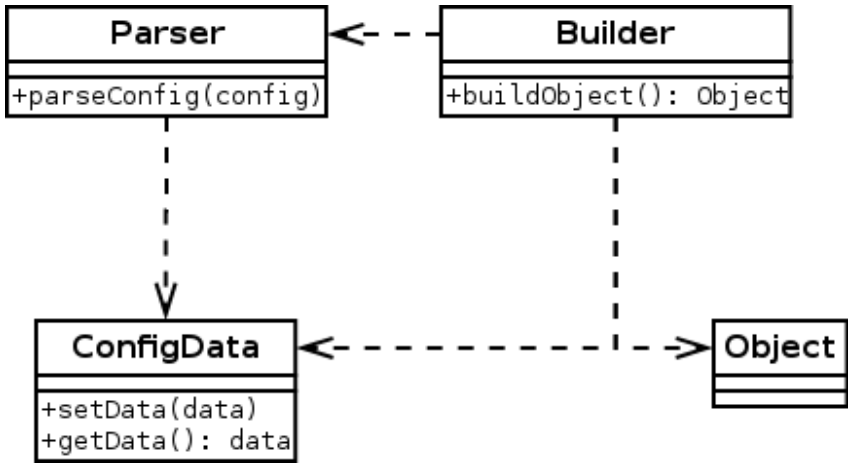


Figure 1.8: UAV Module Builder design

The construction patterns presented are used for the Filters, Controllers, Sen-

sors, Hardware Abstraction Layer and Path Planners. All these Builders are not intended to be used directly but rather by a higher level builder: the FactoryUAV, which will create an entire system. FactoryUAV is a software element that serves as a preprocessor to facilitate implementations by translating the configurations files into runtime modules. It also, performs the required connections. After parsing the configuration file, the FactoryUAV (which depends on the others Builder's) creates and connects all components of each subsystem. The Class design for the FactoryUAV and all its dependencies are shown in Figure 1.9.

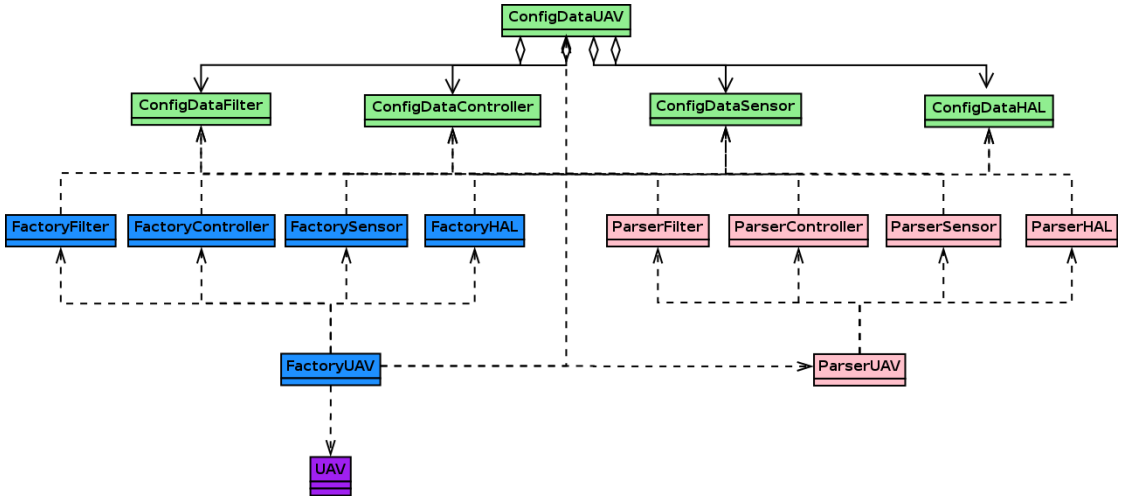


Figure 1.9: UAV builder and its dependencies

In Figure 1.10 we present the design of the UAV class with all the dependencies, which include Navigation, Guidance, Control and Hardware Abstraction Layer. As mentioned previously, with the help of the FactoryUAV multiple components can be implemented on runtime. The BaseElement interface provides the two methods required by the autopilot, *init* and *update*. The first function, *init*, which contains the calls to all the other methods and properties required to initialize the compo-

nent, is executed after the autopilot powers up, and is a part of the initialization process. On each step of the iteration the second method, *update*, is called. This function is a wrapper for all the operations required by a component.

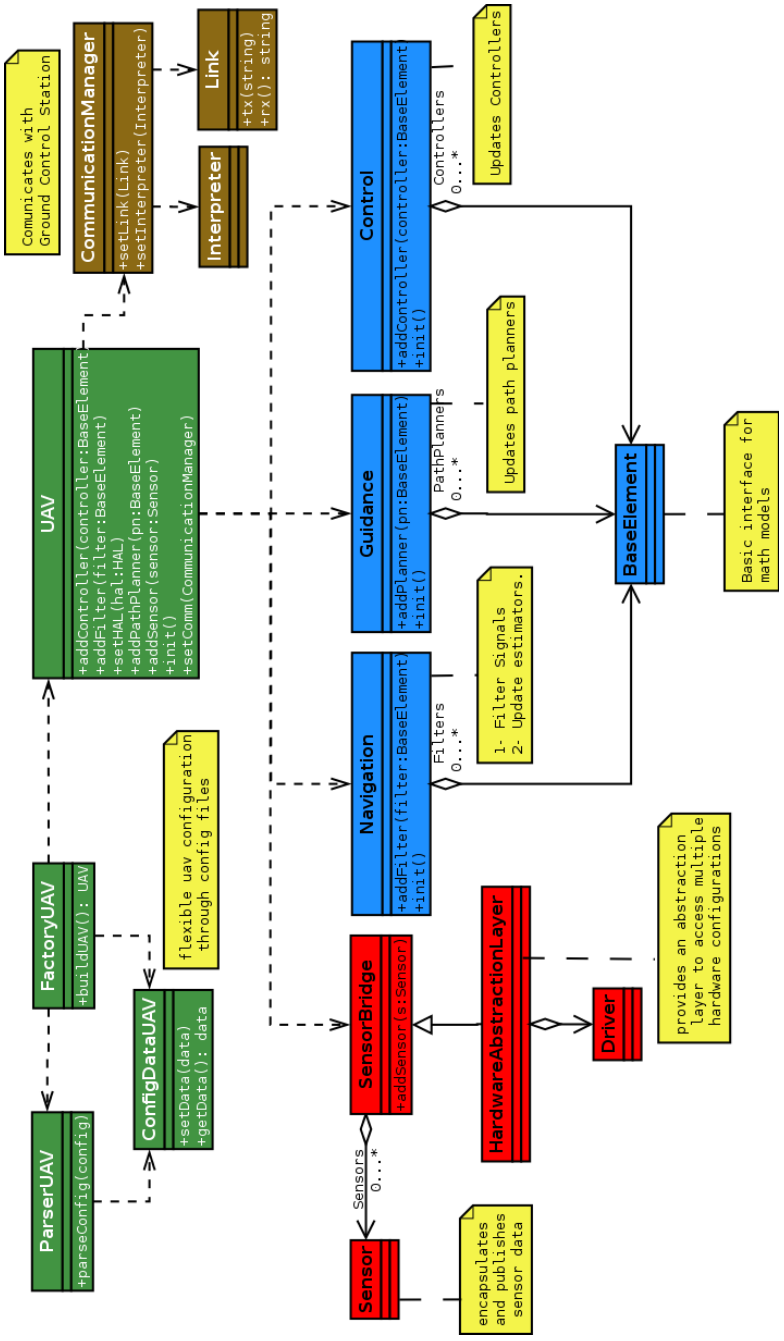


Figure 1.10: UAV Class UML Design

AUTOPILOT FILTERS

Filter is a special implementation of BaseElement. An additional method enables or disables each filter component. When a filter is disabled it keeps processing input signals, but does not publish the corresponding output data. The main idea, is to have the ability to switch between multiple filters based on the confidence of the estimation. For instance, the system can run simultaneously two Extended Kalman Filters. One to estimate attitude and position (primary) and a backup filter to estimate the attitude alone. If the GPS required by the primary filter loses the satellite signal, then the primary filter is disabled and the backup filter is enabled. Since the backup filter has been running in the background, there is no need to wait for initialization. The function that is called at each time step handles the update and propagation of the filter. It is up to the person implementing the filter to decide what to do in each cycle.

AUTOPILOT CONTROLLERS

Controller is a special implementation of BaseElement. As with the filters, a function enables or disables the controller; however once deactivated, the controller does not operate in the background. This behavior is particularly useful when operating vehicles whose controller structure changes with the regime of operation. For instance, the landing controller for a UAV maybe different from the controller used for takeoff and there is no need to run either of these controllers while cruising.

Some components need to be updated based on events rather than at a fixed

frequency. For example, the update function of a Kalman filter should be executed only when new sensor data is available, whereas the propagation is based on fixed frequency. To allow a mixed behavior, the Observer pattern is used, which is composed with two objects Observer and Subject. The Observer, is a software artifact utilized to monitor the state change of a specified variable. The Subject, is the variable to be monitored. All objects that need to be notified of the state change register with the Subject, and when the state change occurs the Subject notifies all the observers. A special kind of Sensor, Filter and Controller inherit from both Observer and Subject; this gives the system great flexibility since the communication utilizes a mixed configuration, namely frequency based and event based. The Kalman filter observes the sensors. When new sensor data is available the Kalman filter is notified. Upon receiving the notification the Kalman filters performs the update. On the background, the propagation of the states are performed based on the internal clock.

INTERMODULE DATA EXCHANGE

A key challenge in the modular design that needs to be addressed is the data exchange between different modules in the software implementation. The general approach in autopilots functions is to take the required parameters as inputs to the system. For instance a PID controller update function receives an error value, and the three required gains to synthesis and implement the controller system. The main drawback is that the calling function needs to know that the update function is

a PID controller and must accommodate the data accordingly. With this approach, whenever a new controller is added or an old one modified, the calling function also needs to be updated. To decouple the calling function and the module, the update function is standardized and shares the same prototype where the input is a pointer to a memory location. In the function implementation, the input parameter needs to be interpreted before its used. In an object oriented design, all the constant parameters can be stored inside the object reducing the size of the required input data on the update function. Although the use of a function pointer allows the separation of the calling function from the module, we still have a tight coupling between the filters that set the input structure and the controller using the data structure. To achieve a complete decoupling between objects, all data required for the update is stored internally. In case of time-varying parameters, the object holds a set of pointers to them.

With threaded applications concurrent data access is a common issue. Although data can be copied as many times as required by each operation, the access and copy operations add some computational expense and memory management operations that can be easily avoided. The use of pointers allows the sharing of data without the need of multiple copies. However care must be observed when accessing it. A simple pointer does not guarantee a sequential access to a memory critical section. In other words, this practice is not thread safe so there is no guarantee of data integrity. For instance, a filter may be writing a value at a particular memory location when a controller is trying to read it. In this case clearly the data read by controller will be corrupted. An approach used by Flight Simulators[30] is to use

a data pool. Each component requests data to and from the pool which ensures that the queues proceed in an organized fashion. The main thread of a simulator solves all blocks in each time step, so all components run at the same frequency. On the other hand, an autopilot runs in a decentralized schema and actuation. A filter may need to run at 1000 Hz but the controller (due to driver limitations) may not be able to perform faster than 100 Hz, hence multiple threads are used. Using a pool may cause a controller to lock a filter for several time steps which is not admissible. To address this issue, a special object Signal is designed and implemented in SAI. Signal is nothing but a variable container with methods to protect the data from concurrent access. Mathematically, this container provides a sequence that is indexed as a function of the time. Although it seems like these decentralized methods increase the complexity of the data exchange it only adds a mutual exclusion[32] (mutex) for each variable. The mutex ensures that only one thread has access to a critical section of the memory at a time. The most important advantage is that a method only locks the specific variable requested for the period of time that it takes to get or set the variable; this way the wait times are kept to a minimum and the lag get localized.

To assure a proper inter-block connection, all signals are created through a Signal Pool which creates the signals upon request. In case a Signal is requested more than once, the pool returns the signal from the initial request. In this way, blocks sharing a common signal are connected. The Signal Pool is required only when initializing the components since each component stores internally all required signals.

With the Signal-based design presented in this section the connection of each block can be done on runtime which allows a complete system redesign without the need to recompile the entire source code. It also allows a redesign on-the-fly. To add a new filter, estimator, or controller, there is no need to modify any of the existing modules since all the components are independent. The user simply needs to instantiate the elements from the templates provided within the SAI framework.

HARDWARE ABSTRACTION LAYER

The entire system (filters, controllers, sensors) needs to communicate with the autopilot hardware. This software-hardware interaction is accomplished using the Data Acquisition (DAQ) toolkit provided in the selected computing platform. The capabilities and operations of such toolkits vary greatly between computer types. This variations creates a strong dependency between the autopilot software and hardware. The abstraction between the software components and the hardware becomes a key point in the entire system design if reusability and component independence is desired. With a careful design and an appropriate implementation of an additional layer of software, it is possible to design a layout that allows a switch in the hardware by a contained and reduced change in the autopilot software, mainly updating the DAQ toolkit. To achieve this goal, we use the Mediator pattern developed by Gamma et. al. in.[29] This pattern encapsulates the interaction between different objects. The mediator permits the decoupling of each component with the Driver and SensorBridge as shown in Figure 1.11. SensorBridge is the

class that access the inputs of the computing platform. Driver sets the outputs of the plant. This separation between the inputs and outputs of the plant reflects the distinctions between the different hardware modules embedded on the computing platform.

We can see from Figure 1.11 that Control and Navigation do not interact with the hardware at all, instead they rely on Hardware Abstraction Layer to perform all operations related to the hardware. With the proposed approach each computer requires a HAL with the appropriate driver which requires a custom configuration; hence the dependencies on the Factory and ConfigParser.

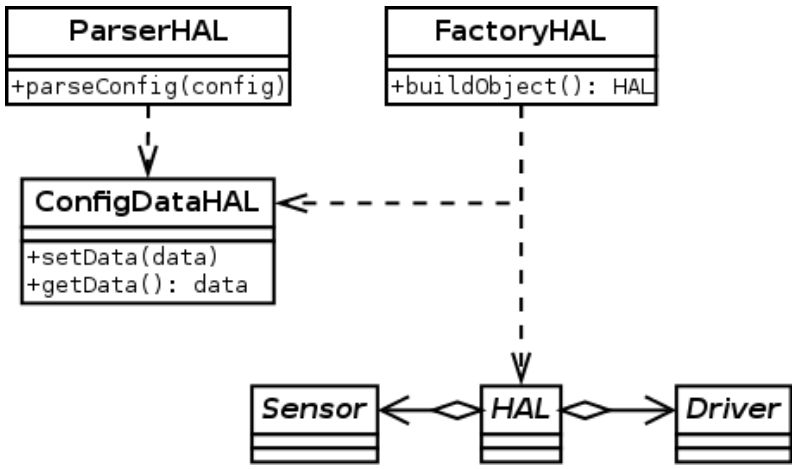


Figure 1.11: Hardware Abstraction Layer Class UML Design

Data acquisition is an important component of the auto-pilot system where the sensed data from pitot tubes, IMU, GPS systems is processed to provide state estimates for control purposes. Our system supports the following data and connection types PWM, PPM, Analog, Digital, Serial, I2C or SPI. Additional formats can be accommodated by using appropriately definitions and updating the HAL software.

The simplest type of sensor is a passive sensor. The reference value is obtained and made available for use by other modules by connecting directly to one of the boards inputs. For example, consider the case of a temperature sensor. It is connected to an analog input available in the IO ports. The Analog to Digital Converter (ADC) connected to this input generates the digital signal that serves as an input to the autopilot system. On the other hand, the active sensors are much more complicated and they are connected to one of the board buses (I2C, SPI) or ports like serial port. Sensors such as GPS or IMU requires an initialization sequence which requires the user to send a set of configuration messages. These connections are bidirectional. The sensor data acquisition is through a bidirectional communication mechanism which requires data polling and/or data broadcasting; in either case a simple read function does not suffice. The design of the active sensor has an interface defined by SensorActive which holds a Link that defines the connection type. When the update function is called, the sensor gets hold of the link which allows the data exchange required for the data acquisition. Subsequently it populates properly instantiated signals to make the information available to other components. With this design several Sensor objects can share a link, a capability required to exploit the bus feature of multiple physical sensors connected at once. The design shown in Figure 1.12 demonstrates how a commercially available IMU (MPU600) is implemented in our framework. Also, there is a serial GPS; this is the most popular connection for this type of sensor.

The output of the autopilot is communicated to the actuators using Drivers. These Drivers are a software object which encapsulates the computing platform

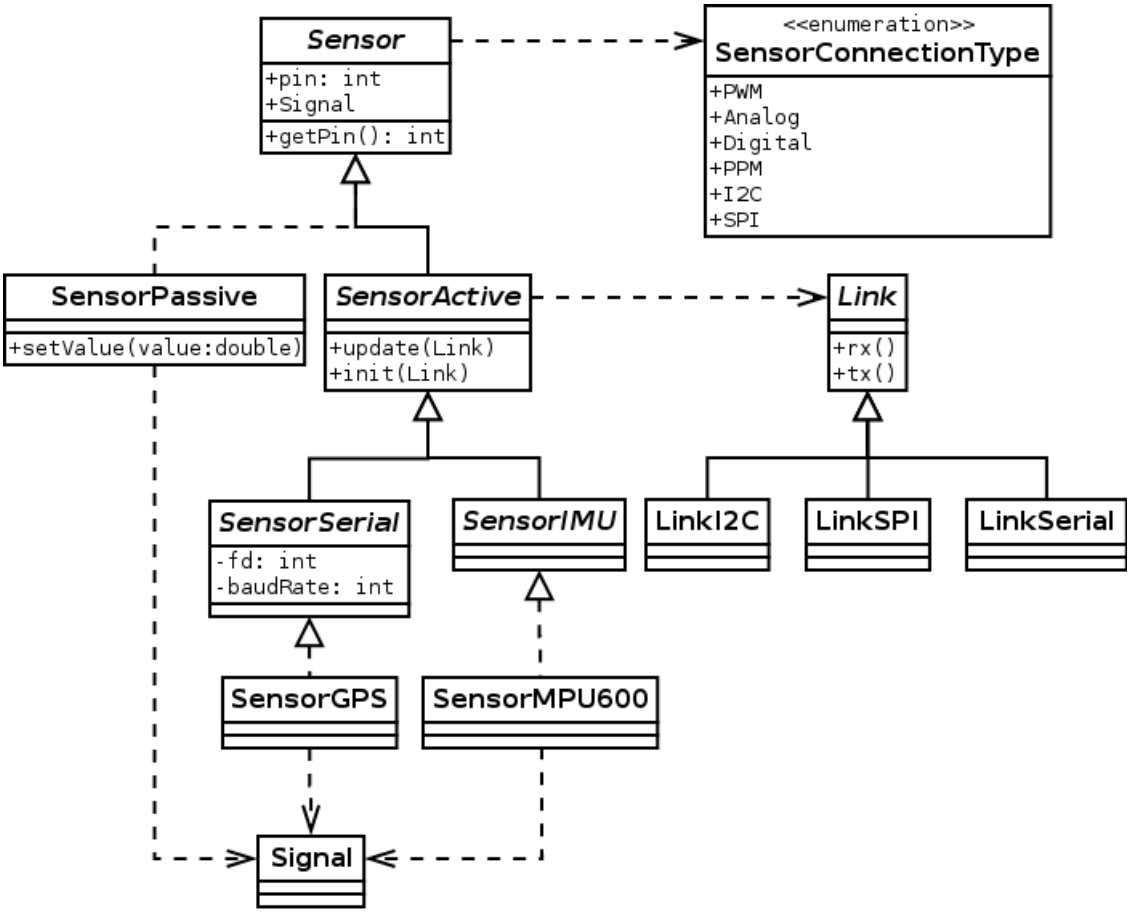


Figure 1.12: Sensor Class UML design

API. The API is a set of lower level functions that access the devices drivers by writing to a specific memory address. Each driver has a type (PWM, PPM Digital or Analog). Much like the passive sensors the Drivers only requires one signal per output. On each update loop the HAL will read the signal and set the hardware output.

In Figure 1.13 we show an example for a gain scheduling controller that takes as an input the wind speed and altitude and sets the throttle. For this example

the autopilot is running on a RoBoard computer. On the left side of the diagram we show the configuration and build dependencies for the HAL, which in this case builds the HALRoBoard that depends on the RoBoard proprietary libraries to access the computer DAQ. We add one analog sensor for the wind speed,one digital sensor for the altitude, and one driver which drives the motor speed controller through a PWM signal. Finally a gain scheduling controller is used to control the motors speed. It uses as inputs the sensor data.

The presented design in this section not only allows a clear and clean way to reconfigure the hardware but also enables us to easily embed a simulator to validate the entire autopilot framework and its subsystems. To enable the simulation we design a HAL that exchanges data with another application which generates the sensor data and processes the controller commands. Another validation approach is to use a hardware-in-the-loop simulation where the simulator output-input is done through the autopilot hardware interfaces. An example of this validation is to generate synthetic GPS measurements in a computer and feed them through a serial port to the autopilot. In this case, since the simulation is done with an external hardware to generate the sensor data and process the driver signals, the autopilot does not require any implementation changes.

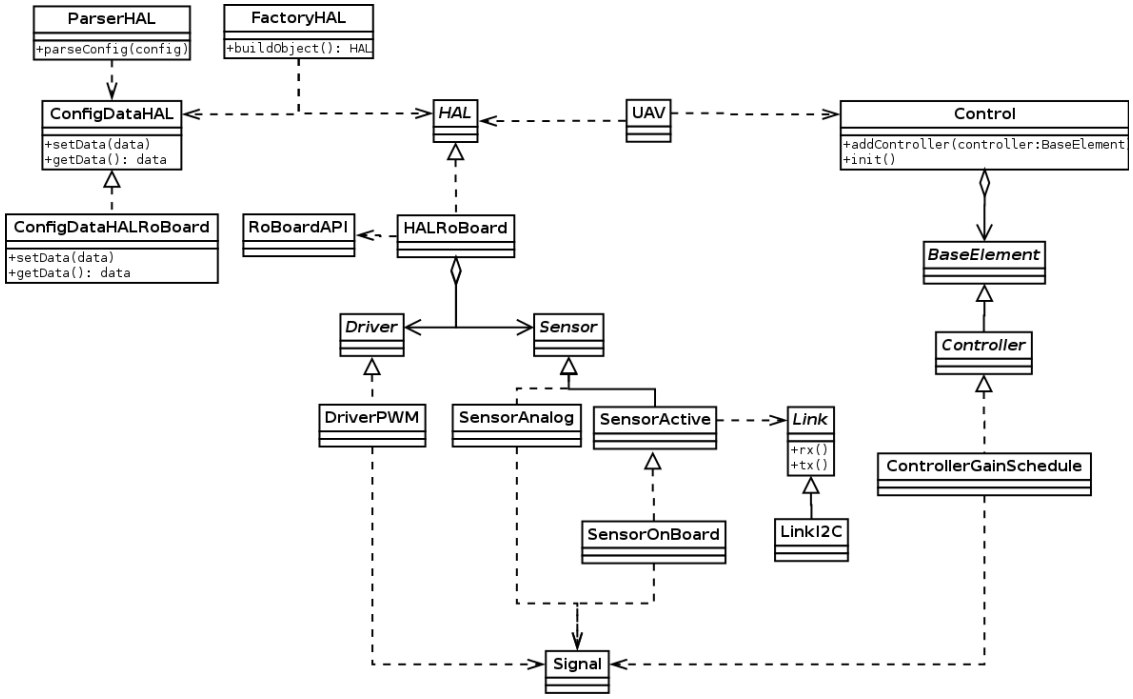


Figure 1.13: RoBoard HAL Class UML design

1.5 GROUND CONTROL STATION

INTRODUCTION

The main idea behind autonomous flight is to allow a vehicle (plane, helicopter, etc) to accomplish a set of task with minimal to no human intervention. To complete this task the vehicle relies on sensors and advanced algorithms that need to be executed off-line or in real time. It is clear that a ground control system that can accommodate multiple tasks, vehicles, sensors and algorithms is required to provide flexibility to the autonomous system operation. However the design of

such platform is not trivial.

All autonomous vehicles, no matter how advanced or capable are need a remote control station, even just to start the mission or terminate it eventually. Since the SAI was designed from scratch, an equally capable remote station was required, hence the Ground Control Station (GCS) is developed. The main objective is not only to communicate with a remote vehicle and perform basic mission control, but to also increase the vehicle capabilities and enhance the overall system.

The GCS was designed with flexibility and scalability in mind. To achieve this a modular approach is used to obtain a versatile yet powerful system. To achieve these core features of modularity each mission is divided into tasks and each task is assigned to a module. A variety of standard mission are developed to carry out a variety of tasks ranging in complexity. The main advantage of modularity is that the behavior shared in a mission is translated into a module reuse. For example, if an entire mission is to be done with two different vehicles, only one module needs to be updated. The modularity concept is used in a recursive fashion, where each module is built with submodules which exploit the re-usability concept of the entire system even further. This design allows the user to develop a multiple-vehicle and multiple-sensor mission. The design is flexible enough to integrate existing solutions, toolboxes, and commercially available vehicles.

In this section, we present the general physical layout of the GCS subcomponents, where each component and its required hardware is clearly identified in Figure 1.14. The gray blocks represent the different required hardware implementation devices. These computers only require a minimum TCP/IP network and

should have a kernel that supports multiple threads (almost any embedded and/or single board computer supports both). We can mention Beagle Bone, Raspberry, as examples of embedded computers capable of running GCS. For large systems, (several vehicles) and/or intense tasks such as those required by complex algorithms for route generation, estate estimations, or image processing, the computers can be chosen to satisfy the required computation power. If intense computations are not required, all units can run on the same computer which can be an embedded systems and/or single board inexpensive computer.

Each module is described in detail in the following sections. However, here we present a summary of the task each module performs. The green block represents a vehicle unit. This unit requires a platform (e.g. plane) with a minimum controller on board and a Platform Remote Controller (PRC) on the ground connected through a control bridge. In case the platform carries a sensor a Sensor Station (SS) may be needed to process the data. It is important to note that each platform requires a unique set of PRC and SS. The PRC is connected to the Air Traffic Controller (ATC) which handles the air traffic which in turn is connected to the Mission Planner (MP) that performs the mission control task. All the data is exchanged with external users through the Information Center (IC).

PLATFORM REMOTE CONTROLLER

The platform remote controller (PRC) is the main component on the GCS. The main task of this block is to interact and enhance the vehicle on-board autopilot.

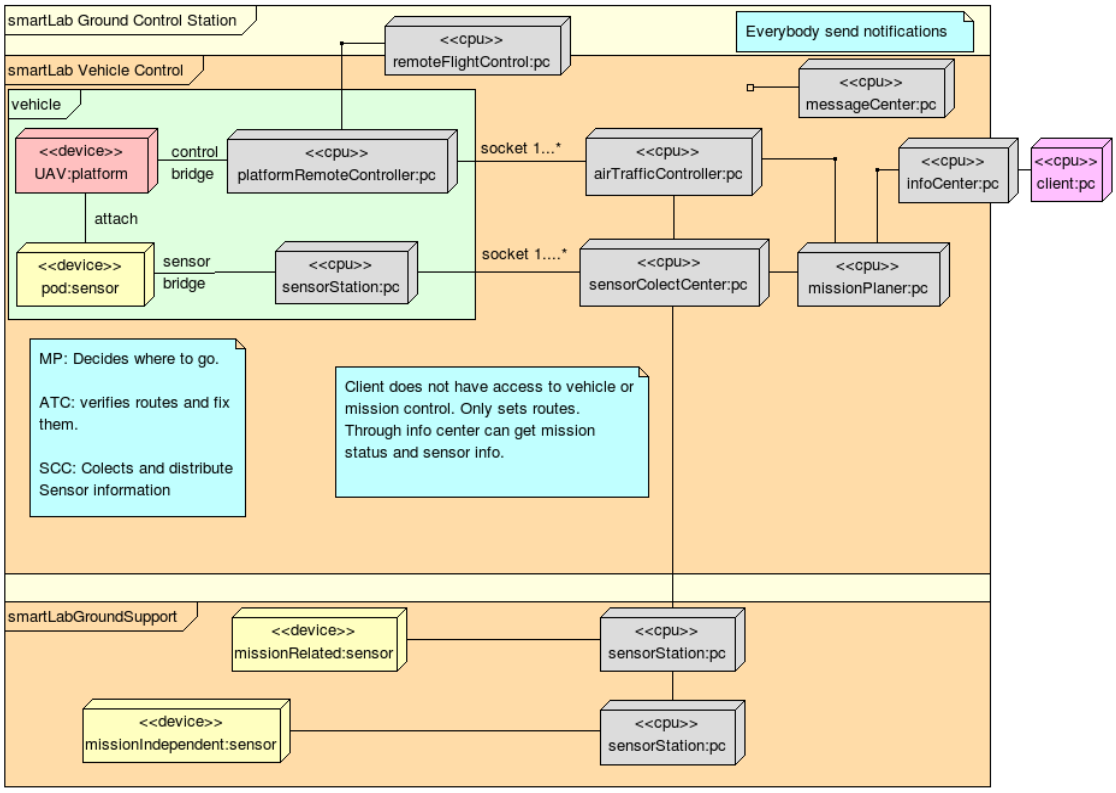


Figure 1.14: Ground Control Station modules layout

Depending upon the autopilot capabilities, this module has a range of tasks ranging from simple tasks such as forwarding a waypoint from the GCS to the autopilot to complex tasks such as data fusion filtering for navigation purposes. This module is designed to adapt the internal protocols, reference frames, control actions and commands to each vehicle custom communication protocol. Although it basically needs to translate the messages between the autopilot and the GCS, it is also granted the capability to effectively implement the complex control laws and estimators. To complete these tasks, the module must have access to the physical connection to the vehicle (e.g. modem, wifi board, etc.) with a proper set of drivers and any API required to get a reliable working connection.

For each different type of vehicle with a unique autopilot, a different PRC is implemented to accommodate the communication protocol. However controllers and filters can be used following the signal and module approach used on the SAI. Most autopilots provide a connection protocol that is transmitted with the use a of a link that requires a particular driver. Generally speaking it uses a wireless modem. The realization of a PID controller remains constant regardless of the vehicle. The ArDone parrot and the Ardupilot, both commercially available autopilots that are implemented on this framework share most of the controller design. For each vehicle a set of gains is chosen.

The first task is simply to accommodate the GCS protocol and commands required by the on-board autopilot. If the vehicle is capable of tracking waypoints autonomously, the route can be split and communicated to the vehicle one at a time. Some autopilots (the most basic) require lower level control i.e. commands

wise (roll, pitch, yaw); in this case the PRC needs to solve the trajectory and feed the required commands to the vehicle. This is the case of the ArDrone Parrot quadrotor which is used to perform experiments on Chapter 6. On the other hand, the second task complexity is heavily impacted by the quality of the autopilot.

Fully featured autopilots may be able to process the entire route, making the PRC nothing but a proxy, this is the case of the Ardupilot. The outcome of the experiments carried out using Ardupilot is shown in Chapter 6. For the most basic autopilots that require vehicle inputs, the PRC must solve position and attitude filtering, stabilization control, and waypoint tracking. Any combination in between these scenarios is also acceptable.

The PRC operates the "Navigation, Guidance and Control" loop. In Figure 1.15 this diagram is shown. In this case, there is an added component "Communication" to manage the message to and from the vehicle and other GCS modules. In Figure 1.16 a more detailed diagram for one operational cycle of the PRC is presented, where, the "Vehicle" block represents messages sent to and received from the autopilot. CommManager is the thread corresponding to the messages exchanged between GCS modules. The navigation module starts by requesting all the navigation sensor information to the vehicle to update the filters. Once the filters are up-to-date, a notification is sent to the others members (guidance, control, and comManager). The control module then requests the next waypoint to guidance and solves the control action, then sends the action to be taken to the autopilot. In the background commManager keep exchanging the vehicle current position and the new routes with the ATC. When a new route arrives it is immediately for-

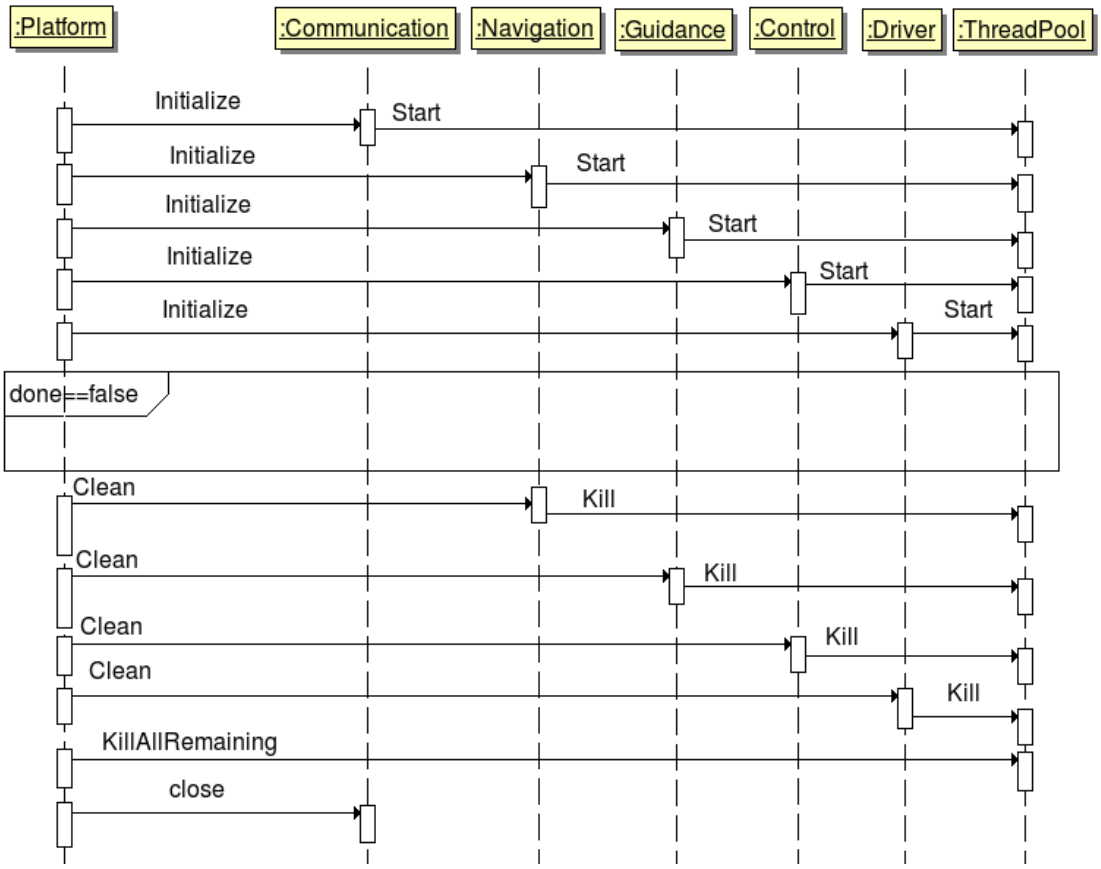


Figure 1.15: Basic operation of the PRC

warded to Guidance. The Remote Flight Center Module RFC waits to gain control over the vehicle whenever the user specifies it. A more detailed description of each sub-block is provided below

- ***Vehicle Connector***: The vehicle connector manages the information exchange between the PRC and the vehicle in a reliable way with the use of industry standard protocols. It provides a bridge to send and receive data by handling the incoming and outgoing queue. The incoming message are added to the queue until the navigation module request a message. After an inter-

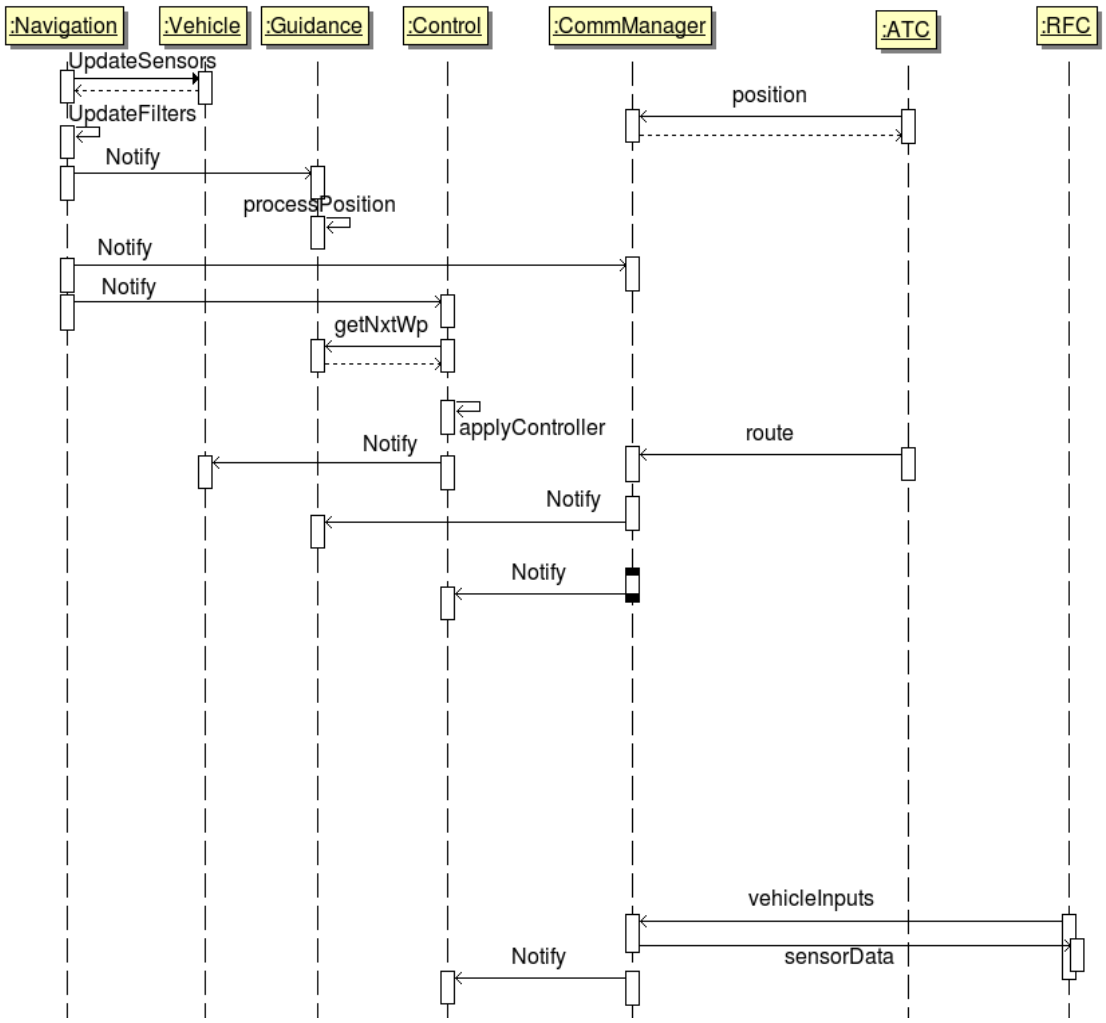


Figure 1.16: Basic operation of one cycle inside the PRC

nal message is received from the driver module, it is added to the outgoing queue. The outgoing queue is processed and each message is sent on the first opportunity based on the hardware load.

- ***Navigation***:This module is the only module with access to the sensor data through the Vehicle Connector module. Therefore it needs to provide a good

estimate of the current position, attitude and their time derivatives and any other quantity required by the controller and guidance modules. When an advanced autopilot has the capability to provide the required estimates this module just publishes these values to the internal modules. Whenever the filter generates covariance information, or any other metric to evaluate the quality of the estimate, will also publish these data.

- ***Guidance***: The main objective of this module is to store and feed the waypoints of the current route to the controller module; however more advanced algorithms can be implemented. For the missions where formation flying is required or a more complex path plan logic is desired the required submodule must be implemented. All the routes are received from the ATC. With the information provided by navigation the decision of whether a waypoint was reached or not is made by the guidance module. When the vehicle is inside the waypoint window the next waypoint in the route is fed. If no more waypoints remain in the queue then a predefined actions such as wait, land, however, etc is performed.

- ***Control***: The control modules generates the a list of required commands and actions required by the vehicle to reach the position and attitude provide by guidance. This module holds an array of controllers that can be run in series or parallel configuration to reach the objective. These controllers operate with high level variables such as roll and throttle. This approach grants the ability to share the controller among different vehicles and missions. The controllers communicate with the signal schema presented in the previous sections.

- ***Driver***: Since the controller generates a high level signal that needs to be converted to a lower level hardware output, a module to accommodate this conversion is required. The driver module not only translates high level signals to low level vehicle commands, but also performs any reference frame and unit conversion required for effective execution of the mission. Then, it packs, the information following the rules of the selected communication protocol and adds the message to the outgoing queue.

- ***Communication Manager***: The communication manager is responsible for the message exchange. Message exchange verification is required to communicate tasks in a reliable manner. When the communication is done using a TCP[33]/IP[34] network such as the one used by ArDrone, there is no need to perform extra checks since the low level protocol ensures message delivery.[33;34] However when communication over a wireless modem is used, the protocol should implement message verification, data validation, and proper replays to ensure message delivery; an example is the MAVLink[35] communication protocol for micro aerial vehicles. The communication with the ATC and FRC is over a socket, hence there is no need to perform checks or data validation, however the data packing and unpacking from the data stream is performed. The communication with the vehicle can vary from vehicle to vehicle hence the need to support multiple protocols.

To further illustrate the flexibility of the system, a brief demonstration of the use of the PRC with two different types of vehicles is provided here. The ArDrone Parrot requires the control actions to be sent at 1000 Hz over a UDP port, where

the control actions should be desired attitude and vertical speed. Setting the proper attitude allows the user to indirectly control the velocity. Since this vehicle lacks the capability of position tracking, the PRC needs to perform the waypoint tracking with the use of an external sensor such as Vicon for indoor applications. To achieve path tracking the PRC receives the routes from the ATC, converts them to a set of attitude commands and implements the feedback control laws to assure the parrot follows the required trajectory.

Another platform that has been added to the library is an RC airplane controlled with an Arduplane[11]. The Arduplane autopilot have waypoint navigation capabilities, but they lack the ability to dynamically updates routes. For these vehicles the PRC needs to receive the routes from the ATC, extract the waypoints and deliver them one at the time. Whenever the vehicle is inside the waypoint window a new waypoint is dispatched. The PRC implements the entire MAVLink protocol to allow a robust communications.

SENSOR STATION

One of the key application of autonomous air vehicles is to serve as distributed sensor platforms. The mission, the sensors, and the vehicle vary greatly from application to application hence accommodating all possible combinations may seem impossible. The use of a Sensor Station (SS) is aimed at decoupling between the vehicle and the sensor, serving as an abstraction layer to interface the GCS with different types of sensors and their required drivers and/or software. This separation,

is made possible thanks to the fact that most autopilots use different communication links for the sensor data and for the control commands. A clear example is airplanes carrying cameras for operations such as objective identification, surveillance or terrain mapping. The video feed is transmitted using an analog link which broadcasts data in a standard TV format such as PAL or NTSC, while the controller communications essential for UAV operations are carried out on a low bandwidth modem. The SS has a handler to the communication channel on a computer that processes this data. Multiple sensors can be carried on the same vehicle and multiple computers can process the data on a parallel schema. The SS design simply consists of two parts: the Sensor and the DataProcessor. The Sensor uses the modem driver to retrieve data from the physical sensor installed on the vehicle and accommodates the format required by the data processor. As an example we can mention a SS using a USB dongle to read a NTSC video stream received through a wireless receiver, as shown in Figure 1.17. The Sensor uses a driver to access the video stream, then accesses a ffmpeg library to process the video, generate frames, and convert them to a RGB matrix. With this design, if the video stream format changes, only the driver and the Sensor block need to be updated, leaving the DataProcessor unchanged.

The DataProcessor receives data in a processed format, such as frames of a video stream in a RGB matrix, data structures containing position and wind speed, among others. It performs different data manipulations to extract the required information. If a 3D map is to be generated from a video feed, the data processor extracts features from each of the received frames. Edge detection for identifying

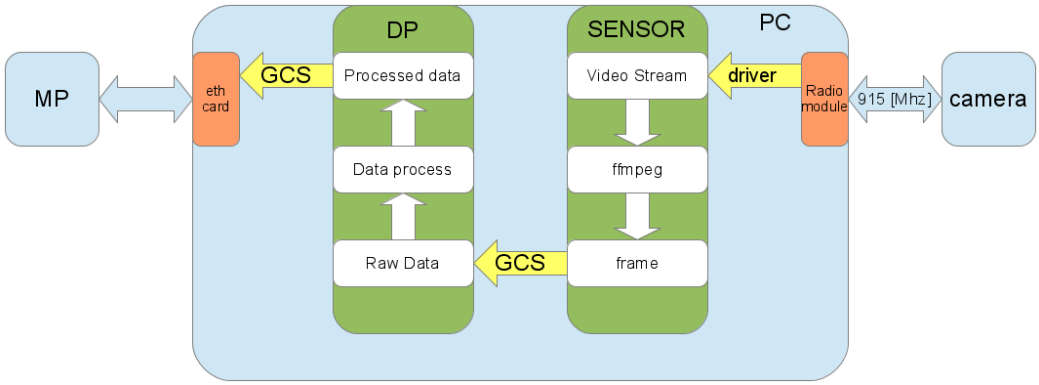


Figure 1.17: Sensor Station layout required to process a NTSC video feed

objects is another task the data processor performs. Other types of data processing not related to image processing can also be performed, such as generating the velocity fields in a local region from wind data, or identifying vehicle position. In the present dissertation, data processor is used to perform object tracking and visual landing the results are shown in Chapter 6.

MISSION PLANNER

The Mission Planner (MP) is the key component to extend the capabilities of the autonomous vehicle. This module handles the vehicle mission and performs the required tasks that cannot be done onboard the plane. For simple waypoint navigation missions, the MP generates routes based on predefined rules and sends them to the PRC. If the mission involves a sensor that requires intense processing

such as generating a 3D map of the terrain based on an image stream, the MP receives the processed data from the SSC (which performs the heavy processing and data fusion) and the vehicle current position from the ATC, then decides which direction to next lead the vehicle, and finally generates the new route and delivers it to the PRC. The MP stores internally all required data for route generation such as vehicle position, vehicle performance and mission objective. With this data, the required trajectory to complete the map will be generated. This process is recursively repeated until the entire area has been mapped with the desired quality, or the vehicles run out of operational time.

It is important to note that the mission planner directs the plane and controls the quality/status of the mission, but does not control the vehicle or the sensor directly. This decoupling of the mission, the vehicle, and the sensor gives the GCS great flexibility. To generate a wind velocity field instead of a 3D map, the only hardware changes required are the replacement of the physical sensor attached to the vehicle, and to update the SS. These changes should be reflected on the configuration files. The mission planner remains unchanged. If multiple vehicles are used, the MP can incorporate easily the multi vehicle capability. The algorithms to solve multiple trajectories must be created, but the SS and PRC remains the same except that multiple instances are used.

When the mission is performed on behalf of an external agent, real time data can be provided. The MP decides which type of data on what frequency can be published, and it also filters external commands. This assures that the mission critical operations are carried out first, and only when the vehicle is on a safe

trajectory the information is updated.

INFORMATION CENTER

Most of the time, the vehicle performs a task to assist in a more complex mission or complex situation. Hence external clients must be integrated with the GCS. The Information Center (IC) acts as a proxy to allow any external program/client to interact with the GCS, but without interfering with the mission. Since the system needs to interact with external clients which may send routes, actions, missions or any other messages related to the mission, (using their own proprietary protocols) the use of a proxy to accommodate this heterogeneous set of messages is required. It may seem daunting to accommodate the needs from multiple input to multiple outputs, however with a careful design the IC is capable of translating messages from external clients to an internal format and vice-versa, much like the PRC does with the autopilot. At this point, it should be clear how with the development of each new module, the possible combinations grow exponentially with minimum coding. The IC also protect the GCS from possible errors, crashes, and failures in the external client algorithms. Since the data is processed and verified before forwarding it to other modules, it is possible to detect system crashes, failures, and other problems before they happen; thus protecting the vehicle.

With this module, there is no need to change the internal protocol and/or connection diagram inside the GCS to accommodate multiple request from different clients. With this module, in the loop it is easy to develop applications in different

OS and languages to gain high level control over the mission. However the most important feature of the IC is to act as an insulation layer preventing any error or bug on the client side to compromise the system and/or vehicles.

The IC receives data from the MP and publishes it and forwards it to the clients; for instance when objects are being tracked, the IC displays a map with a mark on the current position of the vehicle and another mark on the objective if the position is known. This module operates at a lower frequency than any other module and the data refresh has the lowest priority of the entire system.

AIR TRAFFIC CONTROLLER

When multiple vehicles are flying there must be a method to keep them in safe paths to prevent collisions and crashes. This is the task of the Air Traffic Controller (ATC). The ATC acts much like the air traffic controller found in airports; Upon receiving a route, it verifies that there are no impending collisions and that the vehicle is projected to remain inside the airspace. If there is any risk of crashing due to terrain elevation, or the presence of an obstacle (including another aircraft) the trajectory is flagged for a potential update.

The route generation is to be performed either by the MP or by an external agent, however relying solely on this is not sufficient since the routes may be the outcome of rather complex and/or unstable algorithms. A route when received cannot be assumed safe, since errors in the altitude due to the terrain elevation or distance from other objects may have been neglected by the preceding processes.

Furthermore, when multiple vehicles are in the air at the same time it is possible that the current position of others vehicles is not considered by the route generator due to limitations of the algorithm or unavailability of data. Validating this data before forwarding to the vehicle is therefor mandatory for safe operation of unmanned vehicles.

The ATC block guarantees safe operation when multiple vehicles operates in the air simultaneously. The main task is to verify that the vehicles are not on a collision trajectory. If any danger arises, the ATC reroutes the vehicle to avoid the collision. The routes are compared against the defined airspace to prevent vehicles from entering into hazardous or inhabited areas. Finally, the ATC checks routes flight levels to assure there is no potential crash with the ground or any object present inside the airspace.

The ATC receives routes from the MP and as soon as a route has been received its integrity is verified. Once the route is approved, it is forwarded to the vehicle, and an acknowledgement message is sent to the MP. In case a potential problem is found, the ATC fixed and sends the route to the MP for approval. If the problem cannot be fixed, the route is rejected and a notification is sent to the MP.

In the following section, a set of simple yet efficient correction rules are presented and described. These rules are intended to keep the vehicles safely within the defined airspace.

AIRSPACE CHECKING MECHANISM

Since all UAV flights are required by FAA regulations to be confined to an authorized area it is important to check that all the waypoints and routes lay inside the allowed airspace. Usually, the airspace is a concave area such as squares or circles. In these cases only the waypoints should be checked, and not the entire route. If all the waypoints are contained in the airspace, all the routes will also be contained due to the concavity properties. On the other hand, if a convex airspace like an L-shaped area is used, all the routes must be checked after discretization to ensure they lay within the allowed region. The check performed is a rather simple algorithm to push waypoints into the allowed airspace. If any waypoints are in fact outside the area, they are pushed to the closest point inside the airspace. The simplest way to achieve this for rectangular areas is to use an "if limit" check as shown in Algorithm 1. A simple example is shown in Figure 1.18 where in red is shown the original route and in green the fixed route. For more complex areas a numerical algorithm must be implemented to estimate the shortest distance from the point to the area, and this direction is used to push the waypoint.

Algorithm 1 Airspace Check

```
for All waypoints in route do
    for i from 1 to 3 do
        if x_i is greater than x_i^M then
            set x equal to x_i^M
        else if x_i is lower than x_i^m then
            set x equal to x_i^m
        end if
    end for
end for
```

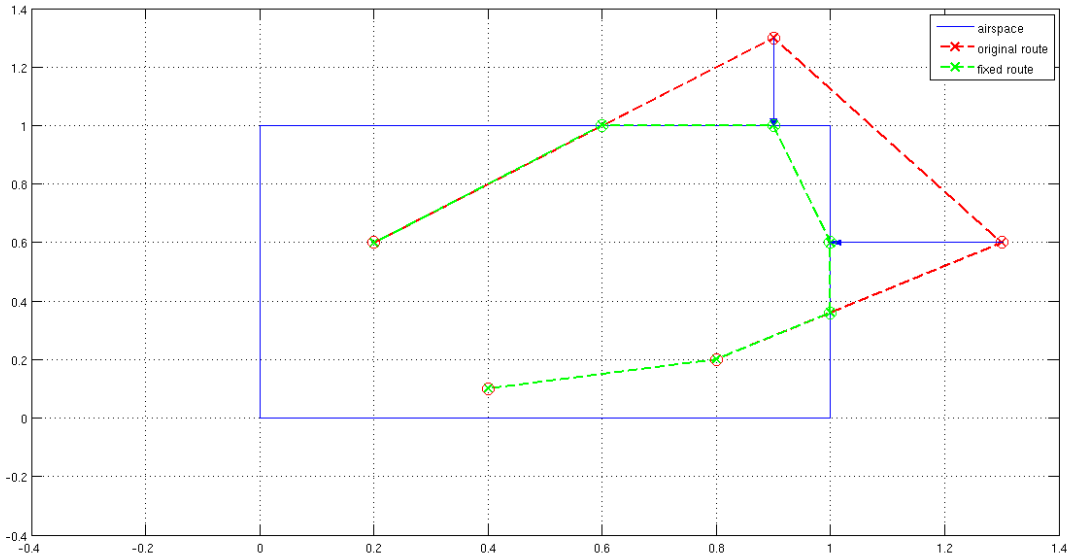


Figure 1.18: ATC route correction example

RANGE CHECKING MECHANISM

For each vehicle, the ATC holds the remaining flight time as reported by the vehicle to reduce the risk of a crashing due to fuel shortage. Knowing the aircraft properties, such as speed, it is possible to estimate the range based on the remaining flight time. For each waypoint the distance to the previous waypoint is compute; this gives an estimated range to each waypoint as shown in Equation (1.1), if for a given waypoint $r_n > r_{max}$ then that and all successive waypoints are dropped because they are outside range. This provides a method to always verify the feasible reference way points for UAV to track.

$$r_n = \sum_{i=0}^{n-1} ||p_{i+1} - p_i||_2 \tag{1.1}$$

DISCRETIZATION PROCESS

Once the route has been received and all the waypoints lie inside the airspace within the vehicle range, the vertical distance to the terrain needs to be validated. It is possible that the surface of the ground is flat, but this is most unlikely since multiple objects may be present and the terrain geometry is generally irregular. To obtain a more robust result the route is discretized by inserting a subset of waypoints between two consecutive waypoints. The distance between two successive waypoints is divided into N segments of length $\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$ where Δ represents the interval of the discretization. The interval length is defined based on the size of the airspace, the distance between waypoints, the size of the plane, and the type of terrain, among other properties.

GROUND AVOIDANCE ALGORITHM

In this section, a simple yet efficient ground avoidance algorithm is described. This algorithm works only by changing the altitude of the route and not the heading; hence the vehicle *always* avoids an obstacle by flying over it rather than around it. The 3D route presented in Figure 1.19 can be converted to a two dimension route using a local terrain fixed frame at near sea level. In the two dimensional projection the abscissa axis represents the discretization step and in the ordinate is the flight altitude and the terrain elevation as shown in Figure 1.20.

After the route is converted to a two dimension trajectory all the waypoints are checked to ensure that they are located at a safe altitude. The flight altitude should be greater than the elevation of the terrain in the current position, plus the

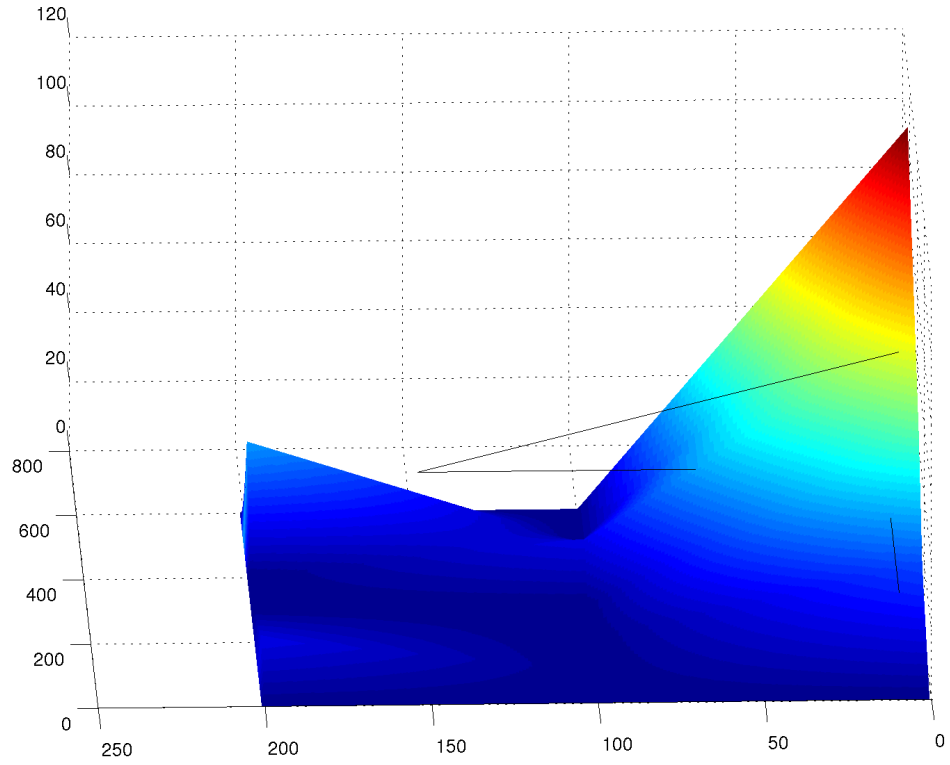


Figure 1.19: Original 3D route as received by the ATC

minimum flight altitude of the vehicle. If not, then the waypoint is pushed up to this boundary as shown in Figure 1.21. The new flight path generated may have climb and/or descent rates far greater than the maximum allowed by the vehicle. All the slopes of the segments are checked to verify that they lie between the maximum and minimum limit. If the slopes are outside the boundaries then they are reduced or incremented to fit between the boundaries and a new route is generated Figure 1.22. The climb slopes are verified starting from the last waypoint and moving backward; the dive slopes are verified starting from the first waypoint and moving forward.

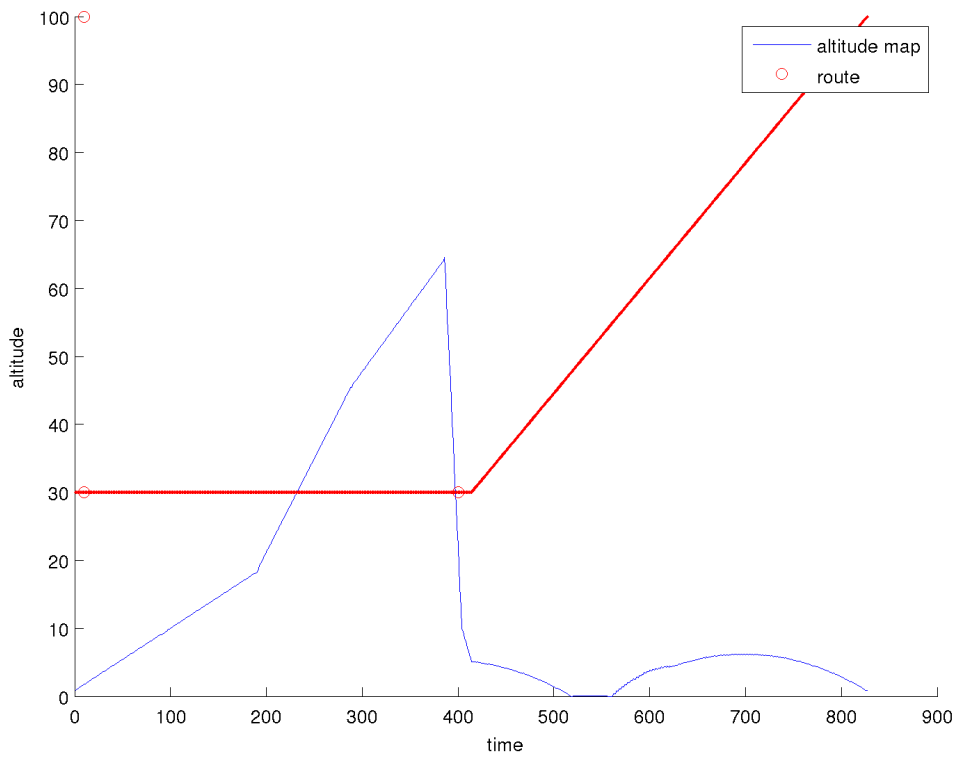


Figure 1.20: Original 3D route as received by the ATC converted to 2D

After all the corrections have been applied, the route is converted back to a three dimensional contour Figure 1.23.

PARSIMONIOUS REPRESENTATION OF REFERENCE TRAJECTORIES

Once the safety checks have been applied to the route, the extra waypoints are removed. The only waypoints that remain are the ones that represent a change of slope in either of the three directions. To perform this check, the slope in the three

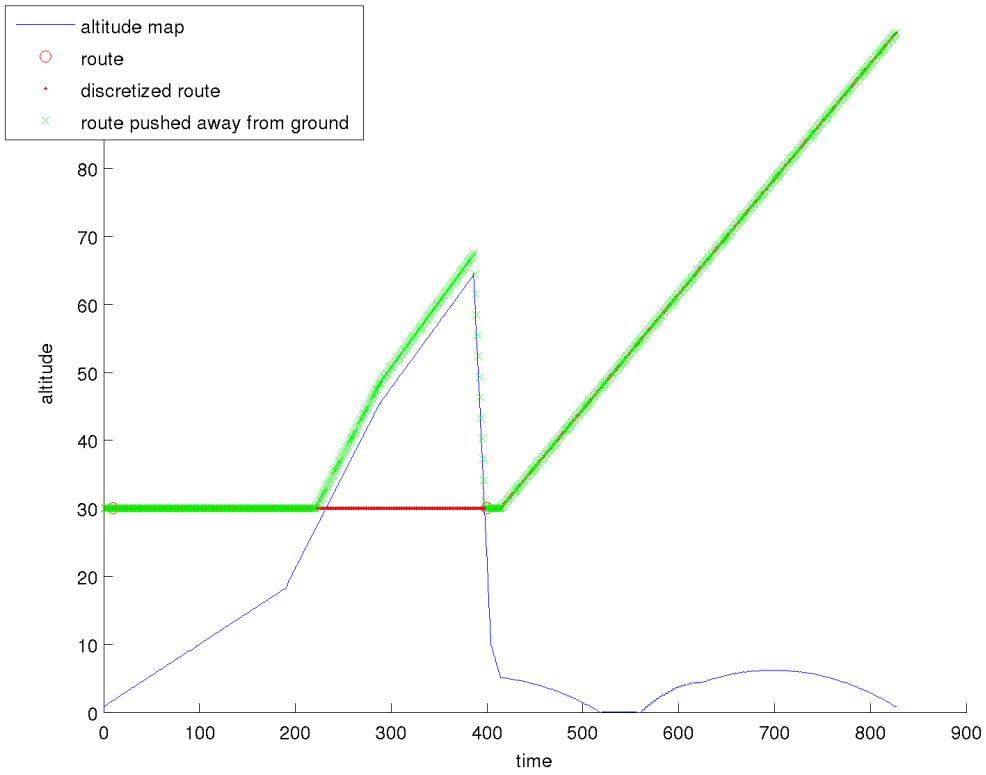


Figure 1.21: Route with corrected flight altitudes

coordinates is computed taking the first two waypoints, and subsequently stored as a reference value. An iterative process is used. The slope is computed with the second and third waypoint and is compared to the reference. If they match the waypoint is discarded, if they don't match the reference is updated. This procedure is repeated until all waypoints are evaluated. An summary of the procedure is shown in Algorithm 2. All the components presented in this subsection act as support for the ATC. The main objective of keeping the vehicles in safe trajectories is achieved trough iterate over each component. Each of this components, provides a validation

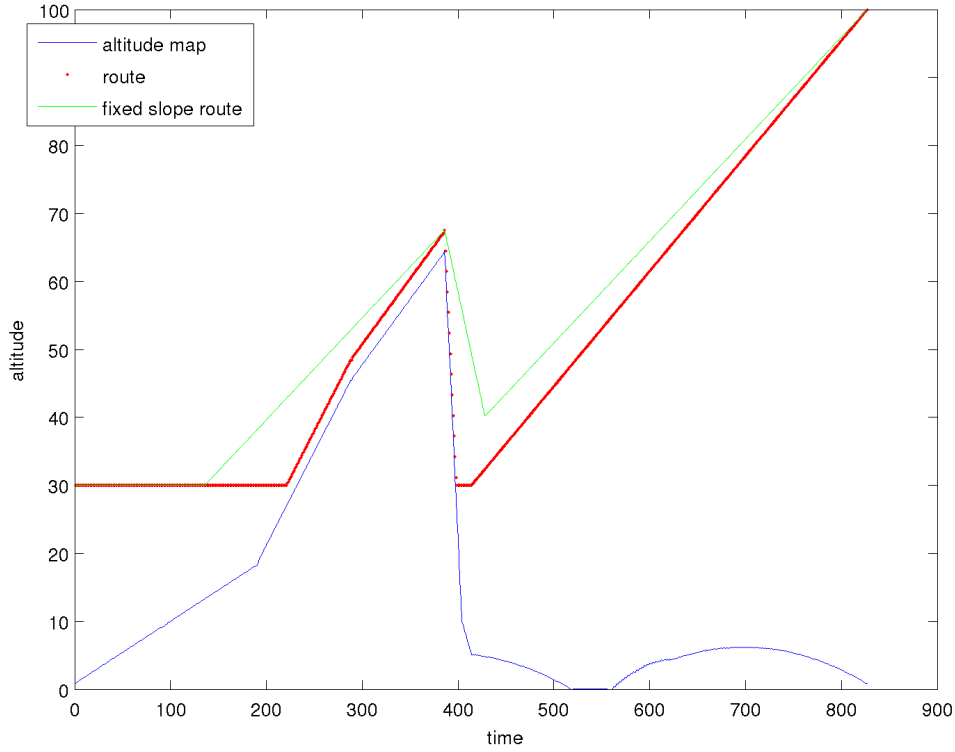


Figure 1.22: Route with corrected flight altitude and slopes

that can be extended to achieve the required constraints.

SENSOR COLLECT CENTER

For a multi-vehicle mission there is the need to process data from multiple sensors and multiple vehicles. With the use of multiple SS all this data can be processed. However, the need for data aggregation arises. A new module is required to collect, manage, and aggregate all the sensor information to reduce the workload of the

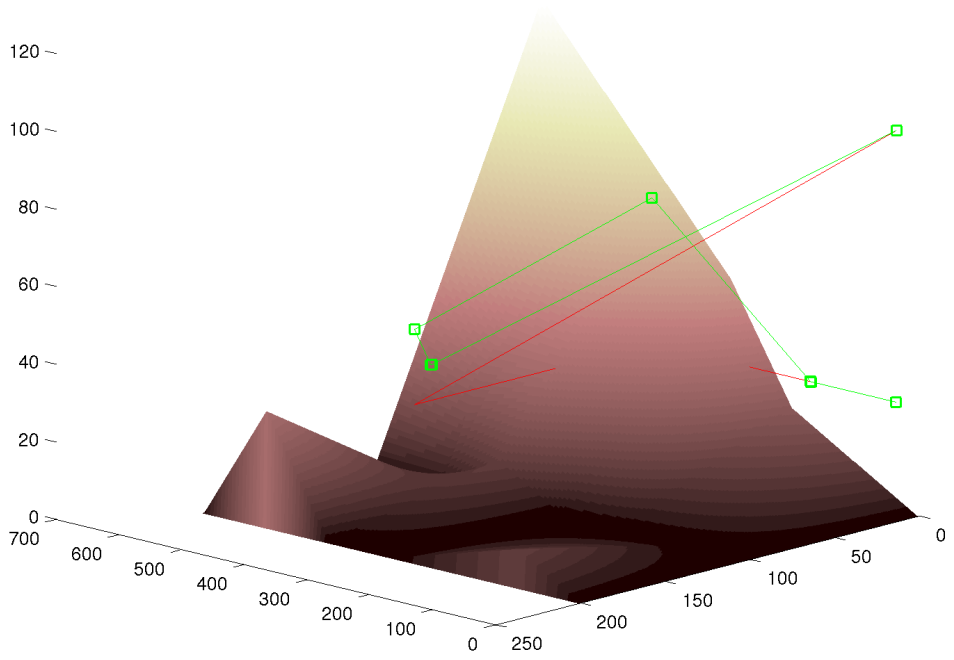


Figure 1.23: Comparison between the received and the corrected route

mission planner–this is the main task of the Sensor Collect Center (SCC). The SCC makes use of the internal protocol to receive different data types, perform required data management, and send it to the MP. To extend the UAV 3D map generation example to a multiple vehicle mission, the SCC is required. The SCC receives the point cloud geo location as generated by the multiple SS, then with multiple frames aggregates the clouds and generates the required map. Next, the SCC stores the generated map in the computer memory. Finally, with the information received, the SCC generates a reduced map which assigns a map generation-quality metric

Algorithm 2 Route compression

```
compute reference slope (wp1 − wp0 / ||wp1 − wp0||2)
for all waypoints in route do
    compute current slope
    if current slope ≠ reference slope then
        set reference slope to current slope
    else
        delete waypoint
    end if
end for
```

at each discrete point, information required by the MP to reroute the vehicles.

REMOTE FLIGHT CENTER

The Remote Flight Center (RFC) is a required module to reduce the dependability of the system on trained pilots. A ground station can be set up to hold a complete set of joysticks and screens to give an immersive, real time experience to a designated operator. The system encapsulates all commands through the internal protocols of the GCS, then communicates to the PRC which in turn controls the vehicle; with this design it is possible to reuse the same setup to control different kinds of vehicles. Further, with a proper design, the system can be reconfigured dynamically to address the needs of the current situation. Clearly, this setup is ideal for supervising the vehicle operation and gaining control in the case of an emergency, or mission failure. The RFC not only enhances the vehicle controllability for the operator, but also eliminates the need for multiple pilots. With the use of the reconfiguration and multi-vehicle capability, with the toggle of a switch the

operator can gain control of any of the vehicles in the air, allowing the simultaneous monitoring of all the vehicles in a mission.

MESSAGE CENTER

Keeping track of all the activity, messages, warnings, or errors in the GCS becomes a hassle for human operators, especially when debugging and developing new actions or missions. Each module has its own particular message types, which vary greatly in importance, ranging from simple information messages to critical warnings or errors related to aircraft airworthiness. The Message Center (MC) is a module designed to manage all the messages and information that needs to be presented to the developers and operators of the GCS. It is a simple yet complete mechanism to manage message queues. Each module dispatches a message with a priority level and a signature. Based on this data, the messages are allocated within the queue. The queue is processed as soon as a message arrives. Whenever a message arrives faster than what can be processed, it is appended to the end of the queue, except when it is deemed more critical than the currently queued messages, in which case it is dispatched first. Since all message are routed through this module the feedback system can be globally updated. For example, when operating outdoors where the sun may impede a clear view of the screen, the module uses speech synthesis to read the messages rather than print them on the screen. The MC centralizes the modules notifications making it easier to monitor the entire system health and reducing the amount of people required to operate the system.

COMMON UNIT

The Common Unit (CU) is intended to be the basic unit used to build all other components by implementing and providing the basic and common functionality required in each module. The CU provides a communication engine that performs a robust message exchange, opens and closes the connections, and automatically reconnects in the case of a connection break. Sharing these modules guarantees that all the modules perform the same hand shake, that the data is packed following same standards, and that the connections are kept open. An entire system upgrade can be made by simply updating this module.

The CU provides all the tools required for fast development such as debugging and data logging. It also handles the message system to communicate with the MC. The basic operation of the CU is shown in Figure 1.24. The diagram shows the operations for the server side and the client side, although they behave in a very similar fashion. The operation of these two submodules is decoupled, hence each submodule can be run by itself, or both can be run at the same time. The operation principle is to wait for a new message until a timeout is reached, then send a heartbeat message to keep the connection open and measure the quality of the network. Whenever a message is received, it is preprocessed to remove the network headings and is added to the queue, then the module is notified. After each message is received, the local queue is processed to dispatch any pending messages.

With the distribution of the the tasks among the modules, the communication intermodule is a key requirement. A robust and reliable communication is a must.

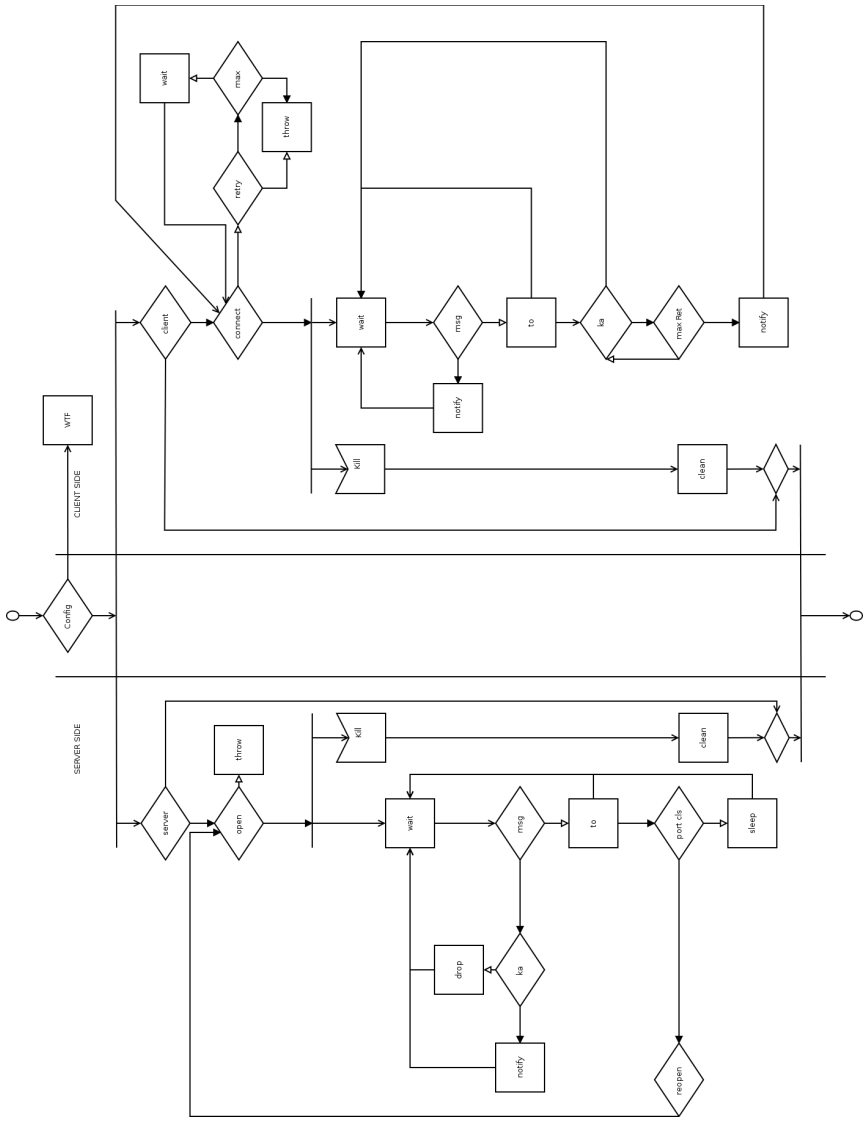


Figure 1.24: Activity diagram for the CU

.

COMMUNICATION

In this section the communication between the different components of the GCS are covered. In Figure 1.25 all participants are shown, where the dashed arrow show which components are directly connected. All connections are bi-directional and the arrow indicates only who opens the connections. If two blocks are not connected then they cannot exchange messages directly, although they can use intermediate components to redirect and forward messages. The pink blocks represents "pure" clients, hence they open all the communication needed. Green represent a pure server–they wait for incoming connections but don't open any remote connections; finally, light blue blocks serve as both server and client at the same time. Although the communication diagram may seem to have a greater impact on the programming elements rather than on the behavior of the GCS, it offers clarity on the system functionality and what are the minimum requisites for operation, and also indicates which of the blocks are critical.

The PRC is the minimum required module and only allows simple behavior;

whenever a more complex mission is implemented, the MP becomes indispensable for the operation. From the diagram, it is clear that whenever the MP is used, the ATC is required as well since it interfaces the MP and PRC; although this might appear as a limitation, there is a special instance of the ATC called bridge that just forwards the message, hence the system does not suffer from a computational overload. With these three modules in place, the system is fully capable and autonomous. If no external sensor is used, the SS is not required. An example of this operation is to test different navigation and control laws, where only position information is required. If the flight is intended to be indoors the need of the RFC may seem redundant, however if the controller fails it is important to be able to regain control of the vehicle. Generally speaking, emergency commands can be issued for different vehicles, where the commands further down the stream take higher precedence, and are hence executed faster. To clarify this behavior, if the PRC issues a land command, this overrides any ATC or MP take off command.

Much like the ATC, the SCC acts as a proxy; hence when a SS is added the SCC also is needed, just to route messages. Finally the IC is only required when interfacing with external developers or operators; if the mission is internal then there is no need of this block.

Clearly from the diagram, it can be seen that the MC is an absolutely optional block since it acts as a message sink. If the mission is stable and the control is correctly implemented then there may be no need of the MC, and it will not used.

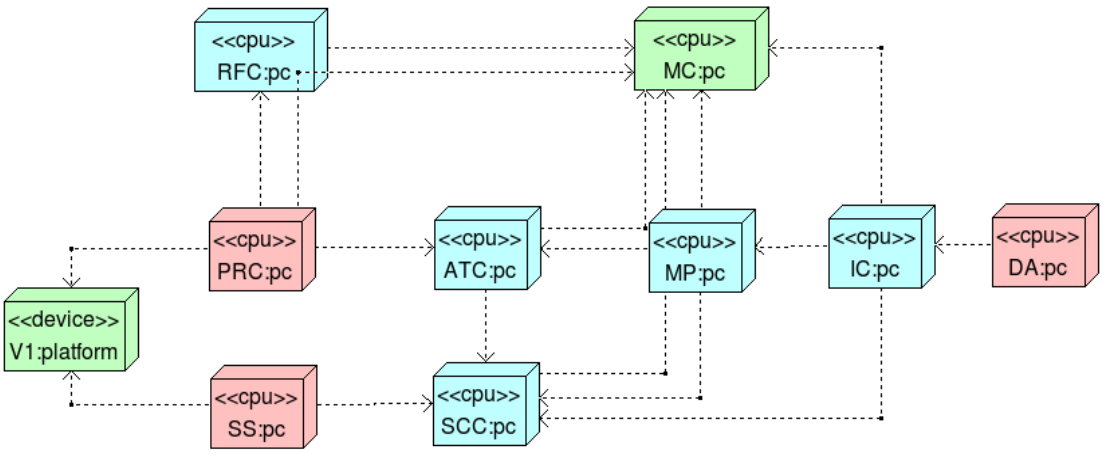


Figure 1.25: Connections layout between GCS members

Chapter 02

MATHEMATICAL MODEL

A smart model is a good model.

Tyra Banks

2.1 INTRODUCTION

Mathematical modeling of dynamical systems is of vital importance for developing controllers and compensators for them. With the advent of digital computers, finite precision implementation of model based controllers have come to prominence. This has particularly affected automatic control system developments for UAVs in particular. Extensive research has been carried out on the effects of controller implementation on digital systems. Knowles et. al.[36] study the error on closed-loop sample-data systems due to finite word length. In their study, an estimate on the drift of the output due to the word length is provided. Hanselmann[37] presents a comparison between the different approaches to realize digital controllers. Bertram shows that the introduction of digital transducers into a sampled-data

system does not result in instability. However it produces an error as a result of the quantization. A bound for this error is also provided.[38] Brubaker and Loendorf derive an expression to estimate the variance of the error at the output due to input data quantization.[39] Astrom presents the theoretical limitations on a control system performance. More specifically, Amstrom rules take measurement noise, actuator saturation, and singularities of the transfer function in the right half plane into consideration.[40] Delchamps proves that an unstable plant, cannot be stabilized after quantization.[41] Miller[42] proves that quantization in LTI SISO systems could led to loss of asymptotic stability of the origin. His results shown that when quantization is taken into account, one only has convergence to a small neighborhood about the origin.

The main focus of the past research was to assess the quality and feasibility of different approaches and platforms to maintain controller efficacy in the presence of quantization, sampling frequency, and hardware constraints such as saturation and measurement noise for linear systems.[39;40] The principal concern regarding quantization levels is that they affect not only the input, but also the internal states and algebraic operations used in the controller. This is because a number is stored in a finite word length variable (fixed bits array size). The effects of finite word length on closed-loop and open-loop problems have been studied in the past due to the limited word length in embedded devices and microcontrollers.[43] The system instability due to quantization can be solved by reducing the quantization level.[42;44] It has been shown by Bicchi that reducing the quantization levels increases the stability of the system and reduces the steady-state error.[45] Bounds for quantiza-

tions errors are established given the quantization levels such as the ones given by Slaughter[46] or Picasso.[47] Roberts[48] shows instability caused due to quantization.

With the development of new digital techniques, new controller designs and new embedded hardware, quantization effects have gathered researchers attention once more.[49;50] The effects of the state and arithmetic quantization are a major concern in the performance of fuzzy logic controllers.[51] These controllers are highly affected by the word length problem,[50] since they are implemented using integrated circuits where the efficiency deteriorates when the word length increases. An example of how digitization affects outputs is shown in a case study presented by Masrur,[52] where a low cost a PWM driver for a brush-less motor used in the automotive industry was developed. In this study different source of error for the PWM driver are discussed. Practical implications and results of different configuration parameters are presented. Distributed systems where multiple components are connected through a data bus with a highly limited bandwidth are also impacted by the quantization quality.[53] In this case, the quantization is done on the bus, so it can be considered as a design parameter[54] or can be modified in real-time based on the system requirements depending on the availability of appropriate hardware components.[55]

Effects of sampling frequency of input and output signals of system digitization have been addressed long ago, and the common assumption is that the sampling frequency needs to be twice as fast than the plant maximum frequency. Based on this assumption, it has been shown that the sampling will not affect the system stability and all information will be available.[56] Methods on how to discretize

continuous time system have been developed based on the sampling frequency.[57] Methods for model reduction have also been presented.[58] These issues have also been explored in the case of a non-constant sampling frequency.[59]

Most quantization problems are not likely to affect the design of certain controllers such as the ones used in Unmanned Air Systems (UAS), where computers with high processing power and large memory banks are popularly implemented. All variables can be treated as floating-point or double precision, which means at least 16 bits are used for each variable. With this state size, it is not likely to suffer from state quantization or algebraic operation quantization errors or saturation. The input quantization with A/D converters which today offers as much as 16 bits, does not drastically impact the flight controller.[38]

The issues described in this section directly impact the digitization of the plant inputs, which drives the digital servo that actuates the control surfaces. A common assumption that the control signal can be freely and instantaneously manipulated. In practice, this is far from the truth. In UAS the preferred method to actuate the control surfaces is to use PWM servos, where commonly it is assumed that there is a linear relation between the duty cycle and the output.[60] Another assumption is that the driver base frequency is much greater than the system frequency. With these assumptions the driver and servo limitations can be neglected. For small aircraft, the servos used have a theoretical base frequency of 1000 hz with a maximum duty cycle of 2 ms which should allow a quantization of 500 levels that can be updated at 500 Hz. However, experience shows that the servo position should not be updated at a higher frequency than 50 Hz, and the quantization levels are by far

much less than the theoretical ones, especially since the servo precision in position will most-often have less than 20 different positions. These servos have a no load time constant of 30 $\frac{deg}{s}$ which dictates that the surfaces dynamics can no longer be neglected. To further illustrate the impact of these limitations on the performance of flight controllers, we now present an example for control of aircraft longitudinal dynamics.

In most geometries and at various flight conditions, it is possible to decouple the aircraft dynamics in two different models. The lateral dynamics and the longitudinal dynamics.[61] For this demonstration, we consider only the linearized longitudinal dynamics which are given by Equations (2.1) and (2.2) where u is the aircraft longitudinal speed, w the aircraft vertical speed, q the pitch velocity, θ is the airplane pitch angle and h the flight altitude. The control action is the elevator deflection given by δ_e. The linearization point is the plane on a level flight at cruising altitude and velocity, with no acceleration. The controller objective is to stabilize the aircraft attitude and control the pitch angle, θ. The plant dynamic was simulated using FlightGear[62] flight simulator, and the controller is implemented on the framework presented on this dissertation (SAI). The controller diagram is shown in Figure 2.1.

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{\theta} \\ \dot{q} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -0.4367 & 0.4431 & -8.3423 & -0.5146 & 0 & 0.0132 \\ -1.0030 & -8.2183 & -4.3260 & 14.9024 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0.0235 & -1.0448 & 0 & -3.9779 & 0 & 0 \\ -0.0213 & 0.9998 & 14.3094 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ \theta \\ q \\ h \end{bmatrix} + \begin{bmatrix} -0.3 & 0 \\ 0 & 0 \\ 0 & 0 \\ -9.2 & 0 \\ 0 & 0 \end{bmatrix} \delta_e \tag{2.1}$$

$$\begin{bmatrix} \theta \\ h \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \end{bmatrix} \tag{2.2}$$

Under the idealized conditions of the control systems operation, the tracking error of the control system can be seen to converge to zero as shown in Figure 2.2. However, when we add the hardware limitations to the simulation the controller performance not only deteriorates, but response becomes oscillatory as shown in Figure 2.3. The failure on attitude tracking is a clear consequence of the elevator control as shown in Figure 2.5. In this figure the effects of saturation, quantization and lag are evident. Finally we present the altitude tracking results in Figure 2.4, where we can see how the overshoot in the real plant is larger. We further note a time delay on the system with hardware constraints in comparison with the idealized control and model.

In the next section a novel approach to deal with hardware constraints is presented. Then this approach is used to design a controller for a single-input single-output plant. A linearized mathematical model of an aircraft is introduced, and all

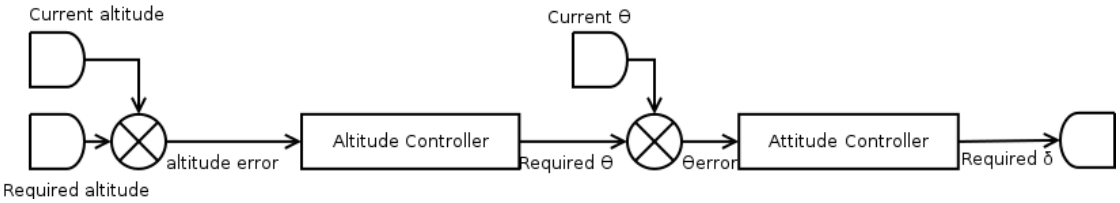


Figure 2.1: Altitude controller layout

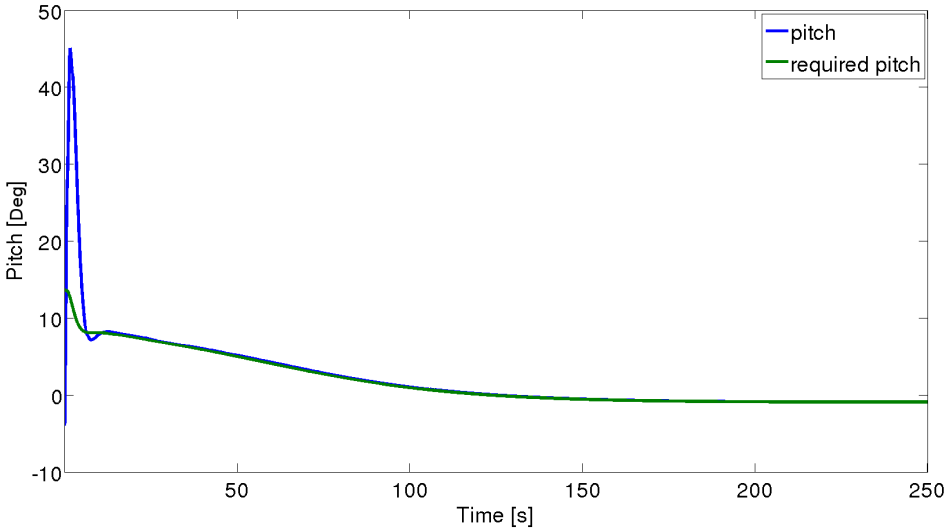


Figure 2.2: Aircraft pitch tracking ideal case

its parameters are described. Finally an LQR approach is used to obtain a state feedback controller for attitude stabilization.

2.2 HARDWARE CONSTRAINTS MODEL

As discussed in the introduction, when the controller is implemented in an embedded system with a microcontroller unit (MCU) due to the hardware-actuator interaction the controller performance deteriorates. When inexpensive MCU with a 8 or 16 bit structure with clock frequencies in the order of the MHz are used,

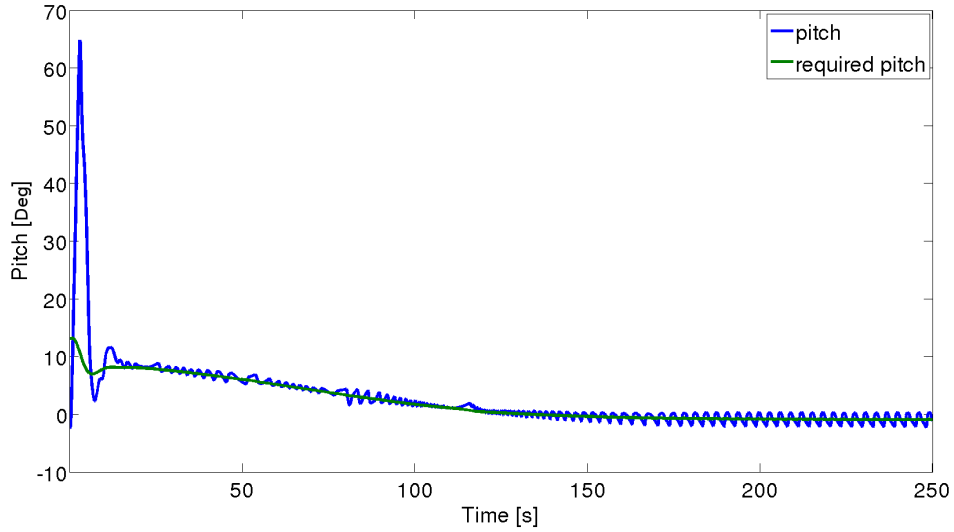


Figure 2.3: Aircraft pitch tracking when hardware constrains are model

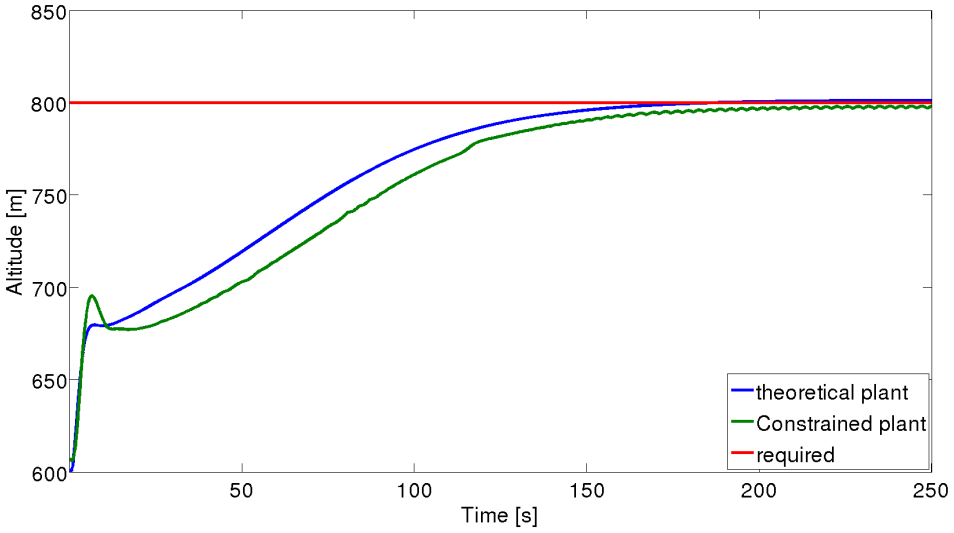


Figure 2.4: Aircraft altitude tracking comparison between ideal case and hardware constrained model

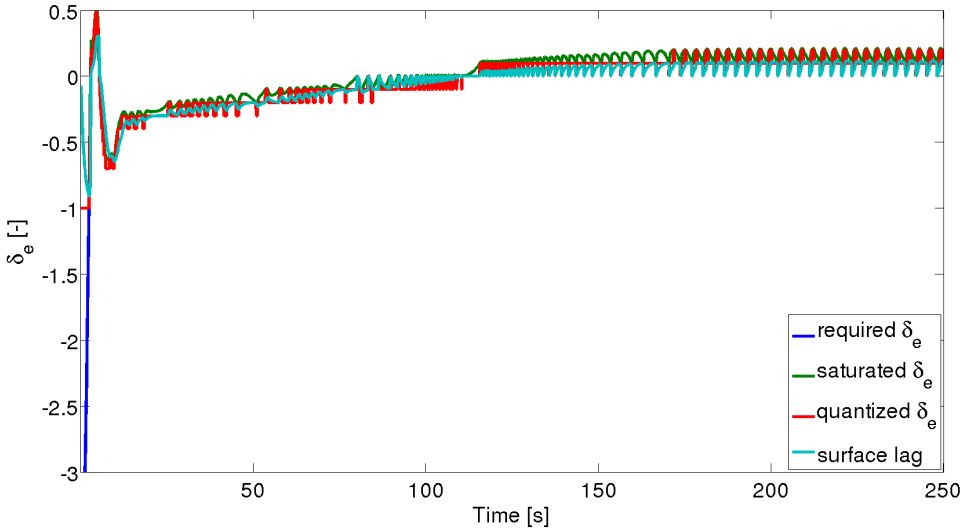


Figure 2.5: Aircraft elevator deflection

the performance decay is more dramatic. In this section, hardware constraints are modeled to achieve better performance. Unlike robust analysis[63] the main goal is to use the modeling tool to take into account the hardware-actuator limitations as part of the controller design rather than leave it for a verification step after the controller is synthesized.

ACTUATOR DELAY MODELING

The classic control problem block diagram with unitary feedback as shown in Figure 2.6 can be modified to take into account the actuator limitations Figure 2.7. Where r is the reference signal, y is the system output and e is the error signal. The controller signal is given by u. The δ symbol represent the error on the quantization.

Let us first focus our efforts into modeling the lag of the servo and the dis-

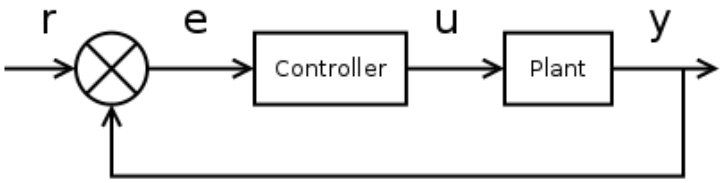


Figure 2.6: Unitary feedback block diagram

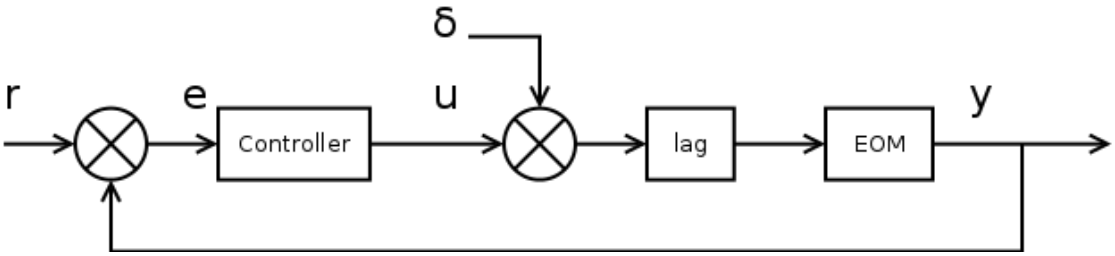


Figure 2.7: proposed system model for input digitalization

cretization on the servo position. The lag on the servo response can be model by a first order transfer function ($G_l(s) = \frac{\tau}{s+\tau}$). The error in the position due to the discretization of the servo can be modeled as a step where the maximum value is given by the discretization level. With this model we can treat a single-input single-output system as a multi-input single-output and use an appropriately suited multivariable control[64] technique to design a controller. Unlike classic multivariable control we do not have the ability to manipulate δ or add a controller on said loop. Only the loop with the reference input can be controlled directly. If we want to separate the problem we can consider each of the outputs to be zero at a time and analyze the remaining loop. In Figure 2.8 we have the reference loop, the only change here is the lag due to the actuator, this lag can be combined with the plant EOM and then the controller can be designed to control this new plant. The sec-

ond loop Figure 2.9 is the disturbance loop, this system has the peculiarity that the input will always be a step, the only concern is to develop a regulator that rejects the steps signal on the input δ. With this two loops we have now a new plant to be controlled under the constrains of the second loop. The multivariable problem is given by the following matrix of transfer functions

$$y = \begin{bmatrix} \frac{G(s)G_c(s)G_l(s)}{1+G(s)G_c(s)G_l(s)} & \frac{G(s)G_l(s)}{1+G(s)G_c(s)G_l(s)} \end{bmatrix} \begin{Bmatrix} r \\ \delta \end{Bmatrix} \tag{2.3}$$

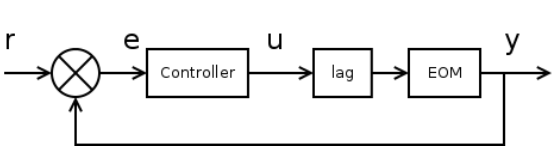


Figure 2.8: plant loop for the proposed model

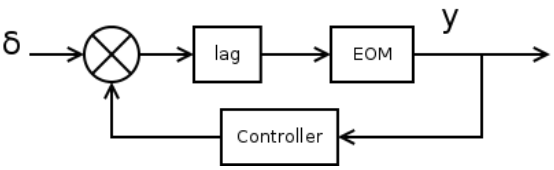


Figure 2.9: disturbance loop for the proposed model

In conclusion, we develop appropriate internal models for the time delays introduced by appropriate actuator elements in the implemented automatic control system design. This is known as the internal model principle in automatic control.[63–65] The additional signals and states introduced by these models are used in the development of the over-all control system, that is rendered multivariable system by the modeling practices instituted by our architecture.

NUMERICAL EXAMPLE

To demonstrate the method stated in the previous section we now show a simple example where a controller is synthesized for an unstable plant given by Equation (2.4).

$$G(s) = \frac{4}{(s-1)(0.02s+1)^2} \tag{2.4}$$

The controller objective is to stabilize the plant. We use a Proportional and Integral (PI) controller, where the gains are computed following the rules elaborated by Ziegler and Nichols.[66] This heuristic method, consists in finding the ultimate gain K_u at which a proportional controller sets the plant output to a constant amplitude oscillation. Then, using the ultimate gain and the period of the oscillations T_u the gains for the PI are given by $K_p = 0.45K_u$ and $K_i = 1.2\frac{K_p}{T_u}$. Applying this rules to the sample plant gives the following controller

$$G_c(s) = k_p(1 + \frac{1}{k_i}s) \tag{2.5}$$

$$K_u = \frac{1}{|G(j\omega)|} = 2.5 \tag{2.6}$$

$$T_u = \frac{2\pi}{\omega_u} \tag{2.7}$$

$$K_p = \frac{K_u}{2.2} \tag{2.8}$$

$$K_i = \frac{T_u}{1.2} \tag{2.9}$$

With a simulation we can verify the design of the controller. Figure 2.10 shows the response of the system for a step input, in the figure are superimpose the response for the ideal plant and for the ideal plant with hardware-actuators constrains and sensor. It can be observed that the controller although meets the design requirements fails to achieve a stable response when the actuator limitations are taken into account.

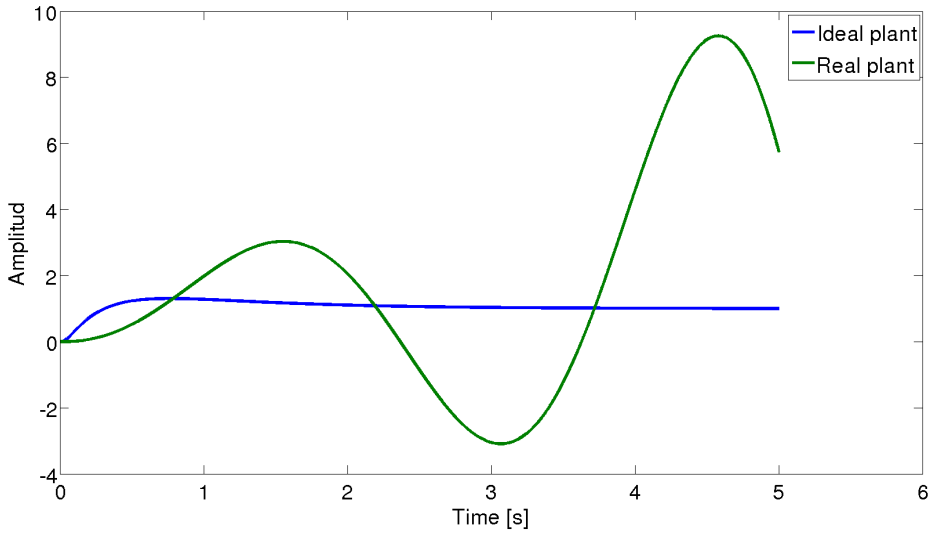


Figure 2.10: unstable plant response to step input

With the new approach presented in the previous section we develop a new controller. First we analyze the digitization loop. It can be shown that for this example if we use an integrator controller the disturbance loop will reject the step input, hence a PID controller given by Equation (2.10) is implemented. The step input response can be analyzed using the final value theorem[67;68] applied to the close loop transfer function given by Equation (2.11). Where all coefficients are given in Equations (2.12) to (2.15)

$$G_c(s) = k_p + \frac{d_i}{s} + k_d s \tag{2.10}$$

$$\frac{y}{\delta} = \frac{G_l(s)G(s)}{1 + G_l(s)G(s)G_c(s)} = \frac{4\tau\alpha s^2 + 4\tau s}{a_0 s^6 + a_1 s^5 + a_2 s^4 + a_3 s^3 + a_4 s^2 + a_5 s + a_6} \tag{2.11}$$

$$a_0 = \alpha, a_1 = (1 + 99\alpha + \alpha\tau), a_2 = (99 + 2400\alpha + \tau + 99\alpha\tau) \tag{2.12}$$

$$a_3 = (2400 - 2500\alpha + 99\tau + 2400\alpha\tau) \tag{2.13}$$

$$a_4 = (-2500 + 2400\tau + 4kd\tau - 2500\alpha\tau + 4kp\alpha\tau) \tag{2.14}$$

$$a_5 = (-2500\tau + 4kp\tau + 4ki\alpha\tau), a_6 = 4ki\tau \tag{2.15}$$

Once more, we use the rules by Ziegler to obtain these gains. The results of the new controller are shown in Figure 2.11 where we can see that the controlled plant with hardware-actuator constrains not only achieve a stable response but also its time response is comparable with the ideal plant with the first PI controller.

Before we can apply this method to design a flight controller, we need to develop the airplane dynamics model. In the next section we present first a non-linear model and then the linearized equivalent model.

ACTUATOR/SENSOR SATURATION MODELING

One key difference between idealized models and system implementations is the saturation of sensors and actuators. Linear control theory considers the control

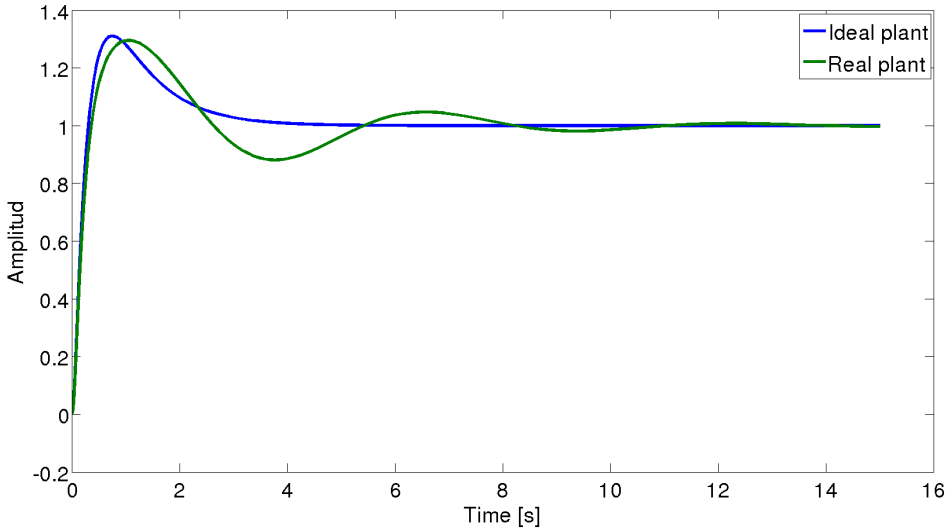


Figure 2.11: unstable plant response to step input with controller designed using the proposed modeling technique

signal to be as large as required to compensate the plant. On physical systems actuators are subject to saturations. More specifically, the deflection of the aircraft control surface are limited to a maximum degree. The motor, as another example, has a maximum thrust that it can provide. Measured quantities also suffer this limitation. The amount of allocated memory and the limitations on the DAQ impose saturation on the quantities sampled. To overcome this limitations, sensors and the DAQ are chosen such as to allow measurements on the entire operation envelop. If the control surfaces are carefully designed, saturation should only occur under extreme conditions. This phenomenon can be modeled with a block which will limit the maximum value of a variable. In this model, this block will limit the maximum and minimum value an internal variable can hold, allowing us to simulate the controller performance under saturation conditions.

2.3 AIRCRAFT DYNAMIC MODEL

The aircraft dynamic model is derived assuming 6 DOF as described by Perkins.[61] The frame used to derived these equations is the body frame and is shown in Figure 2.12. The body frame origin is located on the aircraft CG. The X axis is positive towards the nose of the airplane, the Y axis is positive towards the right hand side of the pilot, and the Z axis completes the frame with the positive direction pointing downwards. The linear velocities in body frame are given by u, v and w. The angular velocity components about each body axis is given by p, q and r. The attitude of the vehicle is described in terms of the Euler Angles.[69] The sequence $1-2-3$ is adopted and the angles are given by roll ϕ, pitch θ and, yaw ψ.

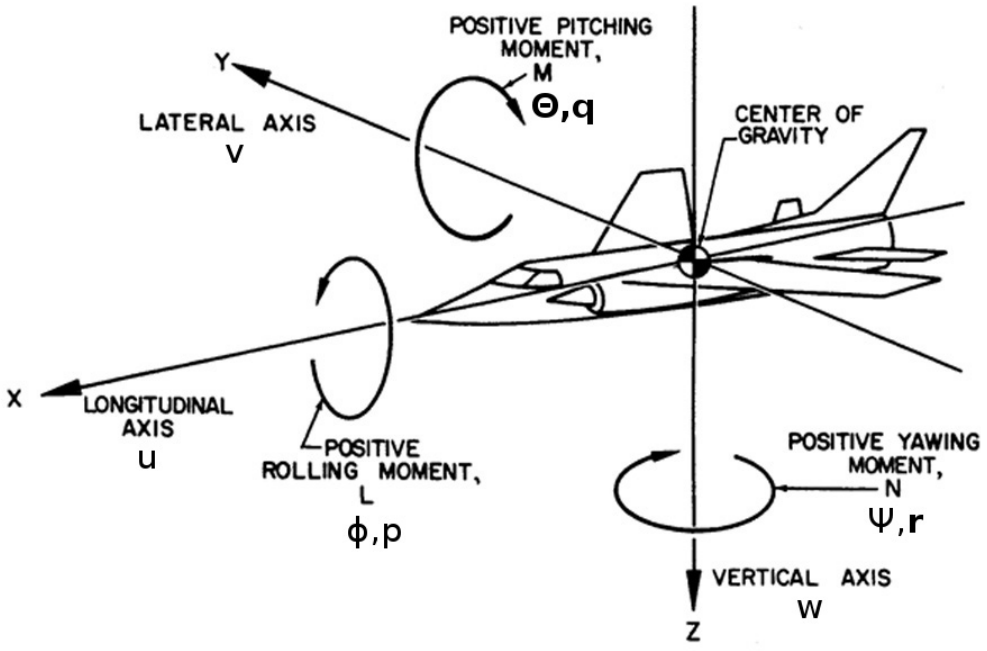


Figure 2.12: Body frame description

Starting from Newton's second law of motion Equation (2.16), and assuming that the aircraft has constant mass, the scalar equations for total forces acting on the aircraft are given by Equations (2.17) to (2.19). (Expressed in body frame)

$$\boldsymbol{\Sigma F} = \frac{d}{dt}(m\boldsymbol{v}) \rightarrow \boldsymbol{F} = m\left.\frac{d\boldsymbol{V}_c}{dt}\right|_B + m(\boldsymbol{\omega} \times \boldsymbol{V}_c) \tag{2.16}$$

$$\Sigma F_x = m(\dot{u} - vr + wq) \tag{2.17}$$

$$\Sigma F_y = m(\dot{v} - wp + ur) \tag{2.18}$$

$$\Sigma F_z = m(\dot{w} - uq + vp) \tag{2.19}$$

Rotational dynamics of the airframe are governed by Euler's equations in Equation (2.20). Assuming that the aircraft has constant mass, the scalar equations for total moment acting on the aircraft are given by Equations (2.21) to (2.23). The body fixed components of the angular momentum vector are given by Equations (2.27) to (2.29).

$$\boldsymbol{\Sigma M} = \frac{d}{dt}\boldsymbol{H} \rightarrow \boldsymbol{M} = \left.\frac{d\boldsymbol{H}}{dt}\right|_B + \boldsymbol{\omega} \times \boldsymbol{H} \tag{2.20}$$

$$\Sigma L = \dot{H}_x - h_y r + H_z q \tag{2.21}$$

$$\Sigma M = \dot{H}_y - h_z p + H_x r \tag{2.22}$$

$$\Sigma N = \dot{H}_Z - h_x q + H_y p \tag{2.23}$$

$$H_x = pI_x - qI_{xy} - rI_{xz} \tag{2.24}$$

$$H_y = -pI_{xy} + qI_y - rI_{yz} \tag{2.25}$$

$$H_z = -pI_{xz} - qI_{yz} + rI_z \tag{2.26}$$

Where I_x, I_y, I_z are the moments of inertia of the aircraft. Typical aircraft has a plane of symmetry (xz plane in the body fixed frame of reference). This symmetry, causes the cross products of inertia I_{yz} and I_{xy} to vanish. The equations that described angular momentum are then reduced to

$$H_x = pI_x - rJ_{xz} \tag{2.27}$$

$$H_y = qI_y \tag{2.28}$$

$$H_z = rI_z - pJ_{xz} \tag{2.29}$$

We can solve from Equations (2.17) to (2.19), (2.21) to (2.23) and (2.27) to (2.29) the time derivatives to obtain the non-linear equations of motion (EOM)

that describes an airplane motion.[70] The forces that act on the vehicle are the thrust, weight and aerodynamics forces and moments. The set of differential equations are shown in Equation (2.30). In the equations m is the aircraft mass, ρ is the air density, v_t is the total velocity given by $\sqrt{u^2+v^2+w^2}$, T is the thrust, S_w is the wing surface, b is the wing span, c is the wing chord, I_p is the propeller inertia. The forces coefficients c_x, c_y, c_z are described in Section 2.3.1.

$$
\begin{aligned}
\dot{u} &= rv - qw + \frac{1}{2}\frac{\rho v_t^2 S_w}{m} c_x - g\sin(\theta) + \frac{T}{m} \\
\dot{v} &= pw - ru + \frac{1}{2}\frac{\rho v_t^2 S_w}{m} c_y + g\cos(\theta)\sin(\phi) \\
\dot{w} &= qu - pv + \frac{1}{2}\frac{\rho v_t^2 S_w}{m} c_z + g\cos(\theta)\cos(\phi) \\
\dot{p} - \frac{I_{xz}}{I_{xx}}\dot{r} &= \frac{1}{2}\frac{\rho v_t^2 S_w b}{I_{xx}} c_L - \frac{I_{zz}-I_{yy}}{I_{xx}} qr + \frac{I_{xz}}{I_{xx}} qp \\
\dot{q} &= \frac{1}{2}\frac{\rho v_t^2 S_w c}{I_{yy}} c_m - \frac{I_{xx}-I_{zz}}{I_{yy}} pr - \frac{I_{xz}}{I_{yy}}(p^2 - r^2) + \frac{I_p}{I_{yy}}\omega_p r \\
\dot{r} - \frac{I_{xz}}{I_{zz}}\dot{p} &= \frac{1}{2}\frac{\rho v_t^2 S_w b}{I_{zz}} c_n - \frac{I_{yy}-I_{xx}}{I_{zz}} pq + -\frac{I_{xz}}{I_{zz}} qr - \frac{I_p}{I_{zz}}\omega_p q
\end{aligned}
\tag{2.30}
$$

The angular rates p, q, and r are the quantities as observed from the airplane. However, the aircraft attitude is described by the Euler Angles. The relation between the Euler angles rates and the angular velocity is given by

$$\dot{\phi} = p + \tan(\theta)(q\sin(\phi) + r\cos(\phi)) \tag{2.31}$$

$$\dot{\theta} = q\cos(\phi) - r\sin(\phi) \tag{2.32}$$

$$\dot{\psi} = \frac{q\sin(\phi) + r\cos(\phi)}{\cos(\theta)} \tag{2.33}$$

If we focus our study on the displacement from equilibrium point we can assume that the angular velocities (p, q, r) are small and their product can be neglected.

The set of equations presented so far describe the airplane motion. Two new equations are added to model the motor dynamics and the vertical speed. The complete state vector is $[u, v, w, p, q, r, \phi, \theta, \psi, h, \omega_p]^T$

AERODYNAMIC FORCE AND MOMENT COEFFICIENTS

The aerodynamic forces and moments acting on the aircraft can be described by a set of non-dimensional coefficients. These coefficients are primarily a function of the angle of attack α and the sideslip angle β defined in Equations (2.34) and (2.35) and shown in Figure 2.13.

$$\alpha = \arctan(\frac{w}{u}) \tag{2.34}$$

$$\beta = \arcsin(\frac{v}{v_t}) \tag{2.35}$$

The aerodynamic coefficients can be obtained by assuming an equilibrium point

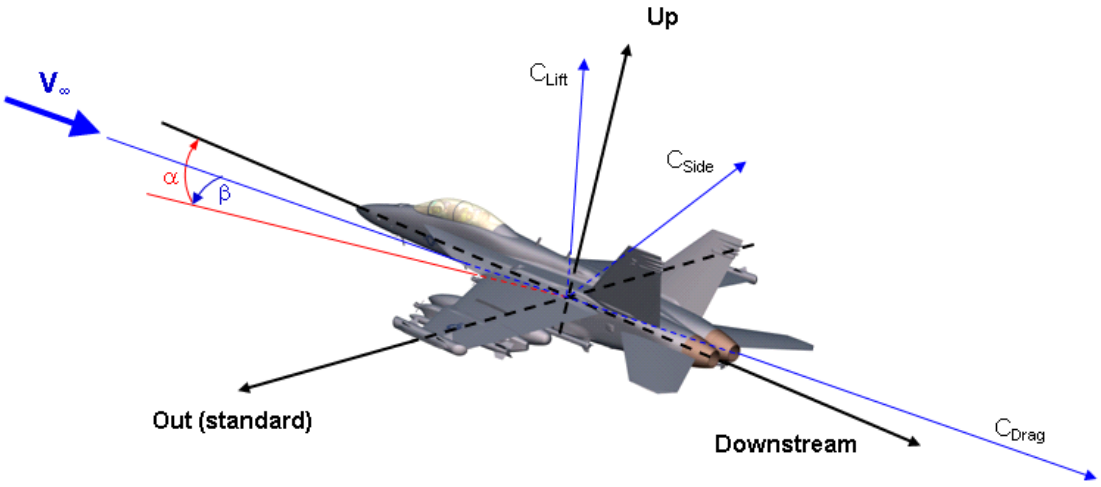


Figure 2.13: Aerodynamic axis

for the aircraft and then, applying small perturbation theory to examine stability of the equilibrium point using Lyapunov's method. This theory assumes that the aerodynamic forces and moments can be expressed as a function of the instantaneous values of the perturbation variables and its time derivatives. Each of the states is replaced by the equilibrium point value plus a small displacement, namely $x = x_0 + \Delta x$. We use a Taylor series expansion of the perturbation variables to represent the aerodynamic forces and moments. Since the perturbations are small higher order terms can be discarded. The derivatives Cx_y described a change in the quantity x due to a change in quantity y. The lift force, which is normal to the wind direction, is proportional to the total lift coefficient C_L as shown in Equation (2.36). C_{L_0} gives the lift coefficient of the aircraft for a 0 angle of attack. C_{L_α} represents the increment in the lift coefficient due to the increment of the angle of attack. $C_{L_{\dot{\alpha}}}$ represents the lift increment due to the rate of change in the angle of

attack. C_{L_q} gives the lift contribution due to pitch rate. The drag force acts in the direction of the wind and the force is proportional to the drag coefficient C_D as presented in Equation (2.37). C_0 represents the parasite drag. $C_{D_{\delta e}}$ represent the increment in drag due to elevator deflection. $C_{D_{\delta a}}$represent the increment in drag due to aileron deflection. The increment in drag due to the increment of lift is given by $\frac{C_L - C_{Lmin}}{\pi e AR}$. The lift and drag coefficients are use to obtain C_X and C_Z as shown in Equations (2.38) and (2.40). The force coefficient in y is given by Equation (2.39). C_{y_β} is the increment in the lateral force due to the sideslip angle. $C_{y_{\delta r}}$ is de increment in the lateral force due to rudder deflection. C_{y_p} represent the increment in lateral force due to roll rate and Cy_r represent the increment in the lateral force due to yaw rate.

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + \frac{c}{2vt}(C_{L_{\dot{\alpha}}}\dot{\alpha} + C_{L_q}q) \tag{2.36}$$

$$C_D = C_{D_0} + C_{D_{\delta e}}\delta e + C_{D_{\delta a}}\delta a + C_{D_{\delta r}}\delta r + \frac{C_L - C_{Lmin}}{\pi e AR} \tag{2.37}$$

$$c_x = C_L \sin(\alpha) - C_d \cos(\alpha) \tag{2.38}$$

$$c_y = C_{y_\beta}\beta + C_{y_{\delta r}}\delta r + \frac{b}{2vt}(C_{y_p}p + Cy_r r) \tag{2.39}$$

$$c_z = -C_L \cos(\alpha) - C_d \sin(\alpha) \tag{2.40}$$

The moment coefficient on the x axis is given by Equation (2.41). $C_{l\beta}$ represents the increment in the roll moment due to the increment of the sideslip angle. $C_{l_{\delta a}}$ Represent the increment on the roll moment due to the deflection of the ailerons. $C_{l\delta r}$ gives the increment of the roll moment due to the defection of the rudder.

C_{l_p} and C_{l_r} give the increment of the roll moment due to the rotational velocities. In Equation (2.42) the y moment coefficients are given. C_{m_0} represents the pitch moment due to aerodynamic forces distribution over the wing. C_{m_α} represents the increment in the pitch moment due to the increment of the angle of attack. $C_{m_{\delta e}}$ Represents the increment in pitch moment due to deflection of the elevator. $C_{m\dot{\alpha}}$ represent the effects of the rate of change of the angle of attack over the pitch moment. C_{m_q} represents the damping of the pitch moment due to the pitch velocity. Finally Equation (2.43) gives the z moment coefficient. C_{n_β} gives the change in z moment due to the sideslip angle. The increment on the moment due to the rudder deflection is given by $C_{n_{\delta r}}$. C_{n_p} and C_{n_r} give the increment of the moment due to the rotational speeds.

$$C_l = C_{l\beta}\beta + C_{l_{\delta a}}\delta a + C_{l\delta r}\delta r + \frac{b}{2v_t}(C_{l_p}p + C_{l_r}r) \tag{2.41}$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta e}}\delta e + \frac{c}{2vt}(C_{m\dot{\alpha}}\dot{\alpha} + C_{m_q}q) \tag{2.42}$$

$$C_n = C_{n_\beta}\beta + C_{n_{\delta r}}\delta r + C_{n_{\delta a}}\delta a + \frac{b}{2v_t}(C_{n_p}p + C_{n_r}r) \tag{2.43}$$

To obtain the aircraft characteristics coefficients experimental procedures are used.[71] The most basic parameters such as mass and inertias are provided by the manufacturer in the user manual.[72] With the use of this parameters the aerodynamic coefficients can be interpolated using experimental data charts[73;74] or estimated using numerical methods.[75] Some popular simulator such as FlightGear[62] or RealFlight[76] have a database of several aircrafts that can be taken as reference.

MOTOR MODEL

For small aircrafts such as the one used on UAVs the motor dynamics have a high impact on the airplane dynamics. The motor not only provides the thrust required but also has a large torque that can not be neglected. Electric motors are gaining popularity due to the simplicity of the operation while keeping a low cost. These motors are brush-less and usually out-runners, which means that the outer case of the motor spins with the propeller. The total inertia of the rotating mass is the summation of the propeller inertia and motor case inertia $I_p = I_{motor} + I_{propeller}$. The differential equation for the motor angular speed is given by Equation (2.44) where T_m is the motor torque and T_p is the propeller torque.

$$\dot{\omega}_p = \frac{T_m - T_p}{I_p} \tag{2.44}$$

The thrust on the vehicle is provided by a propeller attached to the electric motor. The propeller can be characterized using 3 coefficients as described by Merchant.[77] The advance ratio J, Thrust Coefficient C_T, and Power Coefficient C_P. C_p and C_t are parameterized in terms of J. These coefficients are used to obtain the thrust and power for each operation condition. These coefficients can be obtained from the manufacturer manual and from experimental results.[78]

For each flight condition J can be computed using Equation (2.45) where R is the propeller radius and V_t is the air flow. The value of J is used to interpolate the values of C_p and C_t. The propeller thrust is computed using Equation (2.46). The

propeller required torque is computed with Equation (2.47).

$$J = \frac{\pi V_t}{\omega_p R} \tag{2.45}$$

$$C_t = \frac{F_p \pi^2}{4\rho R^4 \omega_p^2} \tag{2.46}$$

$$C_p = \frac{T_p \pi^3}{4\rho R^5 \omega_p^2} \tag{2.47}$$

AIRCRAFT DYNAMIC MODEL TEST

The nonlinear EOM presented in the previous section represent the dynamics for a generic aircraft. To define a custom vehicle a set of coefficients are required. The quality of the modeled time response depends on the quality of the coefficients used to model the forces and moments. For this dissertation we use the Funster V2 ARF[72] from Great Planes, and is shown in Figure 4.15. This aircraft has a wingspan of 1.84 m a wing area of 0.54 m^2. The aerodynamic chord is 0.3 m. The weight of the aircraft with all the sensors and with autopilot installed is 1.34 m. The aerodynamic forces and moments coefficients are taken from RealFlight[76] flight simulator. The coefficients are tuned with a pilot feedback to assure that the plane flight qualities are similar from the real plane. The complete set of coefficients and a Matlab model is given in Appendix A.

To evaluate the model quality we set the initial conditions as required for a level flight condition. The flight level condition is given by $\alpha = 0$, $v = 0$, $w = 0$

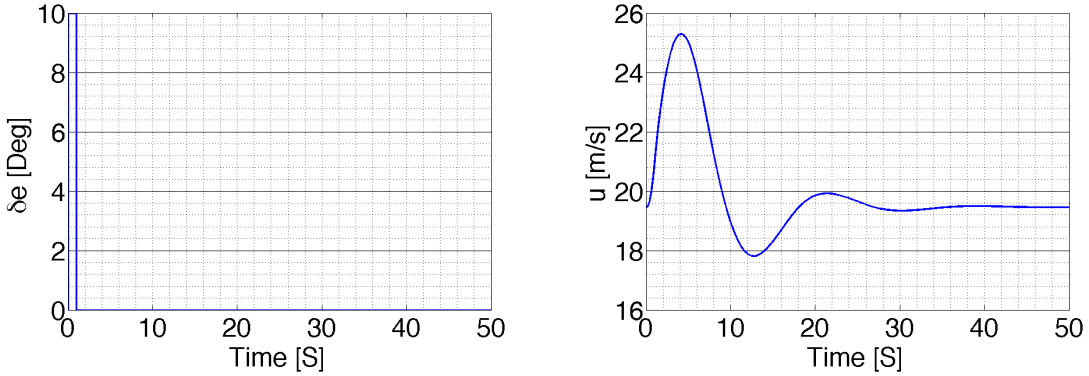
Figure 2.14: Great Planes Funster.

and $u = 19.45\frac{m}{s}$. The aircraft has no rotational speed ($p = q = r = 0$) and the attitude is set to $\theta = 0$,$\phi = 0$, $\psi = 0$. We apply a small perturbation in the elevator Figure 2.15(a) to analyze the time response. We can see in Figure 2.15 the two classic modes for longitudinal dynamics. We observed that the first mode, short period, for our plane is a high frequency mode with high damping. The second mode or phugoid is lightly damped with a very low frequency. The short mode affects mainly the vertical component of the speed w and the angle of attack α. From Figures 2.15(c) and 2.15(e) it can be seen that the phugoid mode has little influence in this quantities. The altitude and the longitudinal speed u are mainly affected by the phugoid mode, where it is clear from Figures 2.15(b) and 2.15(f) that

the quick response from the short period mode has no influence on this quantities. The pitch angle θ is influenced by both modes. The initial response of θ is faster than the ones observed in u or h. Also, the time response takes longer to die out compared to w and α. In Figure 2.16 it can be seen that the the elevator defection have little effect on the lateral dynamics. The eigevalues of the longitudinal mode are given by Equations (2.48) and (2.49). From these eigenvalues, the classical longitudinal modes of an aircraft can be seen. These two modes, are characterized for being both stable. One, with high attenuation and high frequency, and the other with a low attenuation and a low frequency.

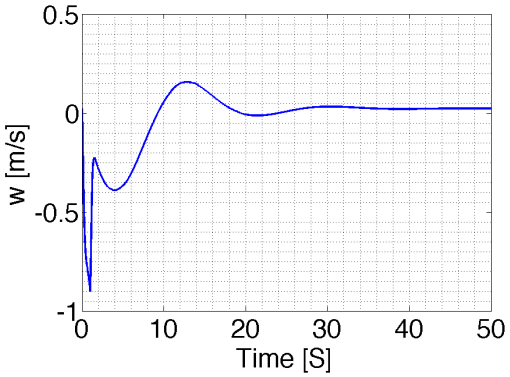
$$\lambda_{1,2} = -6.1375 \pm 3.43i \rightarrow \omega_n = 7.03[\frac{rad}{s}](1.18[Hz]), \zeta = 0.87 \tag{2.48}$$

$$\lambda_{3,4} = -.15 \pm 0.38i \rightarrow \omega_n = 0.412[\frac{rad}{s}](0.06[Hz]), \zeta = 0.36 \tag{2.49}$$

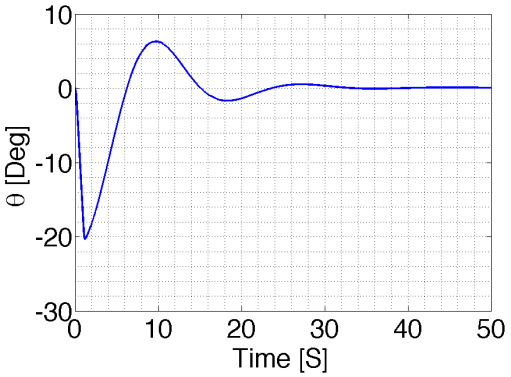


(a) elevator deflection

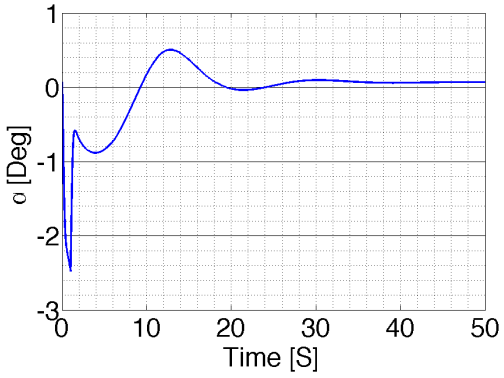
(b) horizontal speed



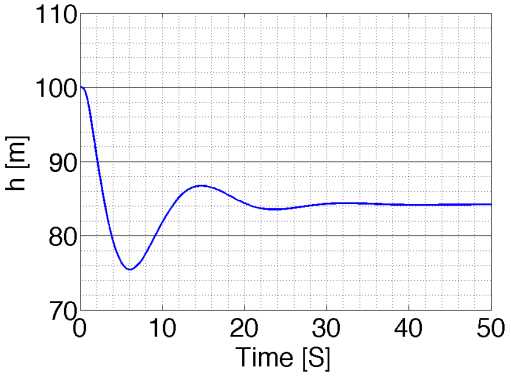
(c) vertical speed



(d) pitch angle

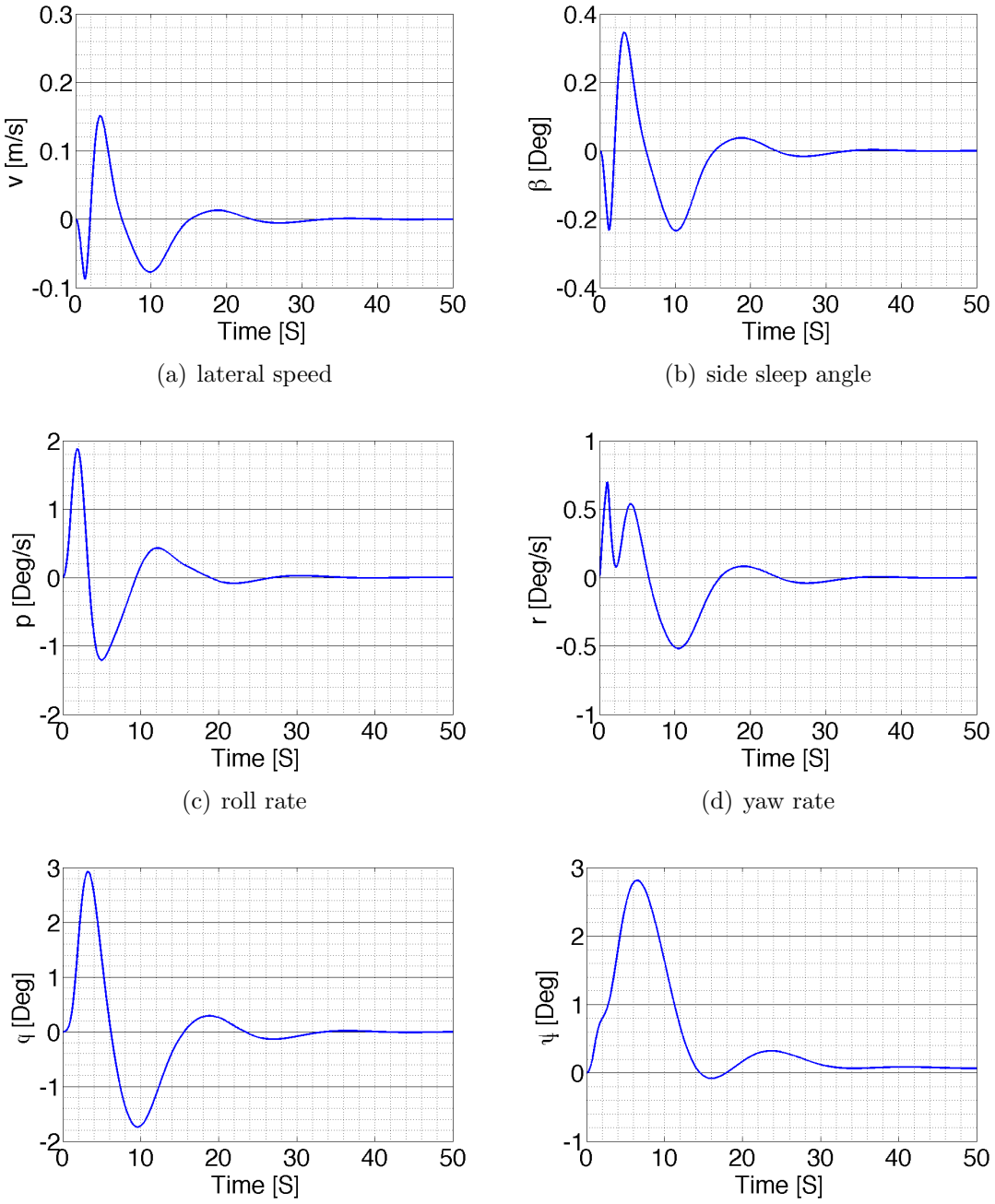


(e) angle of attack



(f) altitude

Figure 2.15: Nonlinear model longitudinal response to impulse input to elevator



(a) lateral speed

(b) side sleep angle

(c) roll rate

(d) yaw rate

(e) roll angle

(f) yaw angle

Figure 2.16: Nonlinear model lateral response to impulse input to elevator

To evaluate the lateral dynamics of the vehicle we set the initial conditions on a equilibrium point and apply a doublet input to the aileron first and then to the rudder Figure 2.17(a). In Figure 2.17 we show the time response for the longitudinal dynamics. Although the deflections of the surfaces are large the longitudinal dynamics remain reasonably constant. The lateral dynamics however Figure 2.18 are highly impacted. It can be seen from the time response the tight-coupling between the lateral and directional dynamics. Although the aileron controls the roll angle Figure 2.18(c) and roll speed Figure 2.18(a) also affects the yaw angle Figure 2.18(d) and yaw speed Figure 2.18(b). The rudder mainly controls the yaw angle and yaw speed but also affects the roll angle and roll rate. This coupling on the lateral and directional dynamics is the main reason why the aileron and rudder need to be synchronized to perform a maneuver. The eigenvalues for the lateral dynamics are given by Equations (2.50) to (2.53). The eigenvalues clearly shown the lateral modes. The slow divergent mode and the oscillatory mode.

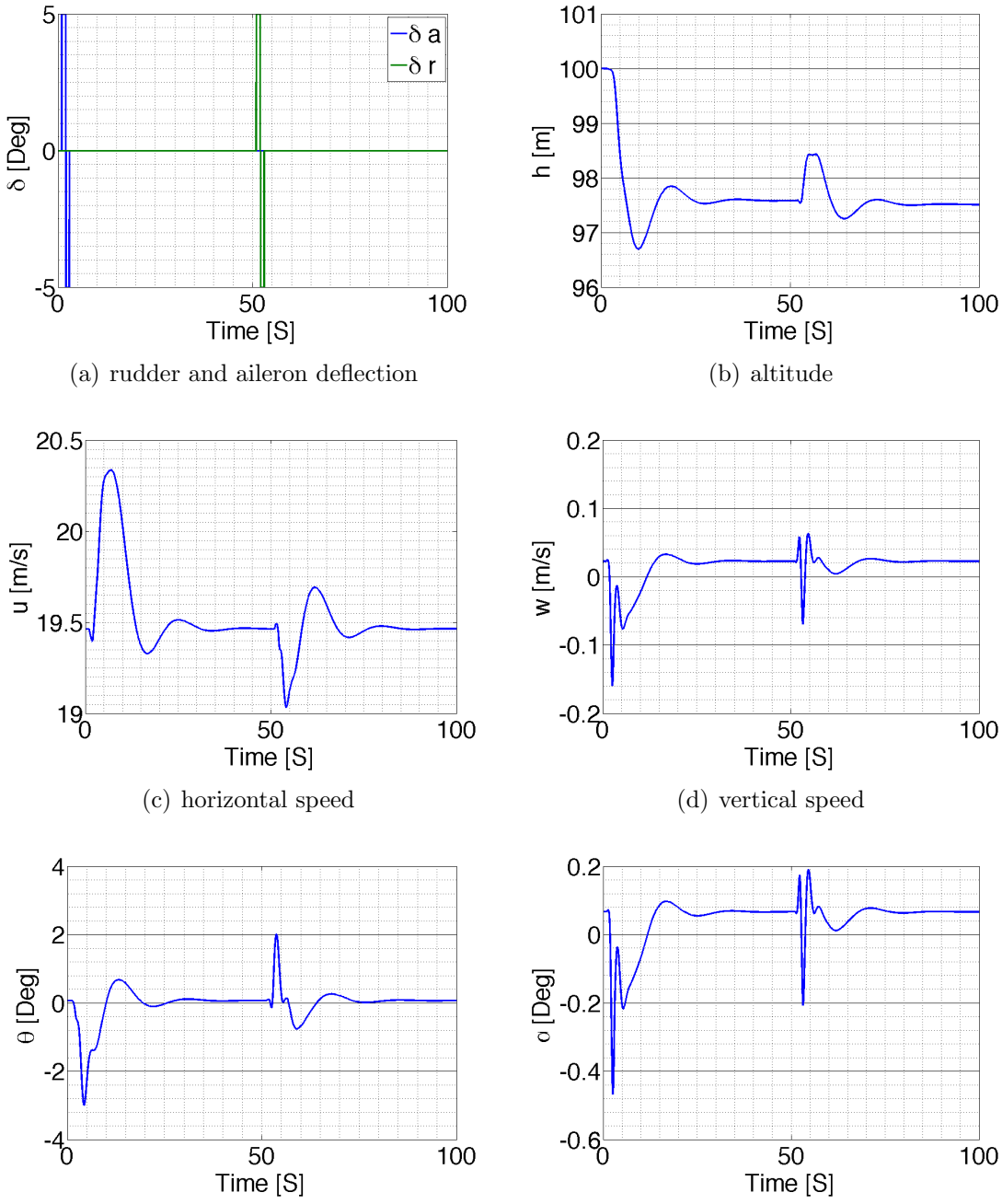
$$\lambda_1 = 2.68 \times 10^{-02} \tag{2.50}$$

$$\lambda_{2,3} = -0.15 \pm 1.97i \tag{2.51}$$

$$\omega_n = 1.98[\frac{rad}{s}](0.31[Hz]), \zeta = 0.76 \tag{2.52}$$

$$\lambda_4 = -2.66 \tag{2.53}$$

$$\tag{2.54}$$



(a) rudder and aileron deflection

(b) altitude

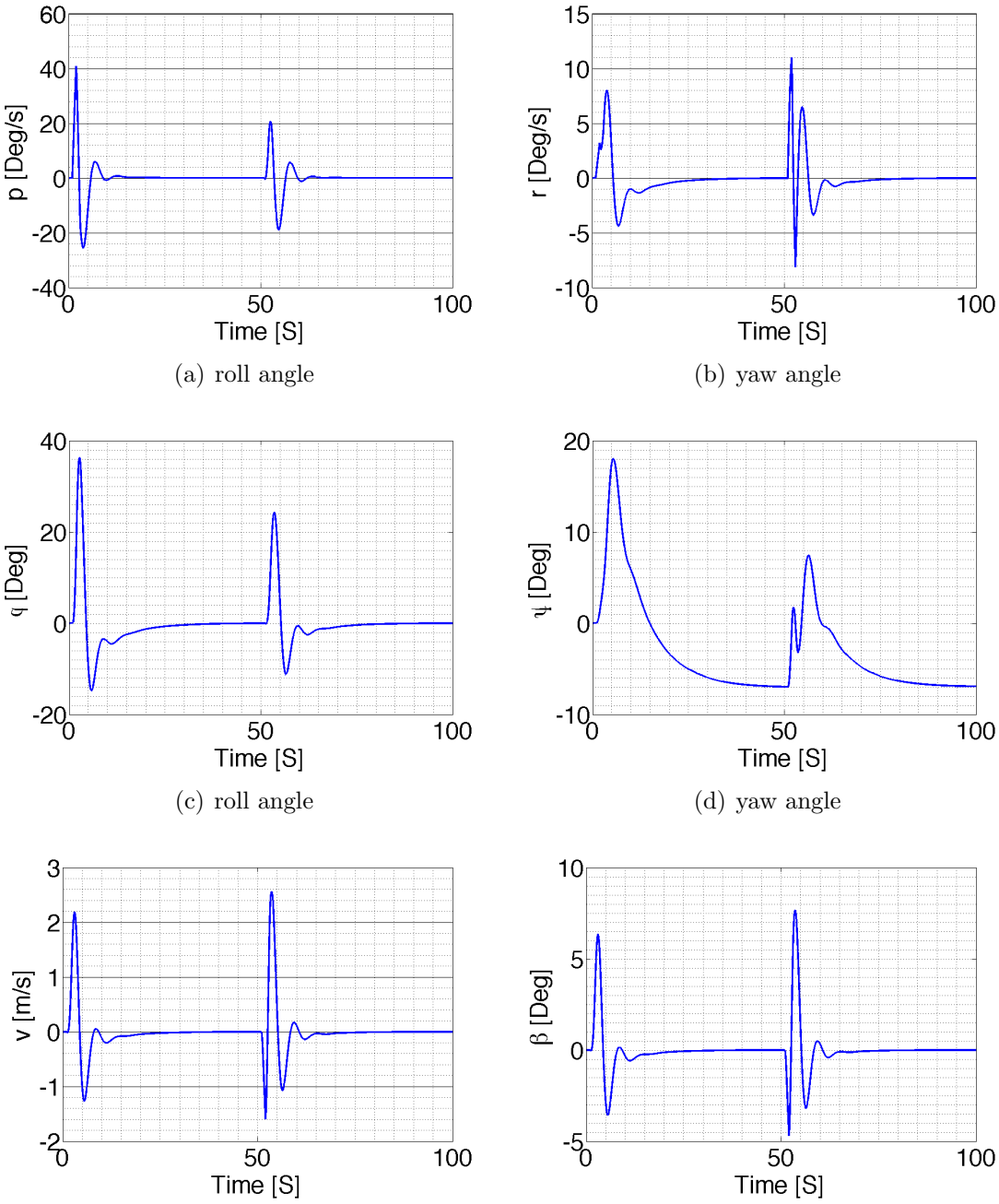
(c) horizontal speed

(d) vertical speed

(e) pitch angle

(f) angle of attack

Figure 2.17: Nonlinear model longitudinal response to doublet aileron and rudder



(a) roll angle

(b) yaw angle

(c) roll angle

(d) yaw angle

(e) lateral speed

(f) side sleep angle

Figure 2.18: Nonlinear model lateral response to doublet aileron and rudder

AIRCRAFT LINEAR MODEL

We propose to utilize the power of linear system theory to develop a controller for the UAV system. To this end, the first step is to linearize the equations of motion of the aircraft. The first step to linearize the equation of motion is to chose an equilibrium point where $\dot{x} = 0$. The second step is to consider small perturbations departures from the main equilibrium point $x = x_0 + \delta x$. For the equilibrium the state-space representation is given by Equation (2.55).

$$\begin{aligned} \boldsymbol{\delta}\dot{x} &= \frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\delta}x + \frac{\partial}{\partial \boldsymbol{u}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\delta}u \\ \boldsymbol{\delta}y &= \frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\delta}x + \frac{\partial}{\partial \boldsymbol{u}}\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\delta}u \end{aligned} \tag{2.55}$$

As was shown in the previous section the longitudinal mode and lateral mode can be decoupled without incurring in much error, this is a common practice.[79] The decoupling of the modes allows as to treat the lateral dynamics and the longitudinal dynamics as separate problems. The longitudinal model as shown in Figure 2.19 is comprised with the states $[u, w, \dot{\theta}, \theta, h, \omega_p]^T$, with inputs $[\delta_e, \delta_t]^T$ and the outputs $[v_t, \alpha, \theta, h]^T$. The lateral model shown in Figure 2.20 and the directional mode shown in Figure 2.21 will be referred as lateral mode or lateral dynamics. The lateral mode is comprised with the states $[v, \dot{\phi}, \dot{\psi}, \phi, \psi,]^T$, the inputs $[\delta_a, \delta_r]^T$ and the outputs $[\beta, \phi, \psi]^T$. The linear model is partitioned as shown in Equation (2.56).

The off-diagonal block matrices are much smaller as opposed to the main diagonal elements. The off-diagonal matrices represent the coupling between the longitudinal and lateral dynamics which are neglected. The small magnitude of these blocks is consistent with the loose coupling between the modes.

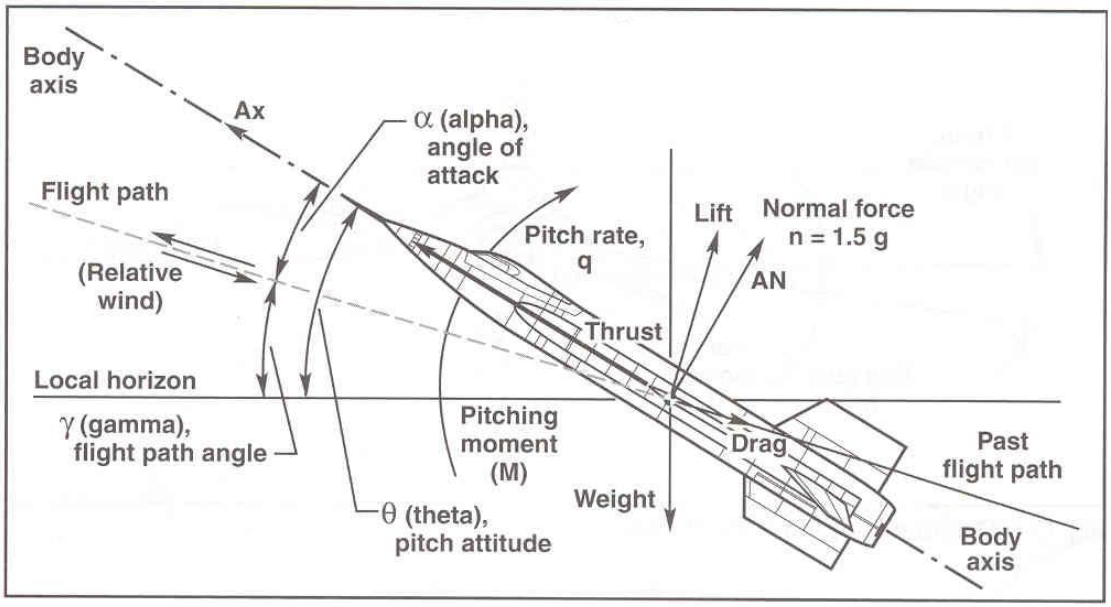


Figure 2.19: Aircraft longitudinal model

$$\begin{bmatrix} \dot{x}_{lon} \\ \hline \dot{x}_{lat} \end{bmatrix} = \left[\begin{array}{c|c} [A_{lon}^{lon}] & [A_{lon}^{lat}] \\ \hline [A_{lat}^{lon}] & [A_{lat}^{lat}] \end{array}\right] \begin{bmatrix} x_{lon} \\ \hline x_{lat} \end{bmatrix} + \left[\begin{array}{c|c} [B_{lon}^{lon}] & [B_{lon}^{lat}] \\ \hline [B_{lat}^{lon}] & [B_{lat}^{lat}] \end{array}\right] \begin{bmatrix} u_{lon} \\ \hline u_{lat} \end{bmatrix}$$
$$\begin{bmatrix} y_{lon} \\ \hline y_{lat} \end{bmatrix} = \left[\begin{array}{c|c} [C_{lon}^{lon}] & [C_{lon}^{lat}] \\ \hline [C_{lat}^{lon}] & [C_{lat}^{lat}] \end{array}\right] \begin{bmatrix} x_{lon} \\ \hline x_{lat} \end{bmatrix} + \left[\begin{array}{c|c} [D_{lon}^{lon}] & [D_{lon}^{lat}] \\ \hline [D_{lat}^{lon}] & [D_{lat}^{lat}] \end{array}\right] \begin{bmatrix} u_{lon} \\ \hline u_{lat} \end{bmatrix} \tag{2.56}$$

The linearization point, and all the matrices are provided in Appendix A. We

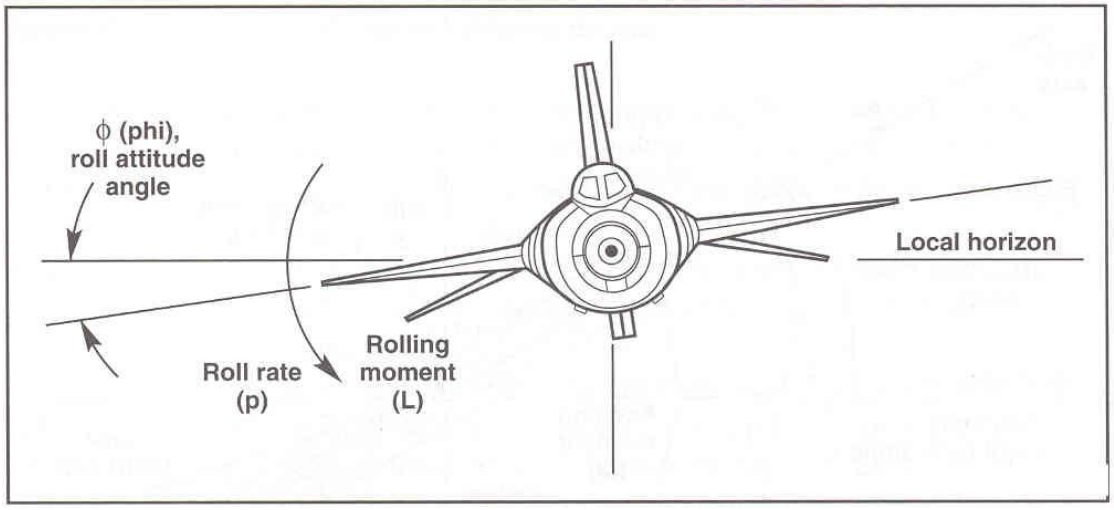


Figure 2.20: Aircraft lateral model

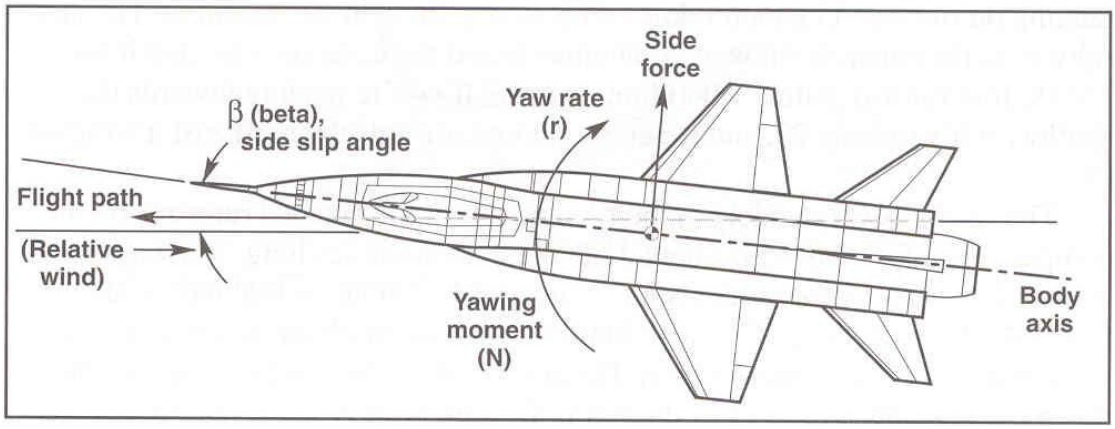
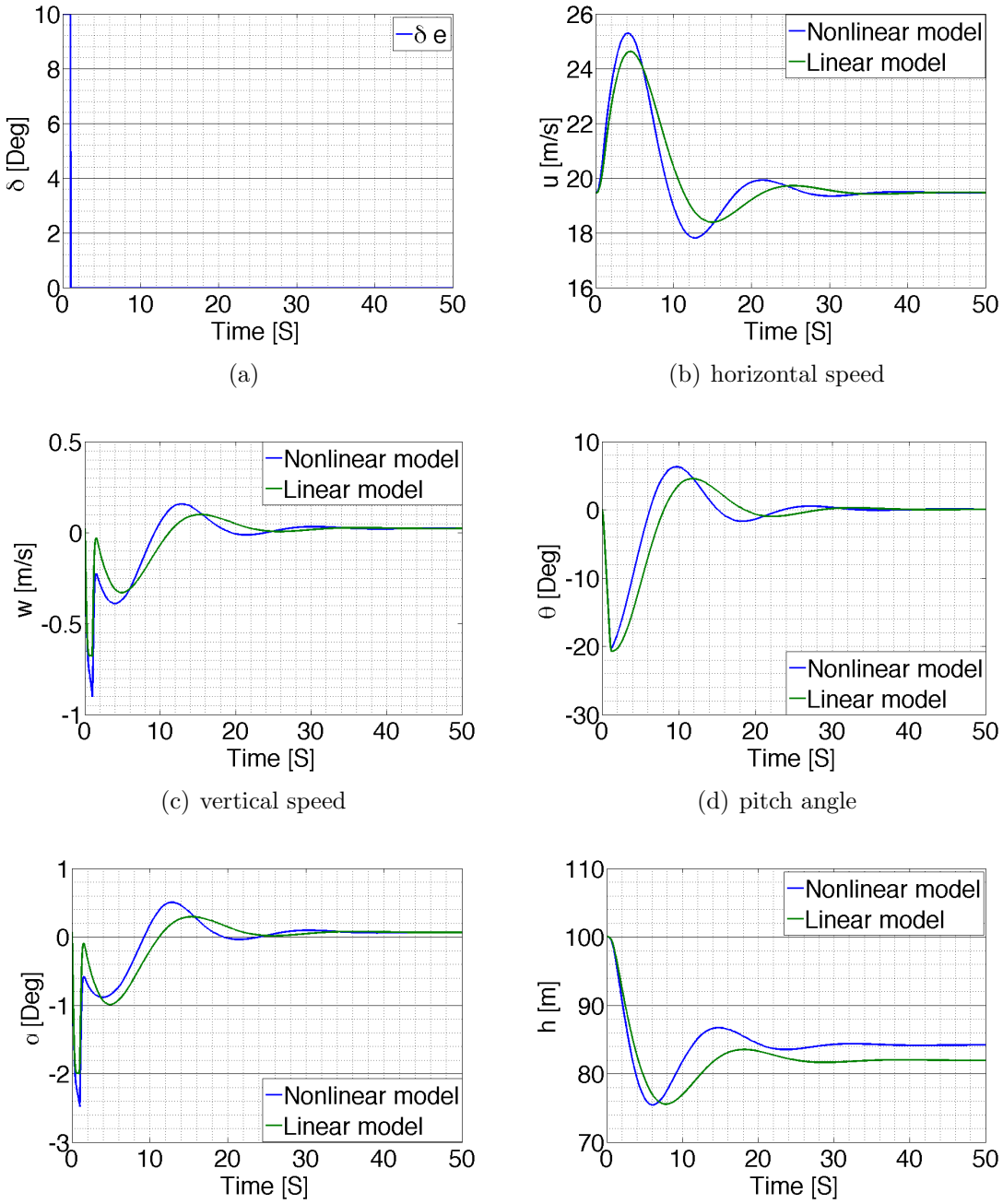


Figure 2.21: Aircraft directional model

perform a test to compare and validate the linear decoupled models. We set the vehicle in the equilibrium point and apply the same perturbations to the linear models and the nonlinear model. The inputs and outputs are shown in Figures 2.22 and 2.23. We can see in Figures 2.22 and 2.23 that the error incurred by the variable separation is small enough to neglect the effects. These decoupled models are used

to design the controllers, however before flying them we test the output in the simulator with the full non-linear model.

We can see from the response comparison that the linearization model captures both longitudinal modes. The linearized pughoid mode its slightly slower as opposed to the non linear model. The damping its observed to be on the same range. This frequency discrepancy should be taken into account when designing the controller. The real plant to compensate will have faster dynamics than the one used on the controller design. The short period mode, as observed from the angle of attack response in Figure 2.22(e) its captured in a more satisfying way.



(a)

(b) horizontal speed

(c) vertical speed

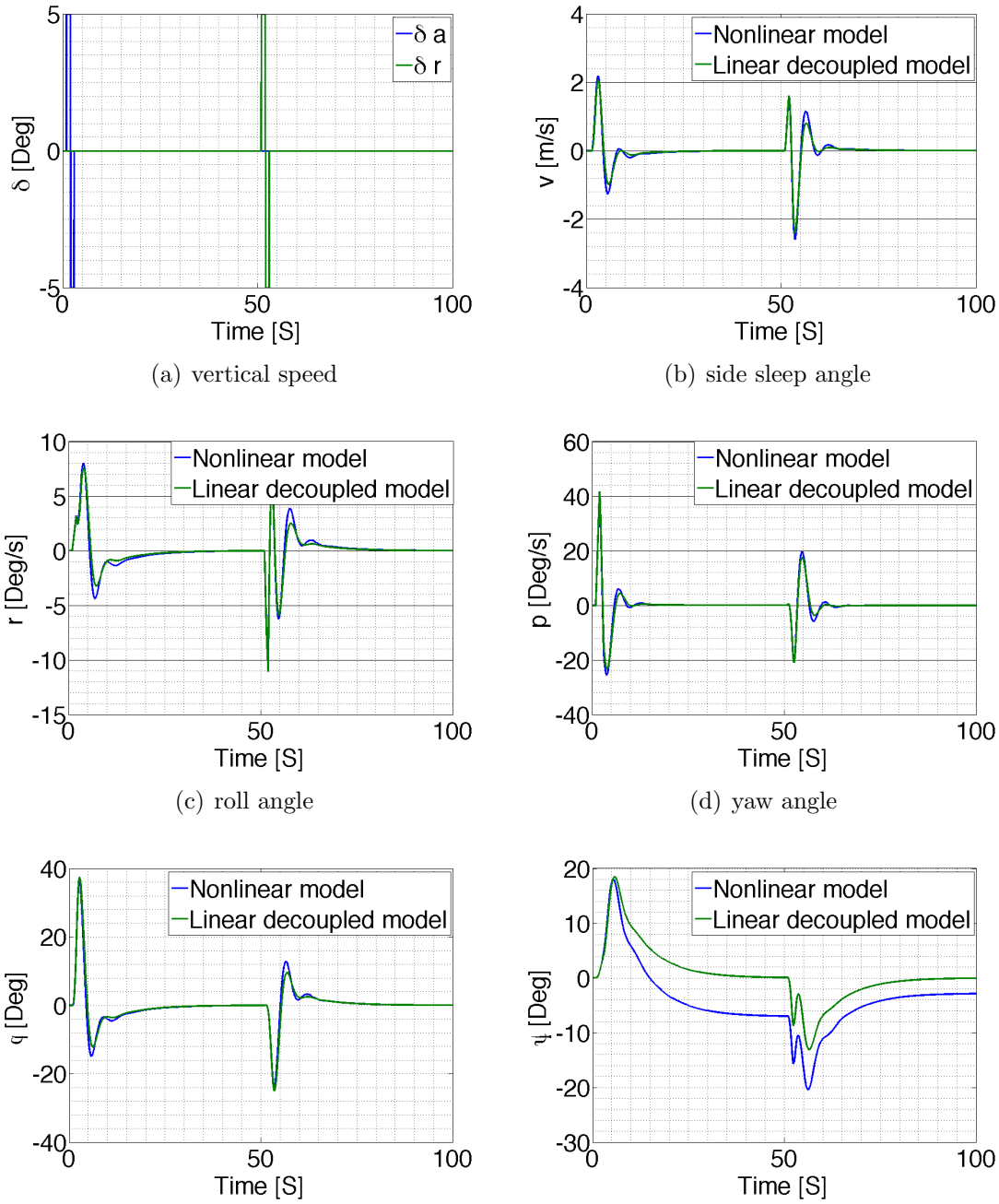
(d) pitch angle

(e) angle of attack

(f) altitude

Figure 2.22: Nonlinear and Linear decoupled model response comparison to impulse input to elevator

To validate the lateral model we repeat the procedure applying a doublet input first to the aileron and then to the rudder. From the time response Figure 2.23 it can be seen that the main frequencies have been completely captured. The Magnitude of the perturbations have been properly modeled with the linearization. However, it can be observed that the yaw angle magnitude its being overestimated by the linear model. This discrepancy in the models should be taken into account when designing the flight controller.



(a) vertical speed

(b) side sleep angle

(c) roll angle

(d) yaw angle

(e) roll angle

(f) yaw angle

Figure 2.23: Nonlinear and Linear decoupled model response comparison to doublet to elevator and rudder

2.4 AIRCRAFT CONTROLLER

The linearized model for the UAV is taken in this section to design the required flight controller. As was discussed, the lateral dynamics and the longitudinal dynamics can be decoupled without incurring in large errors. Two controllers are design in this section. The first controller is used to control the longitudinal dynamics and the second is for the lateral and directional dynamics. The longitudinal controller use the elevator to control the pitch angle. The pitch angle can be used to change the aircraft altitude. The throttle is controlled to maintain the total velocity of the plane constant.

The lateral controller is used to track waypoints. When a new waypoint is received a path planner computes the desired flight direction. This direction is then forward to the controllers who sets the desired attitude. Once the aircraft is following the right direction the controller maintains a stabilized flight, namely roll zero.

The controller structure is as shown in Figure 2.24. The longitudinal stabilization loops takes as input pitch angle, pitch rate, vertical velocity and horizontal velocity. The control signals are elevator and throttle. The output is required pitch and flight velocity.

The lateral controller takes as input roll, yaw, roll rate and yaw rate. The controller, controls aileron and rudders to achieve desired roll and yaw angles. In the next subsection the longitudinal controller is first discussed in more detail then

the lateral controller.

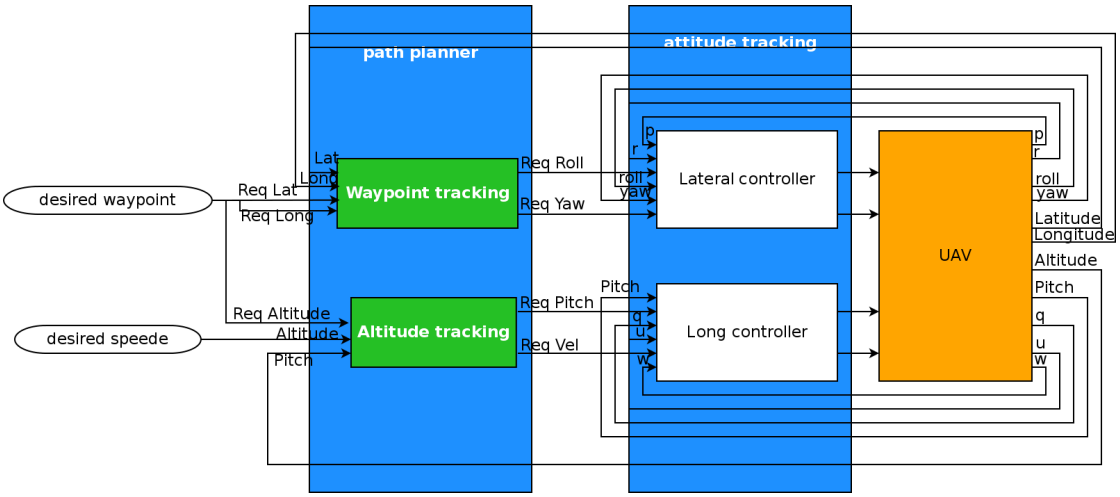


Figure 2.24: UAV controller structure

AIRCRAFT LONGITUDINAL CONTROLLER

The objective of the longitudinal controller is to achieve a straight fly and change altitude when required. Taking as input a reference pitch θ_r and total velocity v_r the controller actuates the elevator δ_e and throttle δ_t to achieve the required value.

Before any controller can be designed an indepth analysis is required. First, we need to establish if the plant can be controlled and second, we have to establish which quantities are available to be use on the feedback controller. To establish the feasibility of the design we compute the observability and controllability gramians.[65] These gramians are used to evaluate the plant capabilities and are defined as shown in Equations (2.57) and (2.58). Since the LTI model is used for the controller design we compute the controllability and observability matrices Equations (2.59)

and (2.60). These matrices should be full rank in order for the system to be fully observable and controllable.[68] For the aircraft linearized model both matrices have a rank 6 hence the system is fully controllable and observable. The detailed matrices are shown in Appendix A.

$$w_c(0,tf) = \int_0^{tf} e^{-At} B B^T e^{-A^T t} dt \tag{2.57}$$

$$w_o(0,tf) = \int_0^{tf} e^{-A^T t} C^T C e^{-At} dt \tag{2.58}$$

$$C = \begin{Bmatrix} B & AB & \dots & A^{n-1}B \end{Bmatrix} \tag{2.59}$$

$$O = \begin{Bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{Bmatrix} \tag{2.60}$$

A popular controller technique is the state feedback Figure 2.25. For this technique is assumed that all states are known and a feedback gain K is used to close the loop. The main advantage of state feedback is the vast methods to obtain the feedback gain K and the capabilities for MIMO systems. Since the dynamic response achieve could be as expected but the steady state error may be to large a pre gain $\bar{N}$ is used to scale the inputs.

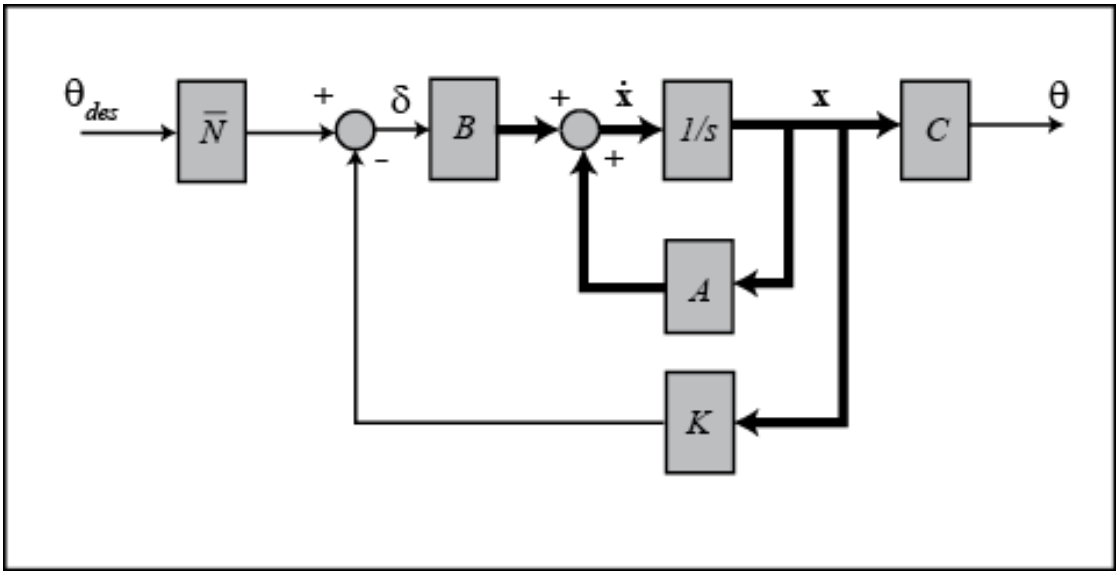


Figure 2.25: State feedback controller

The Linear Quadratic Regulator[80] provides a systematic approach to obtain the feedback matrix K such that minimize the functional cost J defined in Equation (2.61). The matrix weights Q and R provide means to tune the solution. The matrix Q assigns weight to the states that are more critical. The matrix R assigns weights to each control input. The gain K that minimize the functional depends of the system and the gain matrices. Fine tunning these weight matrix is not trivial, and the designer needs to have some sens of the problem to get initial values. A good starting point is to use the identity matrix with proper dimensions for R. A shape of Q can be obtained using C^TC which yields a matrix that have zeros for the states that do not affect directly the outputs of the system.

$$J = \int_0^\infty (x^T Q X + u^T R u) dt \tag{2.61}$$

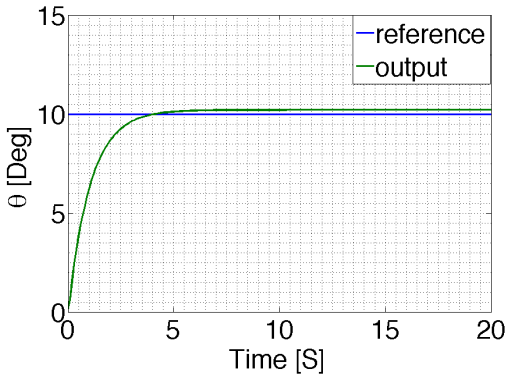
For the longitudinal controller we remove the altitude state since it is strongly related to the pitch angle. This reduced model is used to design the LQR controller. The matrix gain then needs to be padded with zeros on the state that represent the altitude.

The elevator is used to control the pitch angle. The required response should be fast but without overshoot or oscillations. More specifically the requirements for the controller are: overshoot smaller than 10%; small oscillations, which can be expressed as $1 > \zeta > 0.7$; and a fast response with a rise time of at most $2S$. The other states should behave such that allows an altitude hold type of flight. To test the controller the system is excited with a step on the required pitch. The output for the controlled system is then as shown in Figure 2.26 and the control variables are shown in Figure 2.27. The Q, R, K and $\bar{N}$ matrix are shown in Appendix A.

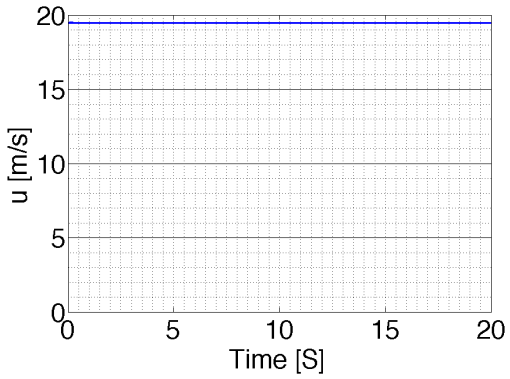
The results shows clearly that the system performs as expected. To test the nonlinear model the simulator is required. The results are shown in Section 3.5

AIRCRAFT LATERAL CONTROLLER

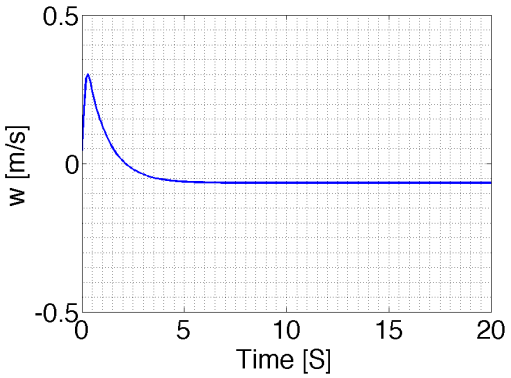
For the lateral controller we repeat the procedure from the previous section. The reduced model and the gramians are shown in Appendix A.0.2. The objective for this controller is to use the rudder to track a desired yaw angle and the ailerons to track the desired roll angle. This surfaces with out control are related to the rates rather than the angles. The results of the controller tracking and the control signals are shown in Figure 2.28. From the control signals it can be seen that once



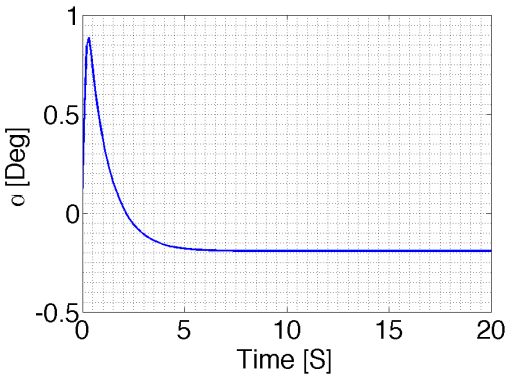
(a) pitch angle



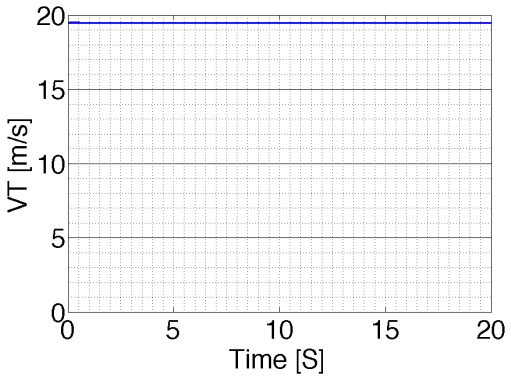
(b) horizontal speed



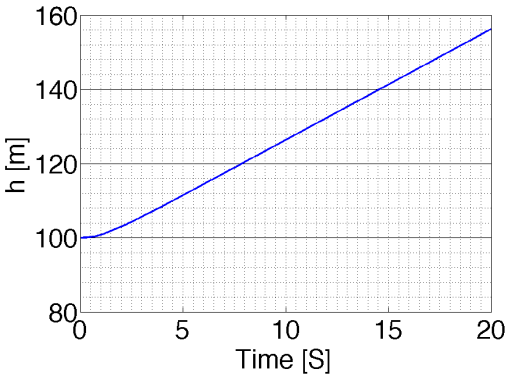
(c) vertical speed



(d) angle of attack

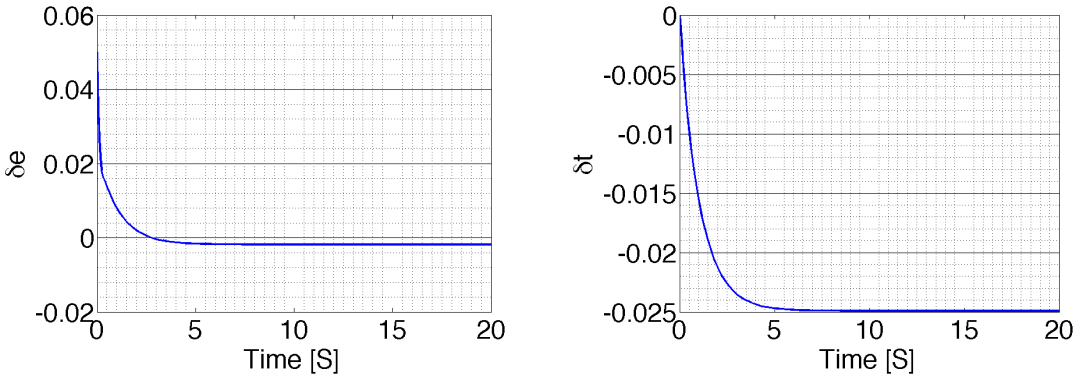


(e) air velocity



(f) altitude

Figure 2.26: Controlled linear model response to step input on θ



(a) elevator deflection

(b) throttle

Figure 2.27: Linear system control variables

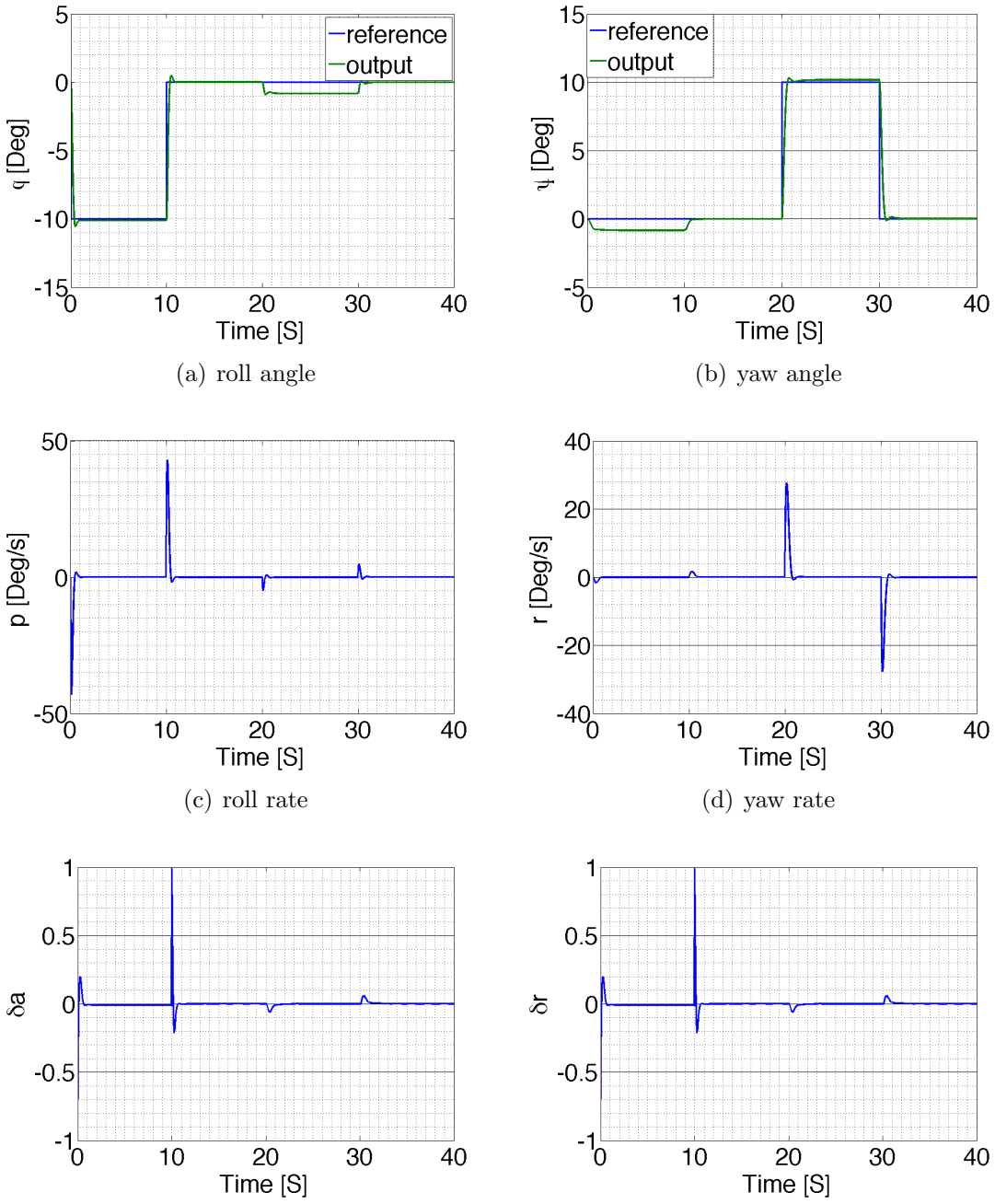
the desired angle is reached the surfaces no longer needs to be deflected.

The results shows clearly that the system performs as expected. To test the nonlinear model the simulator is required. The results are shown in Section 3.5

2.5 CONCLUSIONS

In this chapter we have presented a summary of the hardware constrains that a designer may find when implementing a controller for dynamic systems with inexpensive off-the-shelf sensors and actuators. The effects of said constraints on UAS controllers are discussed. In Section 2.2 we present an approach to deal with the digitization of the plant inputs and an example for a SISO plant is given.

Finally, we use the proposed approach to design a controller for a MIMO plant and HIL simulations are carried out to validate the method. The method discussed in this paper provides an approach to model the plant input uncertainty which allows the designer to develop robust controllers.



(a) roll angle

(b) yaw angle

(c) roll rate

(d) yaw rate

(e) aileron deflection

(f) rudder dflection

Figure 2.28: Controlled linear model response to double step input on ruder and yaw

A nonlinear mathematical model of an aircraft is introduced, and all its parameters are described. The linearization and separation of the longitudinal dynamic and lateral dynamics are discussed. Finally a LQR approach is used to obtain a state feedback controller for attitude tracking. In the following chapter, the Hardware-In-The-Loop simulation is introduced. This simulation, is used to further evaluate the controller with a nonlinear model.

Chapter 03

HARDWARE-IN-THE-LOOP SIMULATION

> We live in a world where there is more and more information, and less and less meaning.
>
> Jean Baudrillard, Simulacra and Simulation

3.1 INTRODUCTION

Aerospace engineering has a long history of design and development of detailed simulation capabilities that can be traced all the way back to the Apollo missions. Flight simulators have been the key elements in the astronaut training programs[81] in the early phases of Project Mercury through the Gemini and Apollo Programs. Subcomponents of flight computers, such as controllers and filters, require extensive testing before they can be implemented on physical systems. The use of different types of simulators[82] with different capabilities provides methods to validate and test these subcomponents prior to deployment. Some testing may compromise the physical system and render it dysfunctional. Other testing conditions cannot be

accomplished in a laboratory, such as a microgravity environment. Simulators allow the designer to circumvent these obstacles of physical experimentation. Simulators also reduce testing time. Setting up an experiment may require hours, days, or even months. Due to the numerous advantages of simulations, there has been a steady increase in their usage. Some of the most common applications for simulators include pilot training,[83;84] and studies regarding pilot health[85;86] among others.[87] Simulators have been designed to aid researchers in the development on new flight control systems.[88] It must be remarked that simulations do not ultimately replace testing on real systems, but can reduce the required iterations.

Following a set of rules,[89] simulation capabilities have been built into the SAI. The simulation capabilities includes hardware-software interaction, third party system integration, and subcomponent testing.

In this chapter, the simulation requirements are first introduced. The methods in which simulation capabilities are built-in into the SAI are then described. Later, all simulation possibilities are discussed. A set of simulations are presented that shows each step required to evaluate a controller. A new systematical approach is introduced to mitigate the hardware constraints on the controller implementation. Finally a flight controller for a UAV is presented and evaluated using hardware-in-the-loop simulation, as a case study for Hardware-in-the-Loop simulation testing using SAI

3.2 AUTOPILOT SIMULATOR CONNECTION

The key feature to enable hardware in the loop simulation is the connectivity capabilities of the SAI. There are different methods to interconnect multiple computers running the SAI framework. These computers can be linked with other systems running simulators of different kinds, and access hardware interfaces at the same time. The framework provides a set of tools that enables the use of different standard communication protocols such as UDP, TCP/IP, sockets, among others, which are popular in third party simulators such as: Gazebo or Flight Gear. With the use of these modules the SAI can exploit the capabilities of said simulators allowing the designer to focus in the problems related to the software/hardware interaction rather than software technical issues.

One of the applications of the frameworks is to implement a hardware bridge between a simulator and hardware component. The SAI can be used to work as an interface for flight simulators and a custom built cockpit. The system has the ability to perform data integrity checks, validation, frequency adjustment, unit conversions, and reference changes among others.

The main objective of the simulations is to allow controllers to be implemented through a systematic procedure where the limitations imposed by the hardware can be gradually added to the simulator. Isolating causes and consequences to obtain a clearer understanding of the situation will help the system designer to develop better solutions in shorter times.

Using a combination of inexpensive hardware with data acquisition (DAQ) ca-

pabilities such as Beagle Bone, Arduino, and other micro-controllers it is possible to transmit data acquisition capabilities to more powerful computers, such as desktop and laptop computers, which often do not have DAQ capabilities themselves. The designer can then focus solely on algorithm development without worrying about integrating hardware or finding expensive computers with DAQ boards.

Simulators can run at higher frequencies than the data acquisition devices. These frequency differences allow the use of network protocols to interconnect the computers. Network latencies usually range from 1 to 10 ms for local networks. The network latencies, can be neglected since hardware data acquisition loops usually run at 100 Hz at most. In some cases, where the latencies increase beyond the tolerable ranges a mechanism is in place to accommodate for this delays. When the SAI code is working with real-time applications such as flight simulators, the internal frequency can be adjusted by modulating the clock speed up or down, and frames can be droped to accommodate for the latencies. Using this mechanism, the SAI can generate data to match the timestamp of the receiver rather than the sender.

The basic configuration layout used for simulations is shown in Figure 3.1. The SAI configuration implements the HALSim which encapsulates the simulator connection protocols.The GNC module remains unchanged allowing direct use and validation in different scenarios. Once the controller is validated with the simulator it can be taken into the next steps without any further controller or filter tuning.

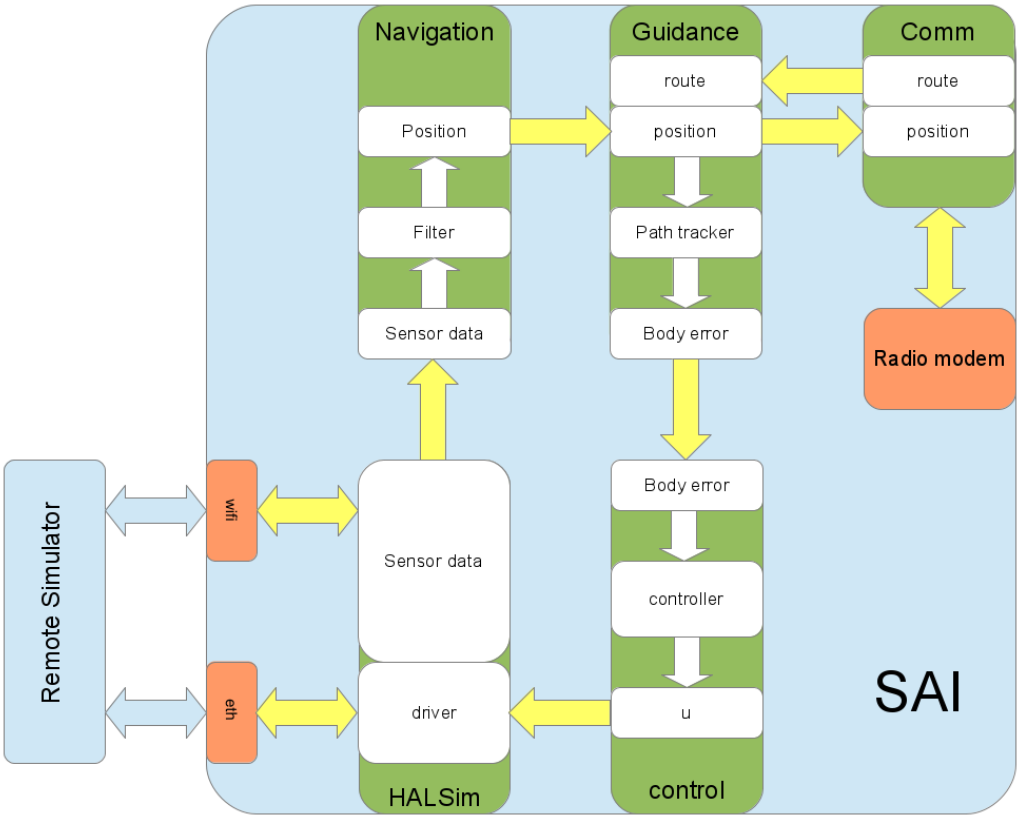


Figure 3.1: Autopilot configuration layout used for simulation

COMPUTER SYNCHRONIZATION

A key challenge on distributed system is the time synchronization between the different participants, since it is not possible to share a common timer or reference time line. Using a service such as a Network Time Provider (NTP) the clocks from multiple computers can be put in synchrony with a precision in the order

of hundreds of milliseconds. For high frequency applications the clock differences should be on the order of a few milliseconds. There exist more advanced methods that allow a much more precise clock synchronization, but the framework relies on an implementation of the Precision Time Protocol (PTP) standard[90] which runs on a external layer. With this extra layer, the clocks are kept synchronized without interfering with the simulation. Using as a reference a NTP or external GPS, the different computers share a common and precise internal timer. If the external reference is not available, the synchronization can be performed taking any one of the network computers as a reference.

Using a Best Master Clock (BMC) algorithm, the computers in the network decide who will assume the role of master; the remaining computers act as subsidiary units. The goal is to estimate the offset (δt) which is the difference between the master clock ($m(t)$) and the slave clock ($s(t)$) at time t as shown in Equation (3.1).

$$\delta t = s(t) - m(t) \tag{3.1}$$

The server broadcasts a message at T_1 that is received at T_1' by the slave. The slave needs to estimate the network transmissiontime to properly synchronize with the master clock. The slave estimates a round trip time to the master by sending a message at T_2 to the master. The master receives the message at T_2' and sends a response with the time when the original messages was received. Through this exchange the slave can determine T_1, T_1', T_2, T_2'; the message exchange is shown in Figure 3.2. With the quantities determined the offset can be compute using Equation (3.2). With this offset, each computer can adjust the internal clock to

match the master's clock. Thus, and efficient and effective communication interface is established within SAI.

$$\delta t = \frac{T_1' - T_1 - T_2 + T_2'}{2} \tag{3.2}$$

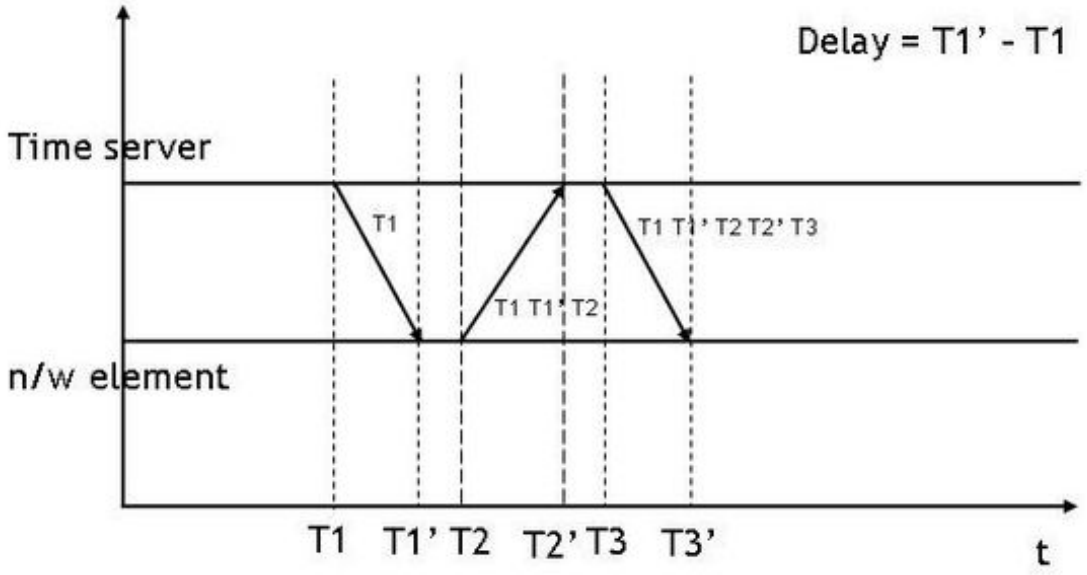


Figure 3.2: Message exchange between the master and a slave time diagram

3.3 NUMERICAL INTEGRATION OF DIFFERENTIAL EQUATION OF MOTION

Most synthetic data generated in simulators is obtained by numerically integrating an ordinary differential equation (ODE). This ODE represents the equation of motion (EOM) of a dynamic system. Any simulator needs to implement a solver.[91] A complex solver will give greater flexibility to the overall system, however this may

have a negative impact on the ease-of-use. The SAI adopts a powerful and versatile solver that have been recently accepted to be distrubuted with boost libraries.[92] To keep the system complexity to a minimum this solver is wrapped in a simple yet flexible interface which allows fast EOM development. Complex methods of the solver are only accessed when extra functionality is required.

LINEAR SYSTEMS

Due to its simplicity, the Linear Time Invariant (LTI) systems implementation is discussed first. LTI systems are governed by a linear ordinary differential equation Equation (3.4) and the corresponding measurement equation Equation (3.4). $\dot{\boldsymbol{x}}$ has dimensions $n \times 1$, A has $n \times n$, $\boldsymbol{x}$ has $n \times 1$, B has $n \times r$, $\boldsymbol{u}$ has $r \times 1$, y has $m \times 1$, C has $m \times n$ and D has $m \times r$.

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \tag{3.3}$$

$$\boldsymbol{y} = C\boldsymbol{x} + D\boldsymbol{u} \tag{3.4}$$

For this type of problems, the four system matrices and the initial conditions required for integration are given by $\boldsymbol{x}(\boldsymbol{h}) = \boldsymbol{x}_0$. For each time step, numerical integration is required to propagate the state vector $\boldsymbol{x}$. This numerical integration requires for each time step the system input $\boldsymbol{u}$ and generates the output vector $\boldsymbol{y}$. The state variables are stored internally from the previous time step. The system uses the signal schema discussed in Chapter 1 to store the state vector, the

inputs, and outputs, giving the designer the ability to access the internal states if required for debugging purposes. On the initialization of the system the module will load from a configuration file the four system matrices since the system is time-invariant there is no need to updated the matrices. The initial conditions are provided by measurement signals. The integration step is carried out using the solver integrators.

To further extend the capabilities of the integrator a transfer function parser is implemented. The parser, when reading a configuration file described by two coefficient vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ will use an algorithm Equation (3.6) to convert the transfer function to its equivalent space state representation. Since the space state representation is not unique we, adopt the controller canonical form.[68] To ensure that the system can be implemented the transfer function must be proper or strictly proper.[93] The main advantage of the parser is that no extra modules or complex mechanisms such as an inverse Laplace transform block needs to be implemented. A set of simple algebraic operations that are computed before the system completes the initialization process is enough to deal with transfer functions.

$$tf(s) = \frac{Y(s)}{U(s)} = \frac{\sum_{i=0}^{n} b_{n-i} s^i}{\sum_{i=0}^{n} a_{n-i} s^i} \quad a_0 = 1 \tag{3.5}$$

$$\begin{Bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{Bmatrix} = \begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 1 \\ -a_n & -a_{n-1} & \dots & -a_1 \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_n \end{Bmatrix} + \begin{Bmatrix} 0 \\ \vdots \\ 1 \end{Bmatrix} u$$
$$y = \begin{bmatrix} b_n - a_n b_0 & \vdots & b_1 - a_1 b_0 \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_n \end{Bmatrix} + b_0 u \tag{3.6}$$

To extend the capabilities to Linear Time-Varying (LTV) systems the entire setup can be reused, however the matrices can no longer be hold in constant data structures. If the matrices are represented with signals (one signal per element) then the entire EOM description can be updated on each time step. If the EOM have only a few time-varying parameters, then, only those can be connected to signals, keeping the other elements constant.

NONLINEAR SYSTEMS

A wide range of problems are described by a set of nonlinear differential equations. To have a capable simulator, nonlinear integrators are required. The nonlinear system is described by two functions Equation (3.7). We first describe how the nonlinear systems can be implemented and later on we discuss how can be solved.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{3.7}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\nu} \tag{3.8}$$

To keep the system flexible yet simple a Class ODENonLinear Figure 3.3 has been designed. All classes that implement differential equations inherit from the same base ODE. These blocks can be used as filters or controllers and not just for simulation. The integration procedure only requires the system to provide the $\dot{\boldsymbol{x}}$ and in return gives $\boldsymbol{x}$ for each time step. Most differential equations are described also with a measurement equation $\boldsymbol{y}$. The general procedure for the integration is resumed in Algorithm 3. This method gets invoked by the integration handler function, which is built-in into the SAI.

Algorithm 3 Integration procedure

```
for All Signals in input do
    set u equal to Signal
end for
for All Signals in states do
    set x equal to Signal
end for
one step (dt) integration
Update system outputs
for All Signals in output do
    set Signal equal to y
end for
```

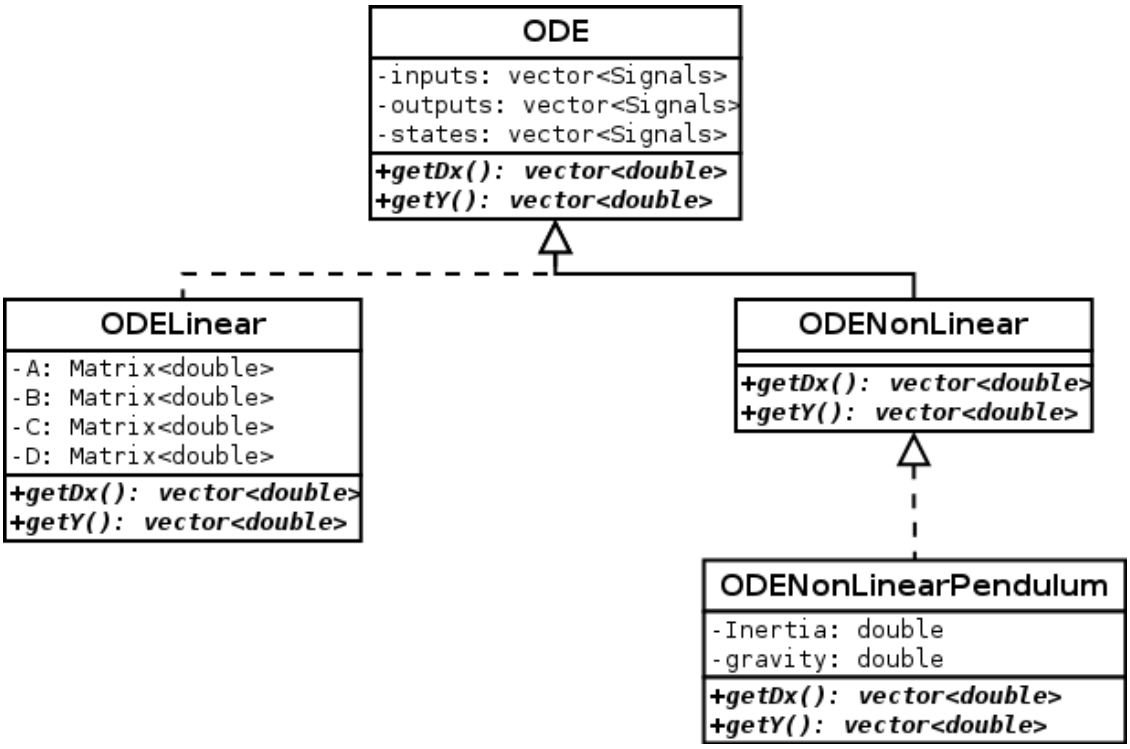


Figure 3.3: ODE Class UML design

NUMERICAL INTEGRATION

The overall integration procedure remains unchanged as can be seen from the algorithm. Some differential equations may required special integration techniques such as error checks, variable-width time steps; just to mention a few.[94] Although these methods increase the integration precision they have a heavy impact on the computational burden. Since each ODE requires its unique features, the integrator can be changed based on the designer requirements in the SAI

The main problem of implementing the numerical integrator in real-time is the control of the dt used for integration. The dt cannot be as precise as an off-line integration where the time is specified in a vector where the precision is given by the variable size. When real time applications are implemented the time clock must be controlled. however, for high frequency problems, the time clock cannot be tuned such that the required integration precision is satisfied. With the use of the real time tools the precision of the controlled dt can be increased. It is important to note the difference between the dt controlled and the dt measured. The controlled dt is the one that is set, on the interrupt control. Interrupts, are the internal mechanism of the computer to control function executions based on a time schedule. If the loop should run at 1000 Hz then the control interrupt should sleep the function for any of the remaining time in the cycle, for instance if the system should sleep for 90 ms the controller may sleep for 92 ms instead. This simple example shows how the control dt can differ from the ideal dt. The precision of the dt measured is usually in the order of several μs. Although the dt can be controlled in the range

of ms it can be measured with μs precision. Knowing the time difference between the controlled and ideal dt the integration can be adjusted. Most importantly, the tolerances of numerical integration process can be stablished in real time.

For each cycle of the integration loop, the differential equation gets integrated one step by a stepper. The step takes as initial time the previous step final time, the step is the dt measured, and the final time is given by the initial time plus the dt. If a higher precision is required the measured dt can be divided in an interval of n steps. The integration procedure is then repeated for each subsample on the interval. To increase the precision of the estimates in the simulation, the stepper can be such as to support error control, and the convergence can be controlled. The steppers implemented are Rosenbrock,[95] implicit Ruge-kuta,[96] Adams-Bashforth-Moulton,[97] Runge-Kutta-Cash-Karp,[98] among others; for a complete list please refer to odeint documentation.[99]

We perform a test to validate the integrator algorithm. A differential equation given by Equation (3.9) is numerically integrated and the result is compared against the analytical answer shown in Equation (3.11). The results are shown in Figures 3.4 and 3.5. In the next section a more comprehensive simulation is presented.

$$\dot{x} = -x \tag{3.9}$$

$$x(0) = x_0 = 0 \tag{3.10}$$

$$x(t) = 1 - e^{-t} \tag{3.11}$$

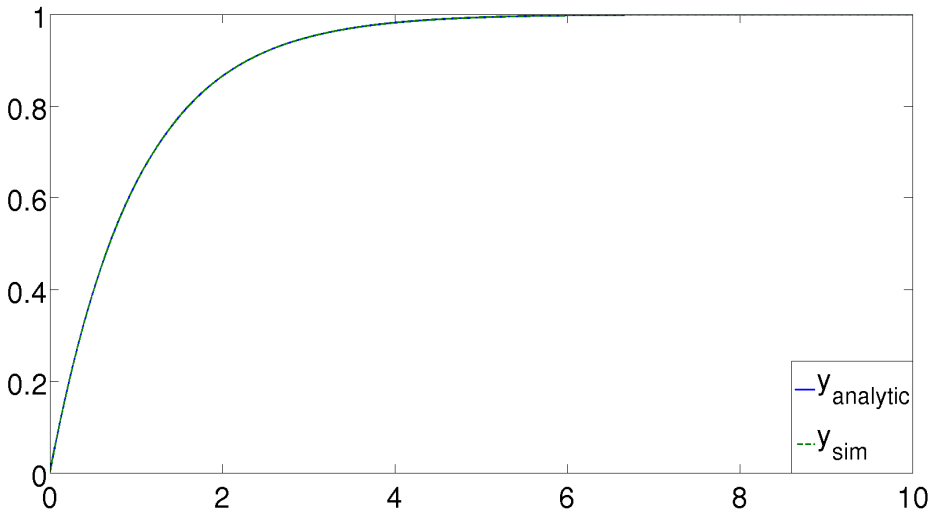


Figure 3.4: Numerical integration and analytical solution comparison

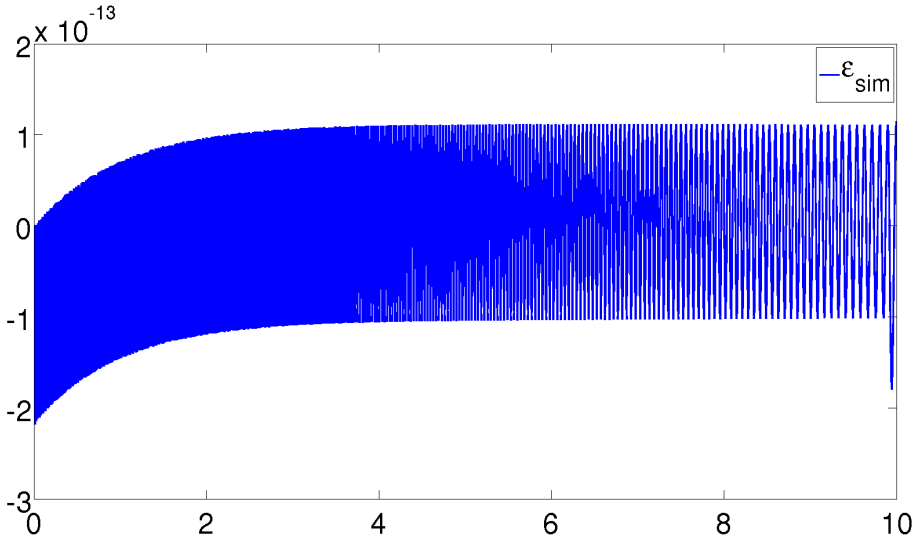


Figure 3.5: Numerical integration error target set at $1 \cdot 10^{-12}$

3.4 SATELLITE ATTITUDE SIMULATION

Simulations are intended to validate a controller design for a custom application. In this section we present a systematic approach to validate controllers with successive phases of simulation. We iterate over the simulations modifying the configuration on each step to increase the similitudes with a real system. The final simulation is performed with hardware-in-the-loop.

The EOM selected is a simple plant with two poles at the origin Equation (3.12). This equation represent a satellite attitude EOM for a single degree of freedom.[69] A simple controller, given by Equation (3.13) is designed to give a stable response to a step input. The closed-loop plant is given by Equation (3.14). Using the inverse Laplace transform the time response to a step can be found Equation (3.15).

$$tf(s) = \frac{1}{s^2} \tag{3.12}$$

$$C(s) = \frac{s+1}{s+5} \tag{3.13}$$

$$G(s) = \frac{s+1}{s^3+5s^2+s+1} \tag{3.14}$$

$$\begin{gathered} y(t) = 1{+}0.0347e^{-4.8359t} - ... \\ (0.5173 - 0.0932I)e^{(-0.082-0.4472I)t} - (0.5173 + 0.0932I)e^{(-0.082+0.4472I)t} \end{gathered} \tag{3.15}$$

The analytical solution gives the theoretical system response to the step input. This response is only for the ideal case and it fails to capture several system limitations, such as noise, saturation, controllers delays, among others. The use of the simulations allows us to add the hardware limitations and constraints one by one to analyze how the overall control performance degrades when each new limitation is taken into account.

The first simulation is a simple real-time integration.[100] The EOM of the plant are modeled and configured in a block, and the controller ODE is modeled as another block. Both are connected with the use of Signals as shown in Figure 3.6. A signal is used to control the reference, for this case a step which is modeled as a constant block. The constant block sets a value to the reference signal when the system is initialized. For this first simulation a single program is used with two main blocks. The results are much similar to the ones that can be obtained with a program such as Simulink. One important remark regarding this simulations is the use of a real-time clock. The time vector is not evenly spaced and some error on the control is present, as described in Section 3.3.3.

We run the simulation for 15 s. After the real-time simulation is run we have two data vectors. The first vector contains the plant output and the second one contains the corresponding time stamp. To perform a comparison with the analytical solution the time vector from the simulation is used to generate the data points for the comparison. In Figure 3.7 the results of the integration are superimposed on the analytical solution. The error between the simulation and the analytical solution is in the order of $1e^{-3}$ as shown in Figure 3.8. The error for this first

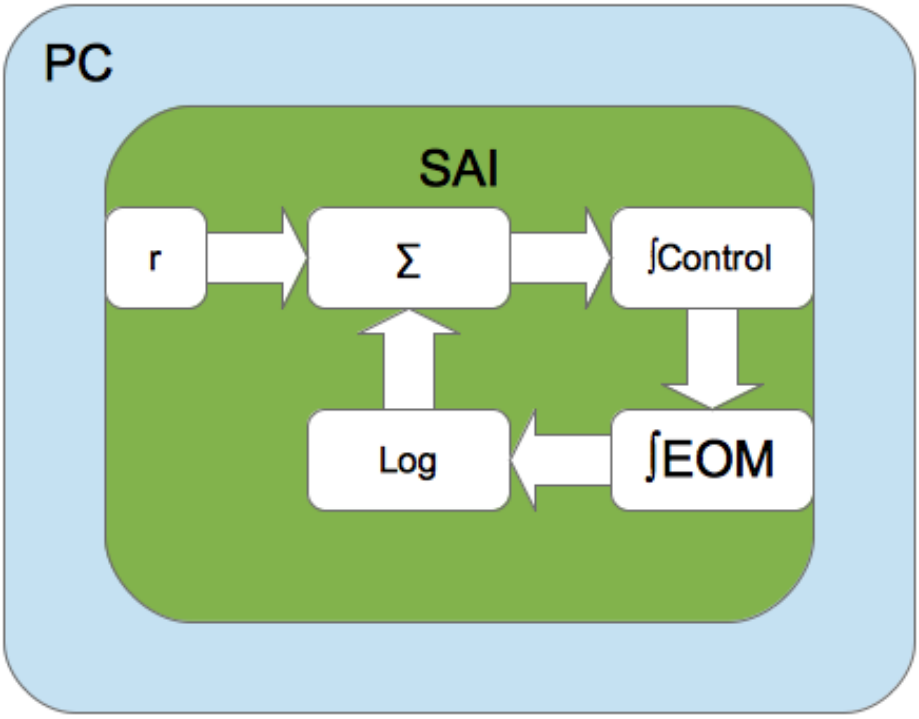


Figure 3.6: SAI Block diagram for the plant-controller close-loop simulation

simulation is due to the numerical integration with an unevenly spaced time vector. The integration is performed at 1000 Hz with $\delta t = 0.001$. A key difference with the analytical solution is that we have two ODEs as opposed to one. One ODE represents the plant dynamics and the other represent the controller. For an analytical answer both ODEs can be combined into the same equation and solve the system at once. On the other hand in practical applications the controller and the plant cannot be combined. For this last case two ODEs are solved and the

output of the controller is fed as input to the plant. The plant output is fed into an output feedback controller closing the loop.

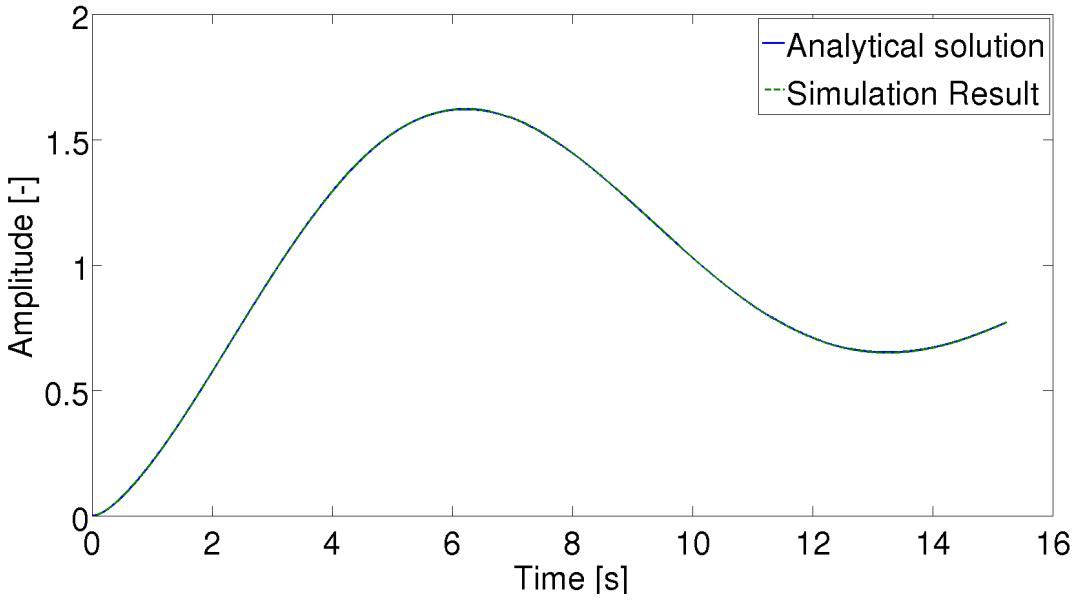


Figure 3.7: Analytical solution vs real-time simulation results

Since the controller still performs within the scope of the design envelop we continue with a new iteration on the simulations. For this new simulation we use two threads. The threads performs data exchange with the use of signals. On each iteration of the integration the threads update the data with the last known states. In this case we have a more realistic scenario, where the controller runs on its own loop with a reference frequency. The plant is integrated at a higher frequency to increase similarities with a real plant. This dual thread configuration tries to emulate a plant that behaves as close as possible to a physical system and a controller that behaves like a micro-controller unit (MCU).

The outcome of the simulation is presented in Figure 3.10 . From the presented

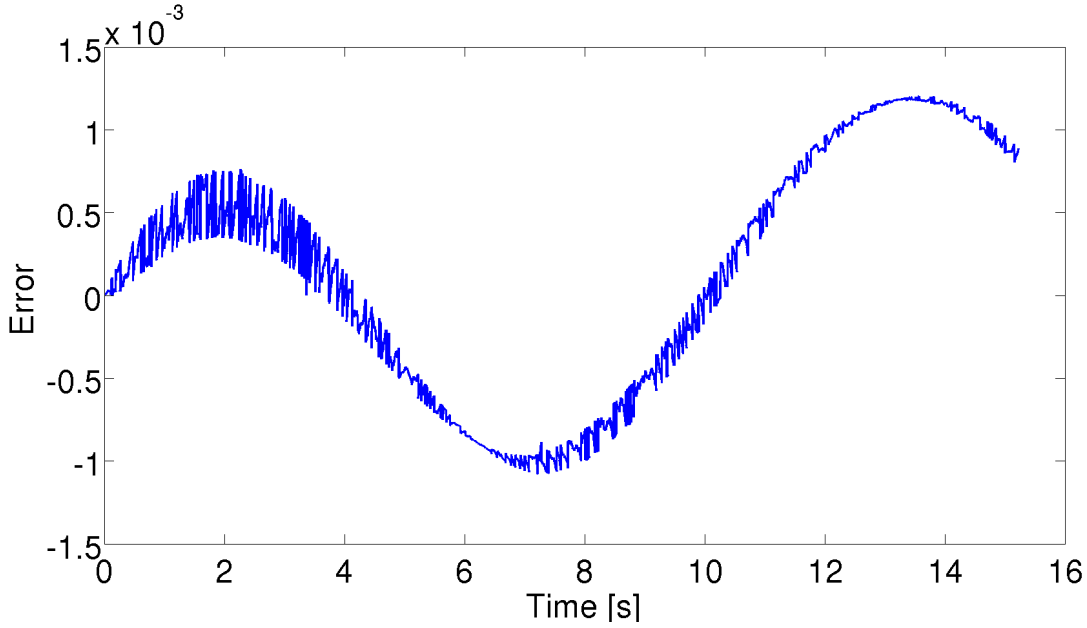


Figure 3.8: Real-time Simulation Error

results it is clear how the simulated response lags from the analytical solution. For this scenario the integration is no longer performed at high frequency and synchronized with the physical system. The lower frequency of the controller is responsible for the overall lag in the time response. For this case the controller no longer can compensate instantaneously for the plant error.

For the final simulation scenario the controller is implemented in a different computer than the one performing the plant EOM integration. Both computers are connected using the hardware I/O interfaces. The controller computer controls the plant input with a PWM signal. This signal is acquired by the computer simulating the plant, which converts the pulse width to a dimensionless quantity. After the EOM are integrated the computer generates a square wave that represents an encoder output. The synthesized encoder signal is acquired by the computer

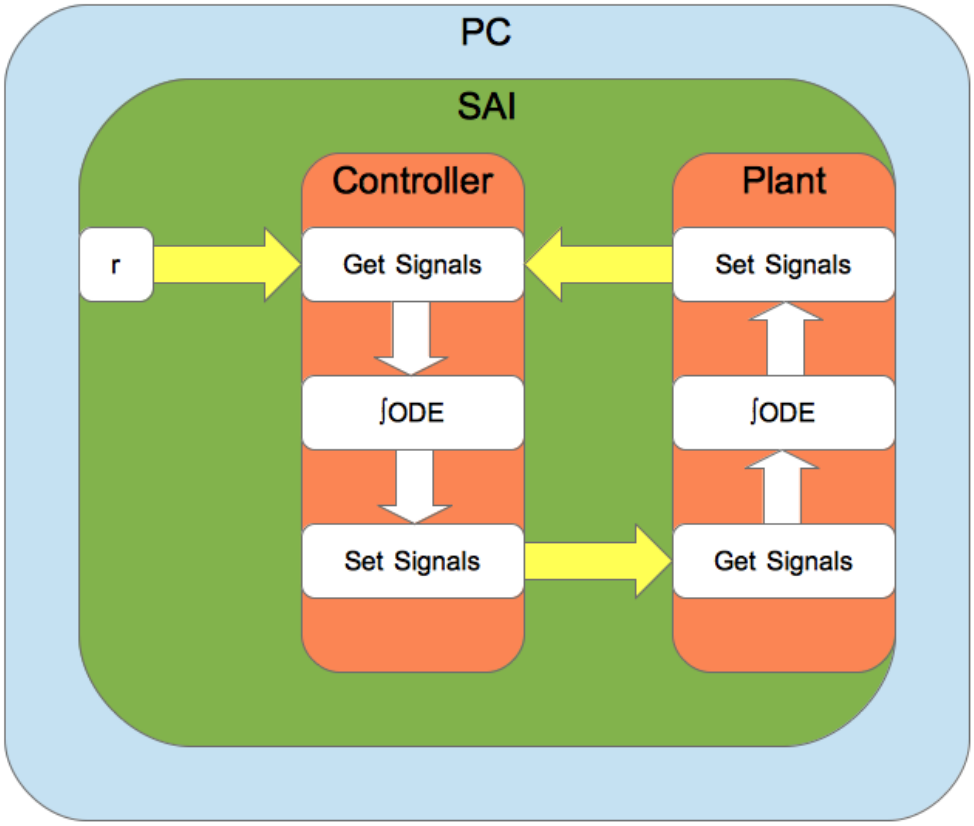


Figure 3.9: SAI Block diagram for the plant-controller close-loop simulation running in two independent threads

acting as controller which uses this signal to generate the controller feedback signal. In Figure 3.11 we show a diagram of the computer layout.

As can be seen from the output present in Figure 3.12 the controller fails to achieve the required response; even more the response is divergent. This set of simulations shows how a controller degrades from the theoretical solution to the final implementation. Several sources of nonlinearities and noise are neglected,

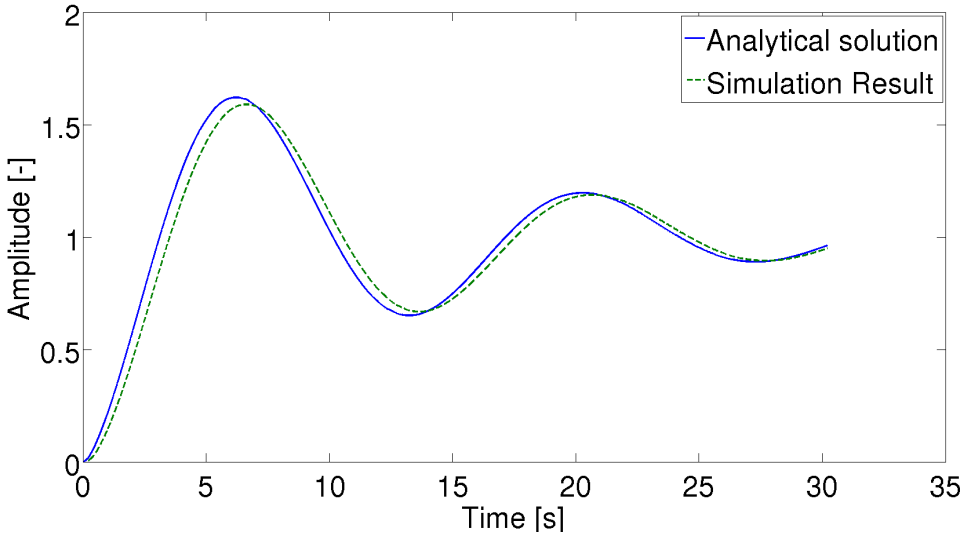


Figure 3.10: Analytical solution vs real-time multi-threaded simulation results

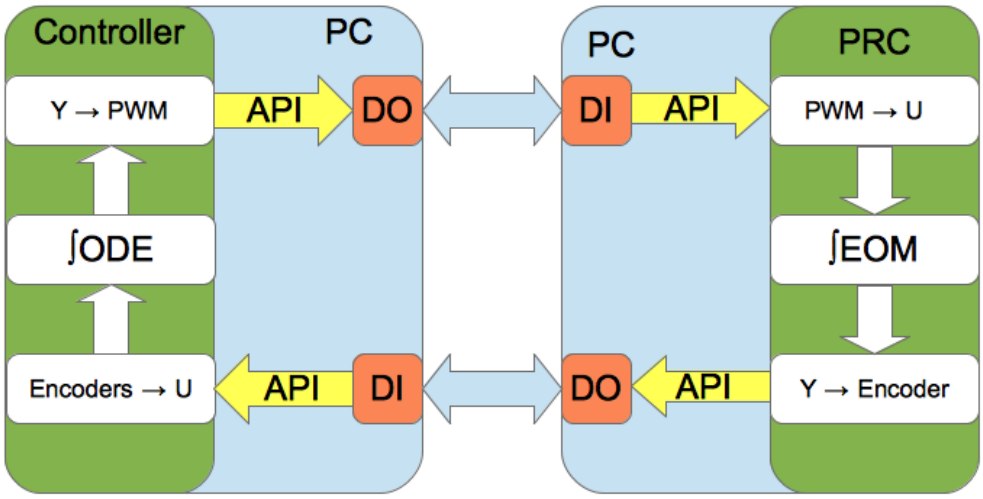


Figure 3.11: SAI Block diagram for the plant-controller close-loop simulation running in two independent computers

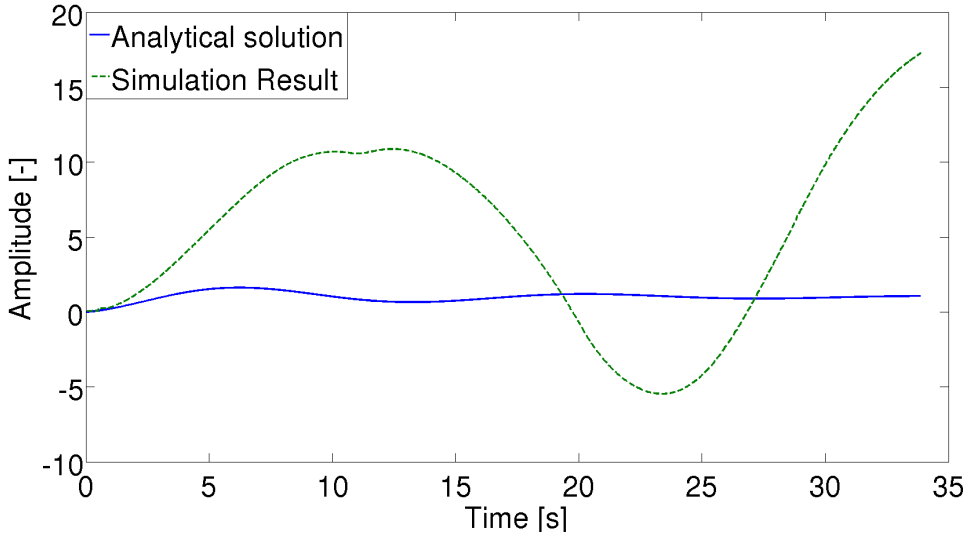


Figure 3.12: Analytical solution vs real-time multi-computer simulation results

but as can be seen from this set of simulations, even without these considerations some controllers fail to achieve the expected result. A method to address these issues is to adjust the gains of the controllers after each simulation; this will allow the designer to adjust the gains in a small neighborhood after each step rather than redesigning the entire controller after the system failure. These methodology, is then used trough out the dissertation to tune the controllers of more complex dynamical systems such as the UAV. This new approach allows the designer to iterate on the controller design to easily incorporate the more complex mode of the dynamical system and/or the hardware constraints. In the next section we use the HIL simulation capabilities to test the controllers designed for the UAV presented in Chapter 2.

3.5 UAV SIMULATION

The mathematical model and the controller developed in Chapter 2 are evaluated using the HIL simulations capabilities of the SAI. For this test, the autopilot is configured with all the controllers and filters required for a proper flight. The IMU and GPS information is fed to the autopilot using a serial port connection. This type of connection is the same used for the IMU/GPS sensor used for the flight. The autopilot sets the PWM outputs following the requirements to operate the RC servos. A secondary computer is used to acquire the PWM signals and set the serial message. This secondary computer is used to control the flight simulator. The hardware setup is shown in Figure 3.13.

The two computers run instances of the SAI. The first computer operates with the autopilot configuration. The second computer is configured as a flight simulator interface; the module diagram is shown in Figure 3.14. A third computer is used to run the flight simulator.

To test the nonlinear model we use the feedback control law designed in the previous chapter. First we test the longitudinal controller applying a step to the θ input. The comparison between the linear and nonlinear longitudinal models can be seen in Figure 3.15. The time response of the linear and nonlinear model are similar, justifying the use of a linearized model in the defined operating regime. It can be seen that the main error is in the steady-state condition. This can be easily corrected by tuning the input scaling matrix.

To evaluate the lateral controller we excite the nonlinear model with a doublet

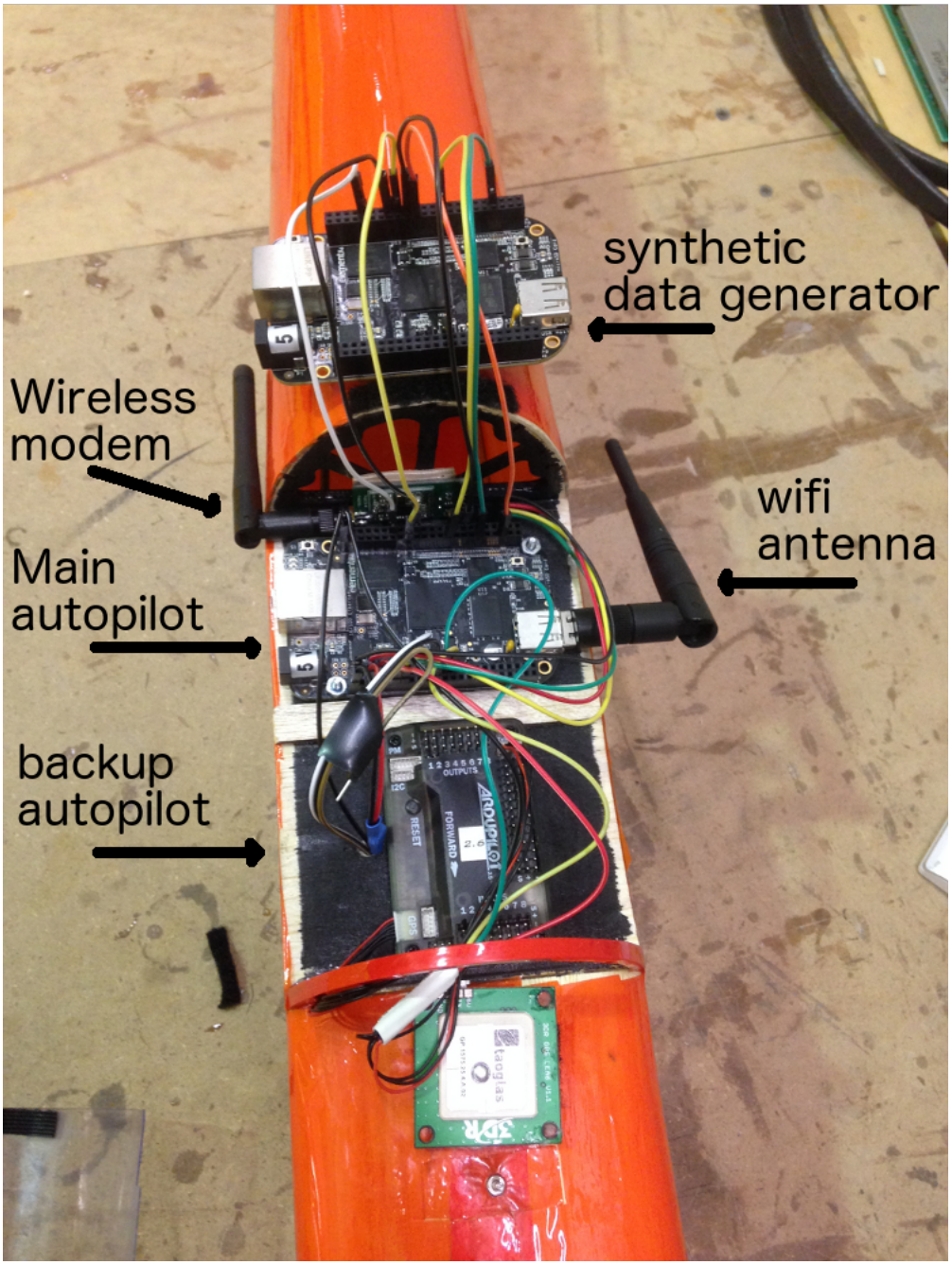


Figure 3.13: Experimental layout for UAV HIL simulation

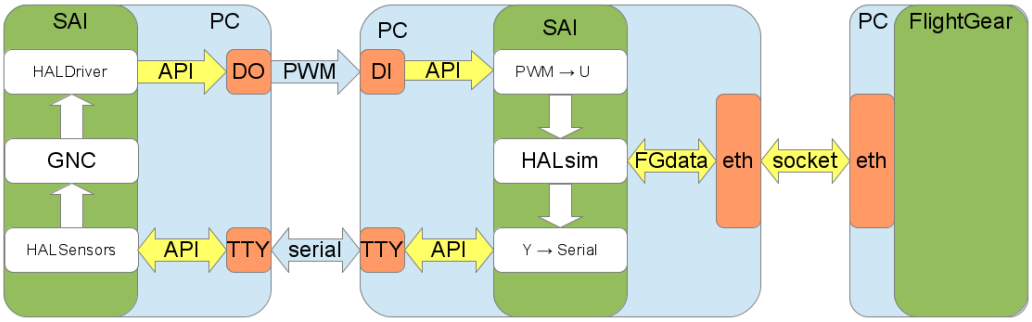
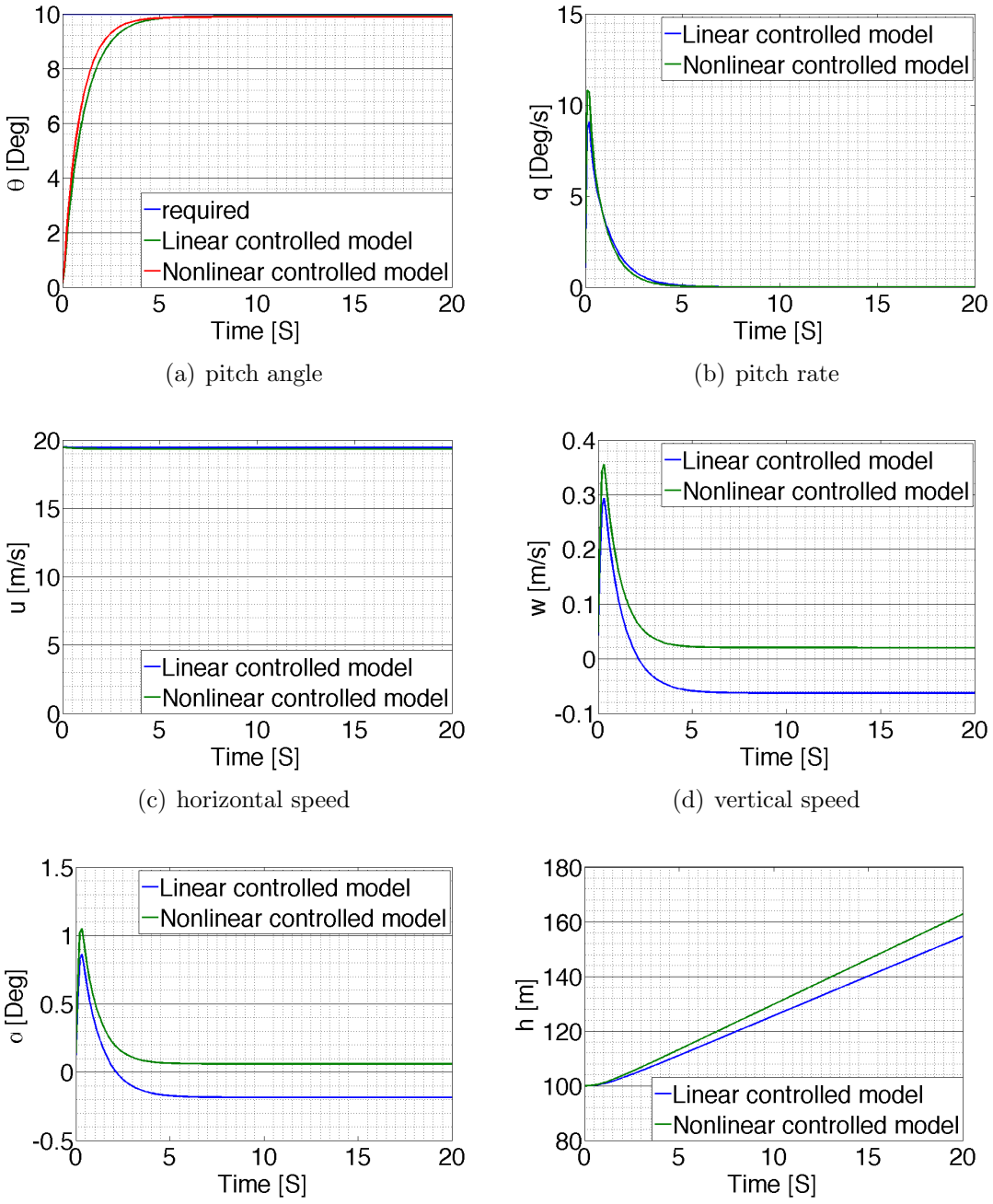


Figure 3.14: SAI module configuration for UAV HIL simulation

input in the ailerons first and then the rudder. The output presented in Figure 3.16 clearly shows that the controller perform as expected controlling the dynamics of the plant. The error again is due to the scaling of the inputs. The scaling matrix which accommodates the input has to be tuned to reduce the steady-state error.

The time response of the system to the different inputs validates the design hypothesis. These hypothesis are the basis for model separation, linearization and mathematical model of the hardware constraints. The controllers designed using linear techniques performs as expected when applied to the nonlinear model. The decoupling of the lateral and longitudinal dynamics for the controller design can be validated by observing the system outputs of the nonlinear controlled plant. Each control surface modulates an independent variable as required with little to no cross-variable influence.



(a) pitch angle

(b) pitch rate

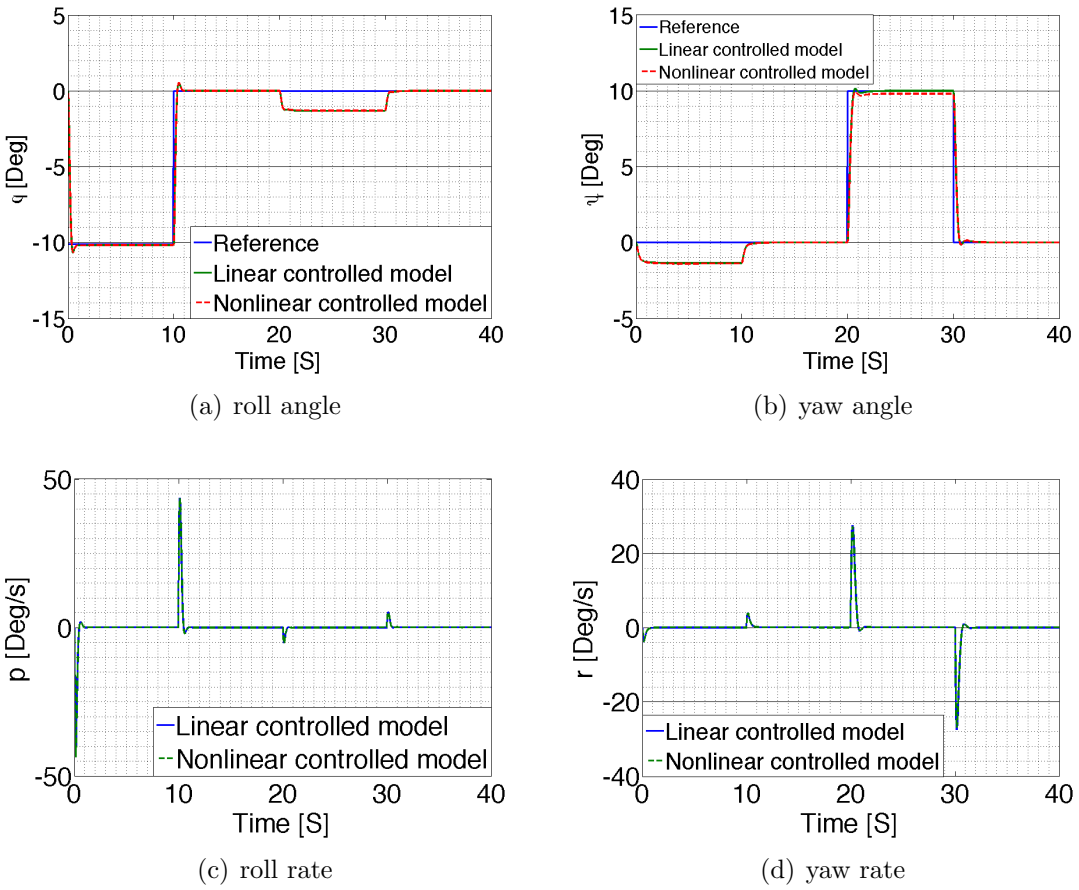
(c) horizontal speed

(d) vertical speed

(e) angle of attack

(f) altitude

Figure 3.15: Controlled nonlinear model longitudinal response to elevator input



(a) roll angle

(b) yaw angle

(c) roll rate

(d) yaw rate

Figure 3.16: Controlled nonlinear model response to doublet input on ruder and ailerons

3.6 CONCLUSIONS

In this chapter the simulation capabilities of the SAI are presented. The key challenges of simulation and how they are addressed by our software are discussed. The software-only simulation capabilities are first introduced and some examples are presented. Then, the hardware-in-the-loop simulation is presented. An example shows how simulations are used on a systematic way to fine-tune controllers. Finally, a flight controller is tested using a flight simulator. The experiments presented in this chapter demonstrate the importance of simulation in the design and tuning of controllers for different systems. The approach presented helps the designer to escalate gradually the controller complexity to take into account different hardware limitations one at a time.

Chapter 04

TESTING AND VALIDATION EXERCISES

> The knack lies in learning how to throw yourself at the ground and miss.
>
> Douglas Adams, Life, the Universe and Everything

4.1 INTRODUCTION

In practice, the performance of the control system to meet specified objectives is conditioned on the implementation methodology. Designing a controller that can compensate the plant and manipulate the plant dynamics according to the desired performance represents just the initial step.[64] Control system implementations typically depend upon the sensors, actuators, communication and data acquisition devices, etc. Robustness to parametric and model structure uncertainties and the presence of noise in the measured signals are among the common issues that the designer must properly address before a controller can be tested on a physical plant. However, there is more to the implementation than a set of algorithms. A set of

predefined and carefully designed validation tests need to be performed before the flight controller can be put to the test on a flight. Several implementation problems such as computer system failure,[101] operational limit on actuators and sensors, system recovery and retrieval among others are typically encountered by the practicing engineers. The objective of this chapter is to identify these issues, while the problems encountered during the development of the SAI-GCS provides a context for this discussion and detail the solutions adopted by us to alleviate the identified problems.

The next section introduces a robust design for an autopilot switcher. The method to override all automatic controls is then discussed. The algorithm used to track waypoints is discussed in Section 4.3. This algorithm, then, is adapted to track routes. The considerations regarding sensor mounting and electromagnetic shielding are presented in Section 4.4. The redundant flight control system and associated algorithms for way point tracking, including the manual over-ride mechanism are validated using a way point tracking demonstration on an Unmanned Ground Vehicle (UGV). Finally the controller designed in the previous section is tested on a unmanned air vehicle. Control system response for our implementation is discussed using the flight test results.

4.2 REDUNDANT FLIGHT CONTROL SYSTEMS

When autonomous vehicles are being used, manual control is always recommended for the safety of the vehicle. In some conditions, it is required to override all the on-board autopilots to regain manual control over the vehicle. In case multiple

autopilots are being used, either as backup or to compare controller performance, it is sometimes required to switch between them. Letting any of the autopilots perform the controller switch is not a reliable operation, because the control hand-off is typically initiated by the human operator. If the autopilot handling the active controller fails, recovery becomes impossible and the vehicle will be compromised or lost. Another limitation on using an autopilot to perform the controller switch is the high energy consumption of the system. In case the battery levels drop below a certain threshold, the autopilot should be turned off to save battery; this requires the control of the vehicle to be independent of the autopilot. The most reliable method to switch and control autopilots is to use a RC controller signal directly, bypassing all computer processes. The RC signal is a PWM with a duty cycle of $0.02s$. The pulse varies from 1 to 2 millisecond repeated at $18ms$. A reference signal is shown in Figure 4.1. To process and interpret this pulse a microcontroller can be used. However, this option requires a microcomputer with a custom firmware and a DAQ which increases the system failure rates. Since the system is intended to work not only as a switch but also as a emergency bypass, a more robust solution is developed.

A more robust method is to design a custom analog circuit to act as a digital switch. Although this imposes an extra effort on the designer, it increases the system reliability. A reference diagram is shown in Figure 4.2, where in green the autopilots are represented, the servos that control the vehicle are shown in brown, the RC receiver block is in orange, and the switcher circuit is in blue. The circuit is simple and efficient. First the PWM signal is converted to an analog signal with

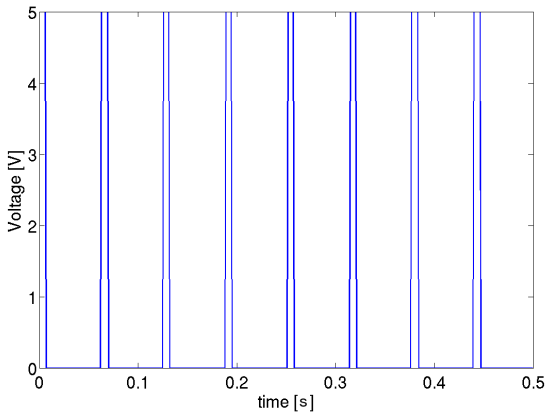


Figure 4.1: Pwm reference signal

the use of a filter and an Operational Amplifier (opAmp). The analog signal is converted to its digital equivalent and decomposed into its bits with an ADC.[102] The independent bits are routed to be utilized as selectors of different autopilots. A set of gates and transistors then translate the control bit signals, to drive the autopilot selection process.

We use frequency domain methods[103] to design the filter that acts as PWM to analog converter. This passive filter thus designed, has the advantages of providing a stable operation with low energy consumption, while providing an elegant mechanism for operations. To improve the response of the filter without compromising ripple quality or quantization level an opAmp is added.

The first step of the filter design is to decide the maximum allowable ripple level, which is defined as the transitory voltage level on top of the continuous, as shown in Figure 4.3. The ripple, is the alternate current that pass trough the filter, and can not be completely removed. Since the Analog-to-Digital Converter (ADC) has an 8 bit resolution we can accept a error in the order of $\frac{1}{2}$ of the Least Significant

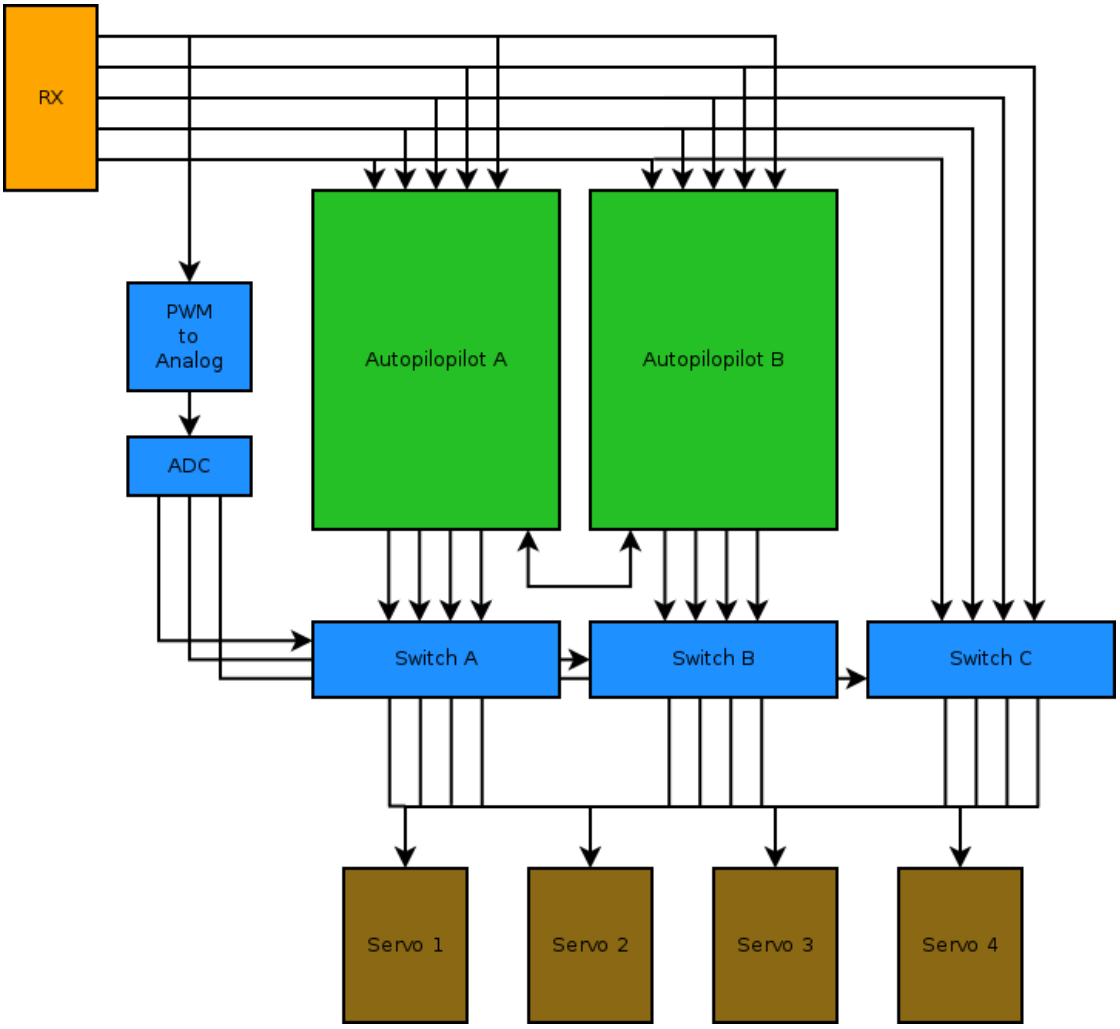


Figure 4.2: Autopilot switch block diagram

Bit (LSB), which for a $5V$ input is

$$V_r = \frac{V_{LSB}}{2} = \frac{5}{256 * 2} = 0.00976 \tag{4.1}$$

Once the ripple and the base voltaje are selected the filter attenuation can be

computed using Equation (4.2)[103]

$$A_{db} = 20 \cdot log(\frac{V_r}{V_{pwm}}) = -12.47dB \tag{4.2}$$

With the required attenuation we can proceed to solve for the corner frequency which is required to select the filter RC constant.

$$f_{3db} = f_{pwm} \cdot 10^{-\frac{A_{3db}}{slope}} = 11Hz \tag{4.3}$$

The gains of the filter are given by

$$f_{3db} = \frac{1}{2\pi RC} \tag{4.4}$$

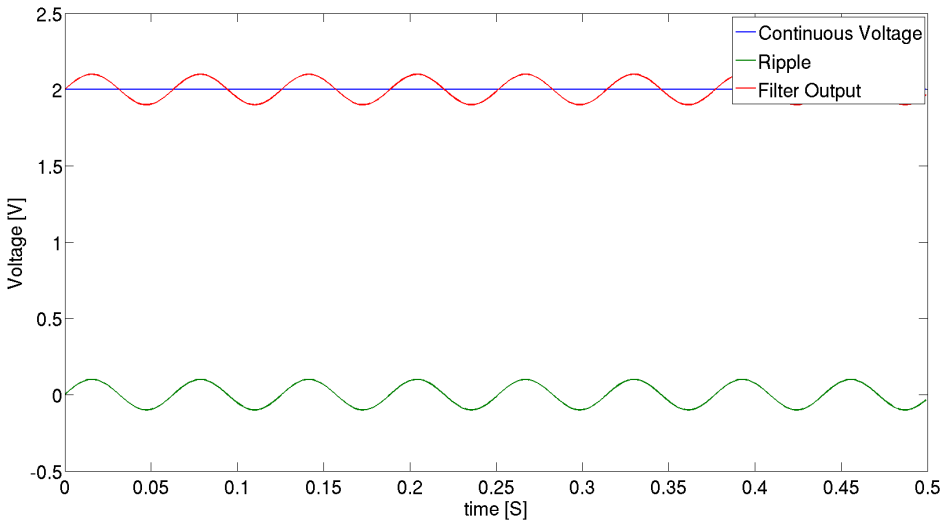


Figure 4.3: Ripple levels on a constant signal

As shown in Figure 4.4 the ripple levels are below the requirements. However,

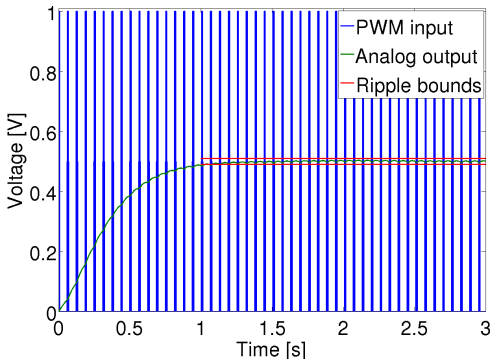


Figure 4.4: Theoretical output for the RC filter

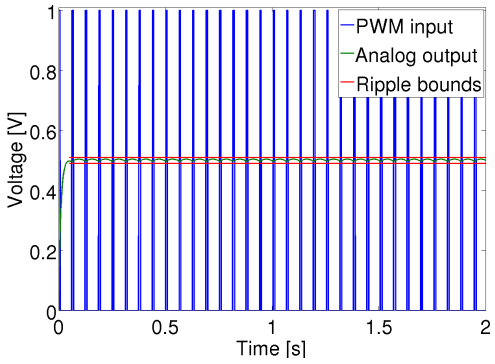


Figure 4.5: Theoretical output for the RC filter with an opAmp

the settling time is too large to be used on a real-time system. Increasing the order of the filter to achieve a faster response requires the use of an inductor to add zeros to the transfer function. To avoid the weight imposed by coils an opAmp is added to reduce the system response time. The main advantage of adding an opAmp is that several off-the-shelf integrated chips are offered with the frequency to analog converter circuit embedded. This embedded circuit can be calibrated with the RC constant obtained in the previous steps. The output of this properly calibrated circuit is shown in Figure 4.5.

The analog signal is used as an input to the Maxim ADC ADC0820CCN integrated chip. This chip is easy to set up and only requires a connection to the main power, ground and analog signal. The configuration pins are typically short-circuited on the breadboard to assure a proper system operation. The digital outputs are connected to a series of switches and transistors to control which outputs are relayed to the servos. A more complex design can be achieved with a set of

gates.

The resulting circuit has low energy dissipation, a weight of few grams, and a tiny footprint. These characteristics make it ideal to be installed side by side with the other autopilots. Since the system requires minimum energy it will be operable even when the batteries are nearly depleted. The switch can also be used to remote shutdown all the devices that are not essential for the flight. In the event that the main autopilot fails, the system can be used to remotely reboot the autopilot and initiate a recovery sequence, while an operator flies the vehicle manually.

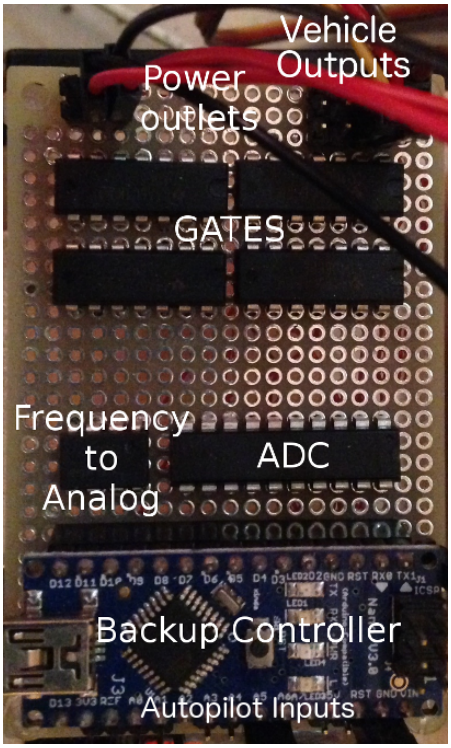


Figure 4.6: Autopilot switch

In Figure 4.6 we show the autopilot switcher. For this implementation a maximum of four different sources for the controllers were considered. Some vehicles

such as the quad-rotor cannot be flown without an autopilot to provide the basic stabilization. Since the NASA standard used on the design states that at least two systems must fail before the system is compromised, a backup controller is added to the selector. The backup controller is a micro MCU which solves simple and critical stabilization controls for unstable vehicles. This autopilot is mounted as a backup and can be completely bypassed. If it is not required it can be completely removed without affecting the routine operation on the switcher.

This section introduced the design of the used autopilot switcher. This switch allows a user to override all commands. The next section introduces the algorithm used by the autopilot to track waypoints.

4.3 WAYPOINT TRACKING ALGORITHM

The route tracking is achieved by the use of a waypoint tracking algorithm over each of the waypoints. The algorithm flow chart is shown in Figure 4.7. The guidance algorithm holds a First in First Out (FIFO) queue with all the desired waypoints. The position and attitude of the vehicle are assumed to be known, The error in the body frame $\boldsymbol{e}_b$ can be computed using Equation (4.5), where $[R(\theta)]$ is the rotation matrix. The error in the Geodetic frame $\boldsymbol{e}_i$ can be computed by subtracting the waypoint coordinates from the vehicle coordinates.

$$\boldsymbol{e}_b = [R(\theta)]\boldsymbol{e}_i \tag{4.5}$$

The error in the vehicle position is shown in Figure 4.10. The error in position (in red) can be decomposed into its two main components, shown in green. The

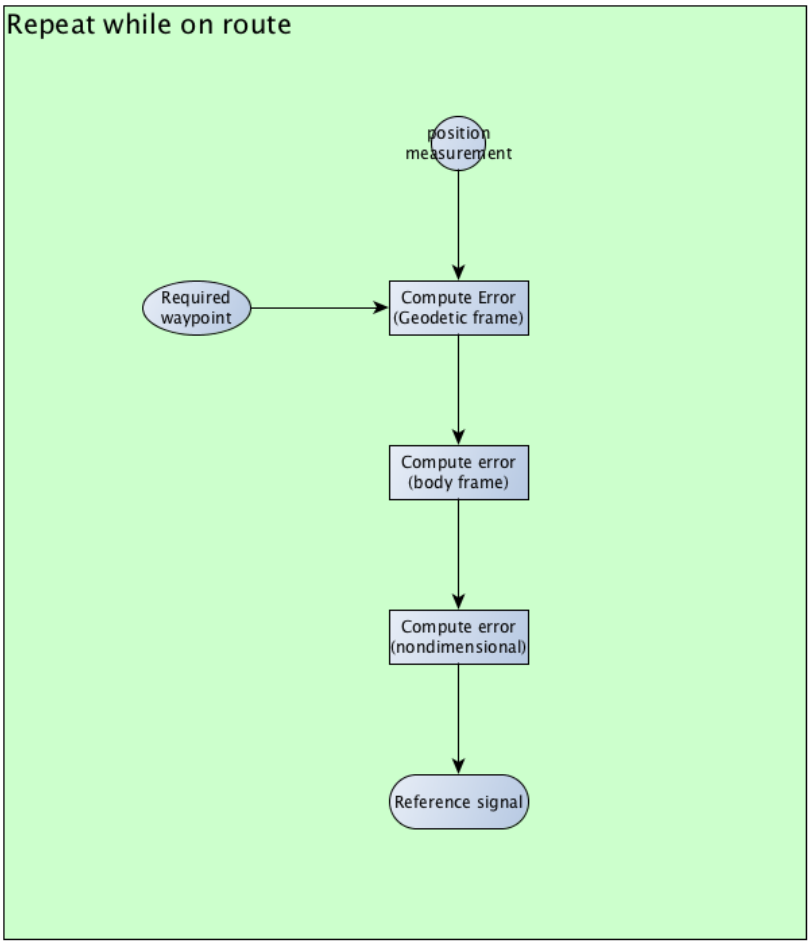


Figure 4.7: Waypoint tracking algorithm flow chart

correction algorithm tries to align the vehicle with the current desired waypoint. To align the vehicle with the objective, the lateral error is nondimensionalized using an approximation given by Equation (4.6). e_ϕ will varies from -1 to 1. When the error approaches 0, the vehicle gets aligned with the waypoint and no lateral correction is required. This algorithm works realigning constantly the vehicle with the next waypoint. If a perturbation push the vehicle outside the desired route the vehicle

will not try to return to the previous path, but rather generate a new one. This new path will be a straight line between the current position and the desired waypoint. Clearly this is a waypoint tracking algorithm rather than a route tracking algorithm. However, this algorithm can be used as a base for a more complex route tracking. A set of extra waypoints are placed to increase the waypoint density on the route. This denser route will yield a better route tracking as shown in Figures 4.8 and 4.9

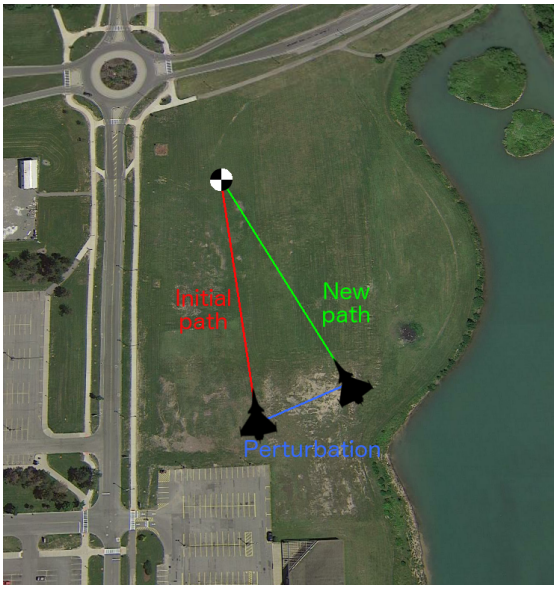


Figure 4.8: Single waypoint tracking

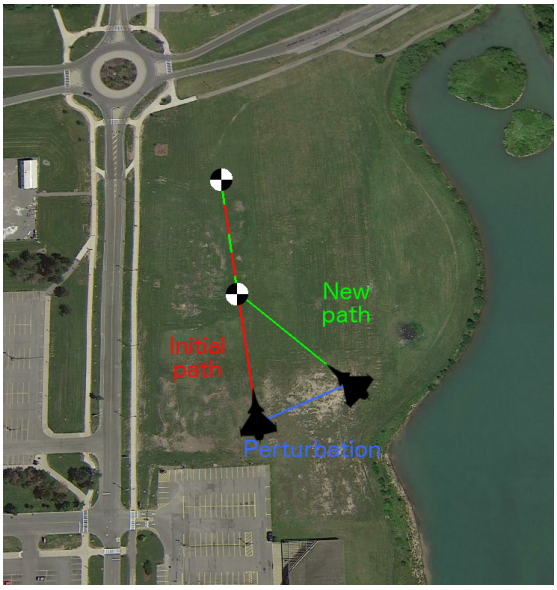


Figure 4.9: Multiple waypoint tracking

$$e_\phi = \frac{e_{b_1}}{\mid e_{b_1} \mid + \mid e_{b_2} \mid} \tag{4.6}$$

This filter is independent of the vehicle. The only requirement is the orientation control capability. For an UGV, the output of the controller is connected directly

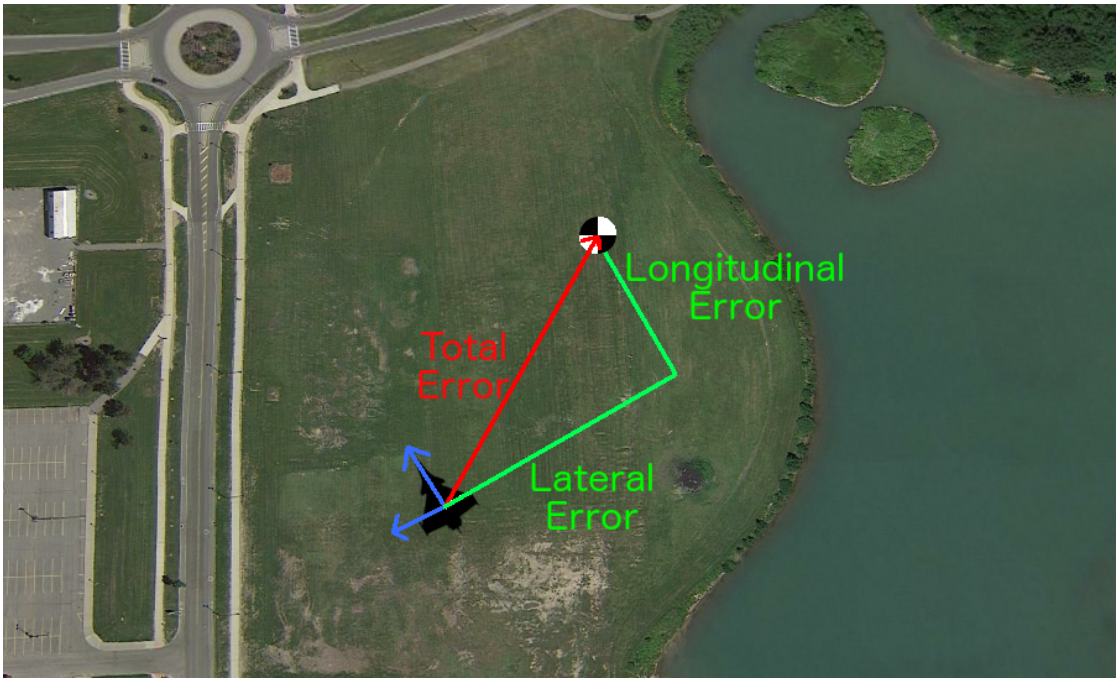


Figure 4.10: Error projection in body frame

to the steering wheel. For a fixed wing UAV, the error is used to control the rudder; or it can be used to control a mixed channel between the elevator and the ailerons to perform a sharper turn. To control how sharp the turn is, the error signal is scaled before applying it to the surface control. For quad rotors or helicopters the errors e_{b_1} and e_{b_1} can be directly fed to roll and pitch respectively.

This section introduced the algorithm used to track waypoints. The next section discuss all the required ground test required to validate the waypoint tracking algorithm.

4.4 GROUND TEST

Before testing the entire system on a flying vehicle, a set of ground test are required to ensure the safety of the aircraft and the surroundings of the operation. In this section we present an application of the SAI where the systems is used to control a ground vehicle. This experiment not only acts as a proof of concept for the Aircraft autopilot but also shows how effectively repurposed to be used on different platform with minimal reconfigurations.

The ground vehicle used is shown in Figure 4.11. The vehicle is powered by two acid-lead batteries. The control and propulsion of the vehicle is achieved with four DC motors. The motors controller is a simple PWM mixer and a PWM to analog converter. Two control signals are used as commands; the first controls the speed and the second the yaw speed. The PWM controller of the vehicle is connected to the autopilot selector, which acts as a switch to allow multiple control sources. A Beagle Bone Black (BBB) Single Board Computer (SBC) is used as the main control device, and an arduplane is used as the secondary control device; both, connected to the switch. The RC receiver is connected to the switch as selector and controller. The communication between the main ground control station and the SAI is done using a radio modem. The electronics are powered with a different battery than the motors to reduce the electric noise on the sensor and data acquisition components. A GPS receiver is connected to the boards to allow position acquisition. A copper plate is used to shield the GPS antenna and magnetometer; this sheet is glued to a plexiglas sheet of similar dimension to increase the stiffness. The GPS antenna

and the magnetometer are mounted on top of an aluminum column to increase the distance to the motors, battery, and modem antennas to reduce interference.

The magnetometer, accelerometer and the gyroscope needs to be calibrated. The measurements of this sensors can be modeled as second order Markov process[104] as shown in Equation (4.7) with $\tilde{x}$ being quantity measured, x the physical quantity, K_x initial offset, η_x noise β_x bias. The bias differential equation is given by Equation (4.7) where K_β is the bias offset and η_β is the bias noise.

$$\tilde{x} = x + K_x + \eta_x + \beta_x \tag{4.7}$$

$$\dot{\beta}_x = K_\beta + \eta_\beta \tag{4.8}$$

Barshan and Whyte[105] proposed a solution to the previous equation where the error is given by Equation (4.9). In this fashion, the measurement equations reduces to Equation (4.10). A calibration process is required to obtain the coefficients that characterize the sensor. c_1 is a coefficient that changes on each operation, to estimate it is required to sample the sensor on a know initial condition and average the first n samples on each operation. c_2 and c_3 remains sensible constant. To estimate them, data from the sensor should be gather for a sufficiently long period of time, then using a least-square solution the the coefficient values are estimated.

$$\epsilon(t) = c_1(1 - e^{-\frac{t}{c_2}} + c_3) \tag{4.9}$$

$$\tilde{x}(t) = x(t) + c_1(1 - e^{-\frac{t}{c_2}} + c_3) + \eta_x(t) \tag{4.10}$$

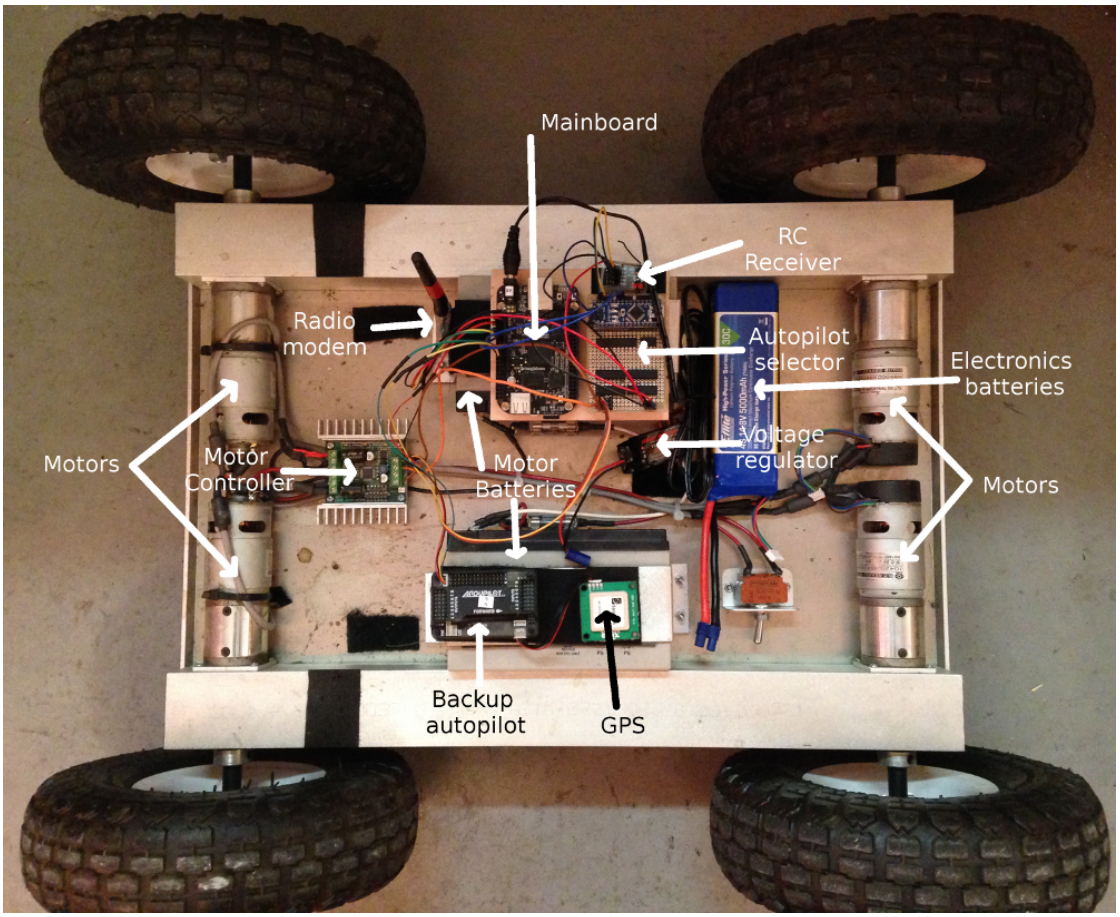


Figure 4.11: Ground Vehicle experimental layout

The GCS configuration is presented in Figure 4.12. The MP generates the route based on predefined rules. This rules are a set of routes to test the autopilot capabilities such as turn left, turn right, return to home. The ATC in this experiment

only forward the routes to the PRC. The PRC acts as a data logger for the SAI.

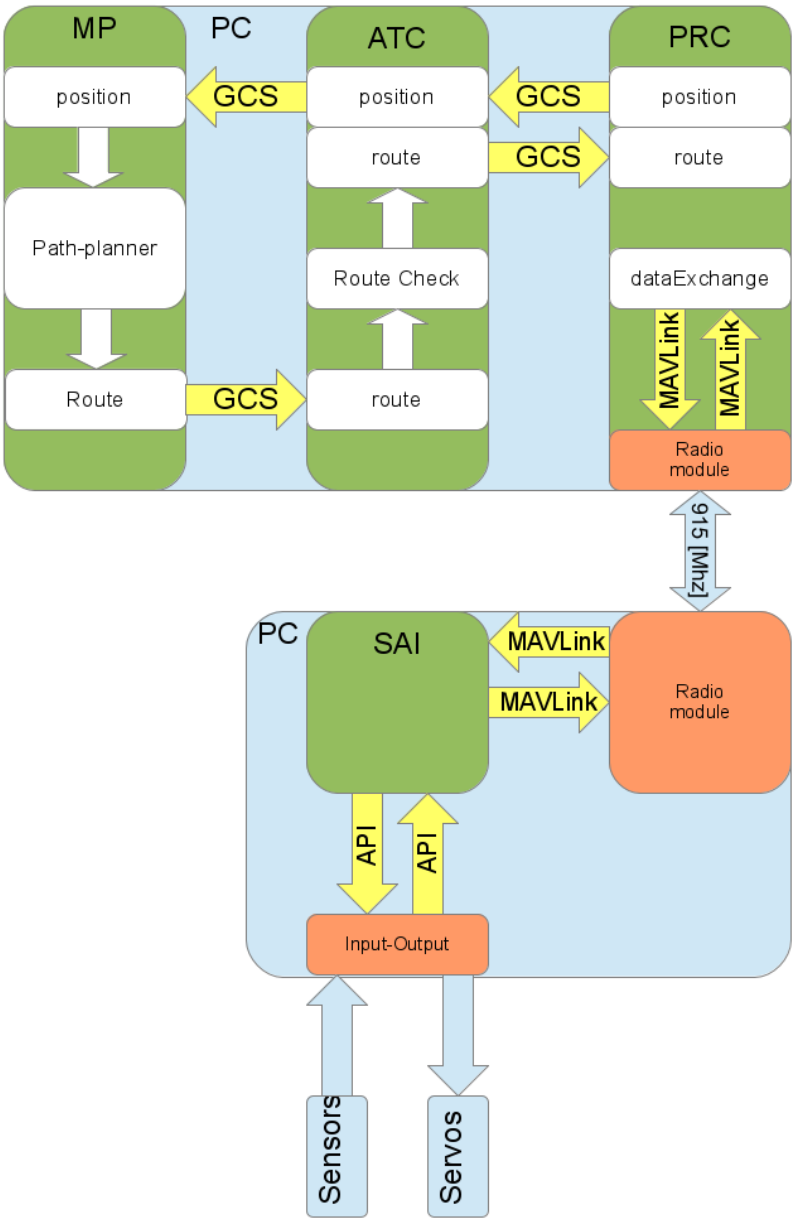


Figure 4.12: UGV Modules requirements

The SAI configuration for this experiment needs to have the entire set of modules and configurations set to perform correctly. The selected modules are presented in Figure 4.13. The communication is done over a radio modem with no message delivery guarantees. The Comm modules that perform the communication exchange, for this particular experiment, needs to be robust. The MAVLink[35] protocol is used; the data stream is acquired and set through a serial port connected to the radio modem. This protocol implements message check, recovery, and queues. The hardware abstraction layer implements the BBB API in a C++ environment. The HAL module can access the data sensor, which is a serial port GPS module, and an I2C IMU. The control is done using PMW outputs. These PWM outputs control the motor controller. The navigation implements a simple filter to obtain magnetic bearing and a smoother is implemented for the GPS position measurements. The guidance algorithm maps the error in position in the body frame. The error in position is computed subtracting the required position and current position. The controller is composed of a set of PID's. A PID is used to connect the yaw error to the steering control. A second PID with X error as an input is used to control the throttle setting. The route tracking algorithm is a simple queue-dequeue list. The first waypoint on the list is assigned as the desired destination; once the destination is reached the waypoint is removed from the list and the next waypoint is assigned as new destination. Once the last waypoint is reached, the vehicle stops.

To test the entire layout a route is sent to the UGV. The outcome of the route tracking algorithm is shown in the map Figure 4.14. he waypoints window are marked in blue for the SAI and in red for the GCS. The black line represents the

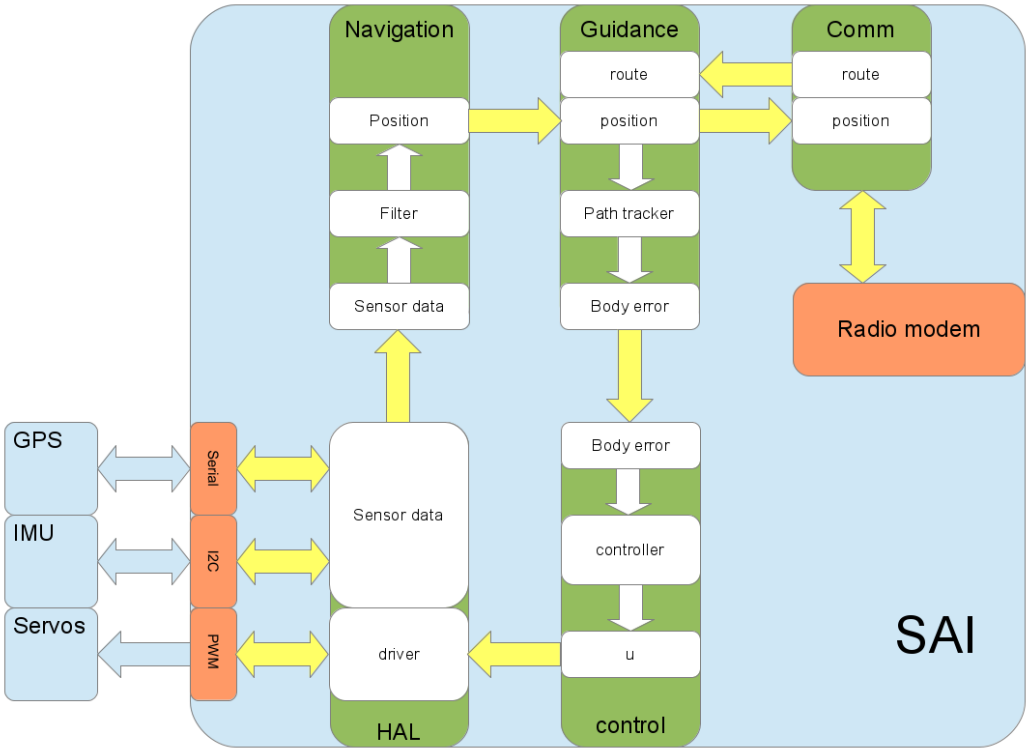


Figure 4.13: SAI modules layout

vehicle trajectory. For this experiment when the vehicle reach the last waypoint, it begins to circle. This behavior, is required for the airplane, since it is not possible to stop midair. The drift from the original trajectory is due to the terrain elevation and slippery surface. It can bee seen however, that after the drift the vehicle keeps reorienting itself to the objective. The white arrows clearly show how the vehicle reorients itself to the waypoint rather than to go back to the original path. Clearly, if the way points were placed closely, the vehicle tracking error is further reduced.

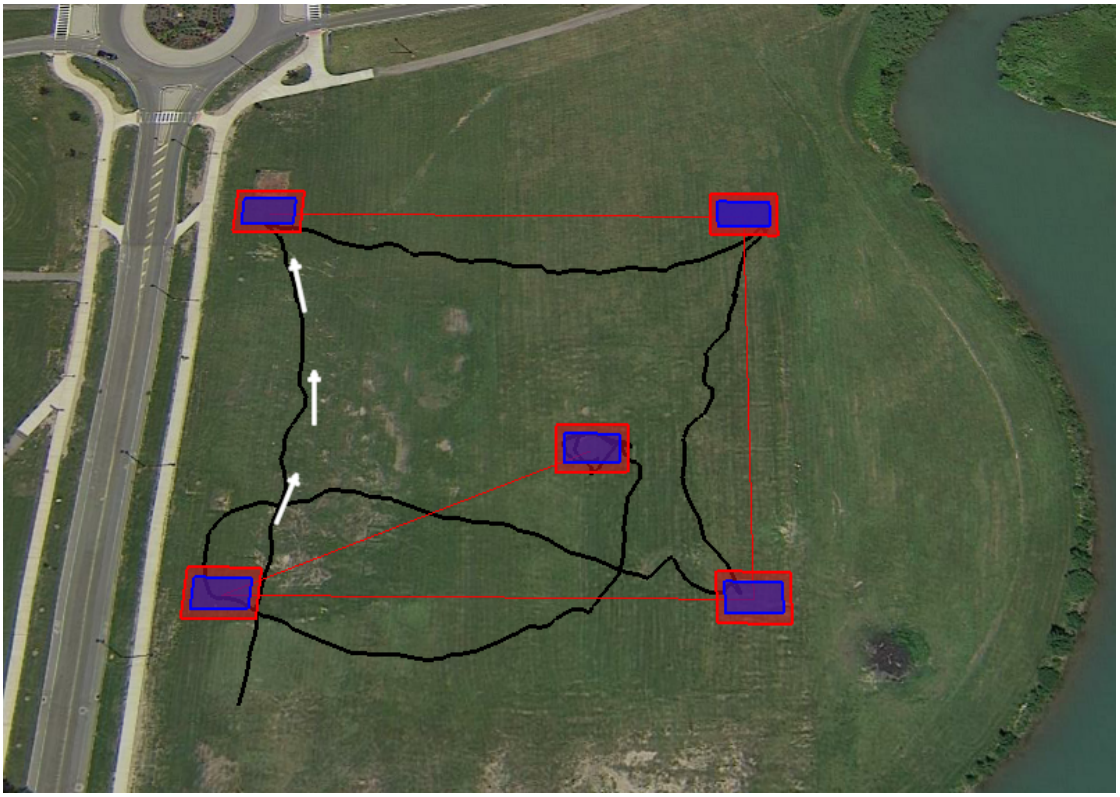


Figure 4.14: UGV path track result

4.5 FLIGHT TEST

After all tests have been passed, the UAV is ready for the flight. The vehicle used is a Great Planes Funster, shown in Figure 4.15, and as described in the previous sections. The vehicle has installed the same equipment as described in the previous section. It carries a GPS, backup autopilot, and autopilot selector. An IMU is used to estimate the vehicle attitude. The selected test flight area is the Lake La Salle at the University at Buffalo. The airplane is flown on a clear day with no wind. The initial test flights, requires to have the minimum possible source of perturbations.

Figure 4.15: Great Planes Funster

The controllers, designed in the previous chapter, is used to perform the attitude control of the UAV. The waypoint tracking algorithm used is the one presented in Section 4.3. The attitude control output is shown in Figures 4.16 and 4.17. The green lines represent the flight cruise condition. The zero value represents the θ_0 and the ϕ_0. To test the different controllers the plane is set to fly in cruise condition by the pilot. Once the plane is in cruise condition the autopilot is engaged. While the autopilot is engaged perturbations are introduced using the manual inputs from the pilot. An asymmetric deflection on the ailerons is used to generate a large deviation from the cruise condition. The autopilot, recovers the cruise condition as can be seen in Figure 4.16. The same procedure is repeated with the elevator to generate

a perturbation on the pitch angle. Again, the autopilot restores the flight condition as can be seen in Figure 4.17.

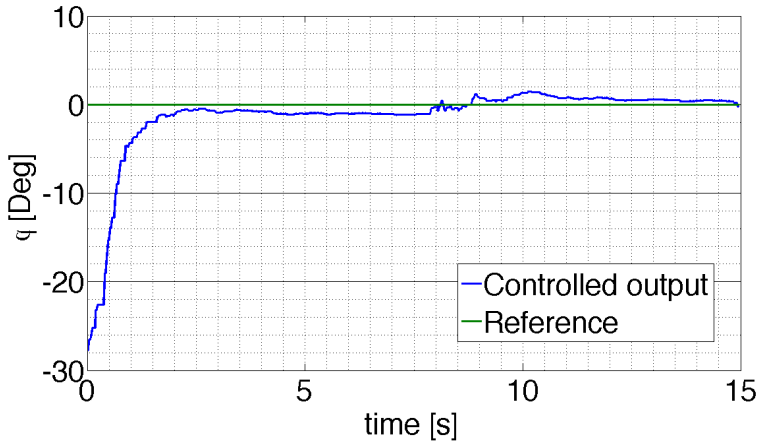


Figure 4.16: UAV roll tracking

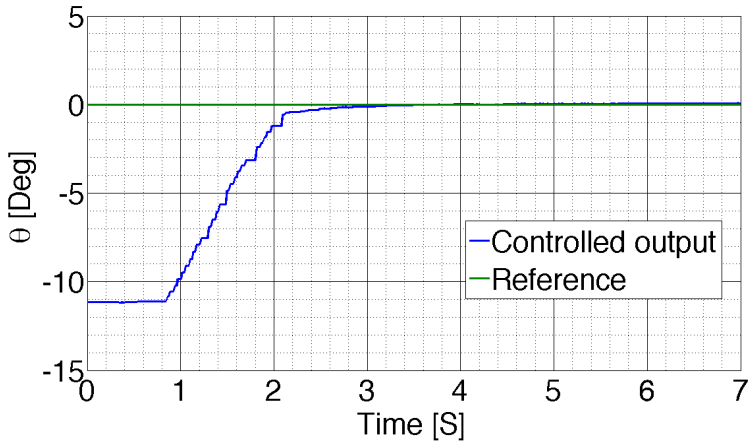


Figure 4.17: UAV pitch tracking

4.6 CONCLUSIONES

In this chapter the practical implications of autopilot implementation are discussed. First a robust design for an autopilot switcher is presented. The method to override all automatic controls is then discussed. The autopilot switcher provides a method to aid the pilot in a manual flight for unstable vehicles such as quad-rotors.

An algorithm used to track waypoints is discussed in Section 4.3. This algorithm, then, is adapted to track routes by holding a first in first out waypoint queue.Then, the results of the waypoint tracking algorithm and controllers are presented on a UGV. The results from the UGV experiment validate the filter design.The test also acts as proof of concept for the aircraft flight. The waypoint loiter circular pattern is tested.

The considerations regarding sensor mounting and electromagnetic shielding are presented in Section 4.4. In this section,mounting of the GPS antenna, radiomodem antenna, and magnetometers is discussed, and the layout to reduce interference is presented.

Finally the controller designed in the previous chapter is tested on a flight and the output presented. The output of the test flight not only validates the controller design, but justifies the model linearization and the coefficients therein.

Chapter 05

MICROSATELLITE ATTITUDE MANIPULATOR

Houston, we've had a problem [...]

John 'Jack' Swigert, Jr.

5.1 INTRODUCTION

Attitude determination and control of spacecraft is important for mission operations, owing to the precision pointing requirements for effective operation of scientific instruments, communication and power related equipment. A wide variety of precision attitude determination and control algorithms have been implemented in the past, as evidenced by the technical details on the state of practice for attitude determination and control used in the development of the Navy's Transit system (a precursor to the popular Global Positioning Satellite (GPS) system, circa 1960). Owing to the small rotational rates of spacecraft, a variety of dynamics based control systems can be designed and implemented effectively utilizing modest computational resources. Optimal open loop and stable feedback solutions have been

implemented for spacecraft attitude control by rigorous application of Pontryagins principle (cf. open loop minimum energy solutions in Junkins and Turner,[106] minimum time solutions in Bilimoria and Wie[107] and nonlinear optimal feedback solutions in Carrington and Junkins[108]) and Lyapunov stability theory (cf. Vadali et.al,[109;110] Tsitotras[111], Wie and Barba[112] Crouch,[113] Wan[114] and Dwyer[108]) for stabilization of nonlinear dynamical systems.

To generate the torques required for precision attitude maneuvers, a variety of actuation mechanisms are employed. Magnetic actuators, including torque rods provide significant torque at the expense of some renewable energy (and communication downtime). Large-scale attitude maneuvers are accomplished using Reaction Control Jets (RCJs) that are unsuitable for precision pointing and higher accuracy reorientation maneuvers. Momentum exchange devices are typically the actuators of choice of space crafts for precision pointing and attitude control and Reaction Wheels. NASA standard configuration of 4 reaction wheels (cf. Junkins and Turner,[106] Hughes[115]) and the pyramidal configuration of wheels arranged to rotate on the four faces of a pyramidal structure (cf. Wie,[116] Schaub and Junkins[117]), are among the prominent mechanisms to provide torque and manage momentum in spacecraft. In fact, the International Space Station (ISS) , among other currently operational spacecraft, uses a combination of control moment gyroscopes (momentum wheels mounted on a pyramidal configuration) and thrusters for fine pointing and gross changes in attitude respectively. The single gimbal control moment gyroscopes (SGCMGs) arranged in a pyramidal configuration has the advantage of redundancy in actuation. This means that for every target torque

required for spacecraft maneuvers an infinite number of orientation solutions exist for each SGCMG to generate the commanded reference torque. However, certain arrangements of actuators produce no torque and a subspace of the SGCMG gimbal angles exists that produces no net change in angular momentum of the SGCMG cluster. Such configurations are said to be singular and researchers have carried out significant investigations to understand the geometry of the attitude control problem with singular configurations.[116;117] Singularity robust solutions for momentum management of spacecraft actuated by a cluster of SGCMGs were developed by Bedrossian,[118] utilizing Nakamura's classical results for robot manipulator control.[119] Krishnan and Vadali further extended this approach by proposing an inverse free technique for singularity avoidance.[120] Subsequent developments aimed at characterizing the subspaces and utilizing the CMG null motion to overcome actuator singularities.[121–124] Vadali et al.[125] provide an elegant development where a preferred set of initial gimbal angles are provided to avoid singular configurations for spacecraft attitude tracking maneuvers. This extensive research made way for recent developments in Variable Speed CMGs (VSCMGs) and associated feedback control techniques to enable singularity free attitude control and momentum management in spacecraft.[126]

Although significant body of literature exists for control of spacecraft using a cluster of SGCMGs and the VSCMGs, the utilization of robotic manipulators for enhancing the effectiveness of the gimbals used in manipulating the gyroscopic torques generated by steering the wheels has not been explored extensively. This is because of the fact that the relative strength in keeping a simplistic gimbal

design and maintaining low inertias in the supporting structure to realize a simple actuator. The advent of recent microelectronics, computing and modeling methods have driven the development of complicated mechanisms for a variety of industrial and research robotic applications. These advances have spill over effects in the development of the next generation gimbaling systems, there-by contributing to the advancing the performance of the CMGs developed using these advanced gimbaling mechanisms. The emergence of novel micro momentum management devices such as the Hemispherical Resonating Gyroscope (HRG), among other similar technologies, is a testament to this fact. Continuing this emerging technology trend, we propose the development and control of a Dual Gimbal Control Moment Gyroscope by utilizing a novel carpal wrist joint in conjunction with a momentum wheel.

In this research, we incorporate a novel Carpal Wrist Joint (CWJ)[127] known as Canfield joint. This joint is composed of a parallel[128–130] closed kinematic chain. The joint is driven by three inputs which are the angular position at the base of each parallel segment. With the inputs, the outputs of the joint given by azimuth, elevation, and plunge distance of the end effector are controlled. However, when the CWJ is used as a gimbal in pointing applications only two of the three degrees of freedom are required; hence it can be treated as a redundant[131] parallel actuator.

Parallel manipulators are susceptible to singularities. Singularity is defined as a configuration where the actuators are unable to affect the end effector position by differentially manipulating the input joints.[132;133] Mathematically, a singularity is described by the Jacobian of the kinematic transformation that maps the inputs to the outputs loss of rank. Canfield in his dissertation developed the CWJ to

have a singularity free semispherical workspace. It is thus, possible to achieve a trajectory connecting any two points on the surface of the sphere, while maintaining a constant plunge distance.

To support this research in attitude control, we have developed several prototype joints to serve as a gimbal using off-the-shelf servos, that have embedded position control. This micro joint is used to build a momentum manipulator and is mounted on a satellite emulator to test the joint capabilities in a micro gravity environment. Analytical expressions for direct and inverse kinematics of the distal end as a function of the primary joint angles are used to derive control laws to manipulate the joint.

Using a controller that is driven by the basal angle, to control the upper plate has the main disadvantage that the controller can not compensate for mechanical misalignment, error in assembly and joint slips. To overcome these issues, a vision-based feedback controller[134;135] is implemented. With the use of basic image processing techniques such as the robust feature extraction and matching methods,[136–139] features on the reference plane end-effector are extracted. The spatial position of the features is then obtained by solving the homography problem.[135] A new approach to solve the homography with a least squares technique is presented. The least squares solution is further developed to be used as a measurement equation in a Kalman[140] filter.

One of the key aspects that needs to be investigated to evaluate the capabilities of the proposed device against more conventional dual gimbal systems is the computation of uncertainties in the pointing mechanism. Extensive studies have been

carried out to understand the pointing uncertainties in parallel manipulators due to design and tolerances. Kiridena[141] has develop a method to model and visualize the effects of the position error of the axes of a machine and the propagation through the kinematic chain. Notash and Podhorodeski[142] shown that even if all the driven joints of a manipulator are certain, there may still be uncertain (lock) configurations. Frequently, these configurations correspond to singularities of the mechanism or its realizations. A solution to this problem, is to add extra branches to the manipulator. In this chapter, we present a study on how the input uncertainties are propagated to the azimuth and elevation of the joint. As opposed to the research based on interval analysis done by Rao,[143] Tannous[144] among others[145;146] we focus our analysis on a probabilistic approach.[147] First, analytical expressions are derived that can be used to measure the uncertainty in any open or closed kinematic chain. It is also identified that the recently developed Unscented Transform[148] (UT) can also be used for uncertainty quantification. The two methods are subsequently validated using MonteCarlo simulations to determine the region of validity of the linear error theory.

The uncertainty propagation methods are used to evaluate an open and closed kinematic chain design for a 2 degree of freedom (DOF) gimbal for attitude manipulator design for comparison purposes. A satellite attitude hybrid emulator is presented and its design is discussed. This device consists of a pendulum and a weight system distributed to obtain a dynamics equation equivalent to those of a satellite attitude dynamics. The emulator is designed to be mounted on vehicles that can perform microgravity flight tests. To this end, a set of features are added

to assure the system integrity throughout the entire experiment. These features include a locking mechanism to prevent movements during take of and landing; a safety switch to cut of the electrical supply on an emergency; and linear actuators to remotely perturbate and give initial conditions to the emulator. Using the SAI and GCS it is possible to interface the experimental layout with FlightGear to perform a more realistic simulation.

5.2 CARPAL WRIST JOINT

The Carpal Wrist Joint (CWJ) is a novel three degree of freedom parallel manipulator developed by Steven Canfield.[127] The joint consists of 8 links, a parallel actuation scheme similar to the flexor and extensor carpi muscles along the forearm, and an open interior passage, which forms a protected tunnel for routing hoses and electrical cables, much like the well-known carpal tunnel on the human wrist. A CAD model is shown in the workspace chart of Figure 5.1. The device workspace is hemispherical and singularity free as shown in Figure 5.2. The workspace is parameterized in terms of the plunge distance (p_d). The p_d is the distance from the center of the wrist to the upper plane. For each plunge distance a spherical workspace is generated with a radius of p_d.

Using feedback control the CWJ can be used as a pointing device for various applications such as robotic manipulation, satellite attitude control, and robotic stabilization, among others. For precision pointing applications, the angles between the legs and the basal plate (fixed) are driven by a set of servos that control the distal plate. The distal plate has 3 degrees of freedom that consist of the elevation

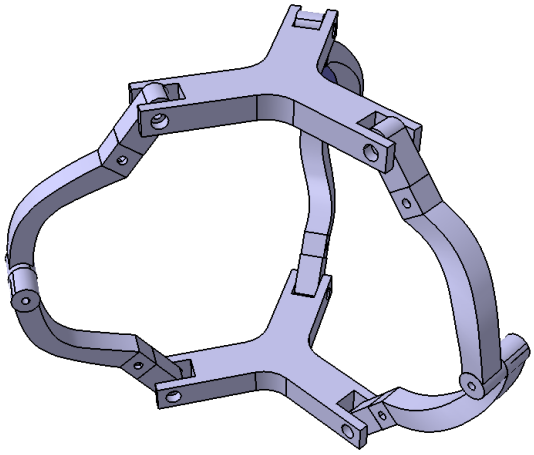


Figure 5.1: CWJ CAD Model

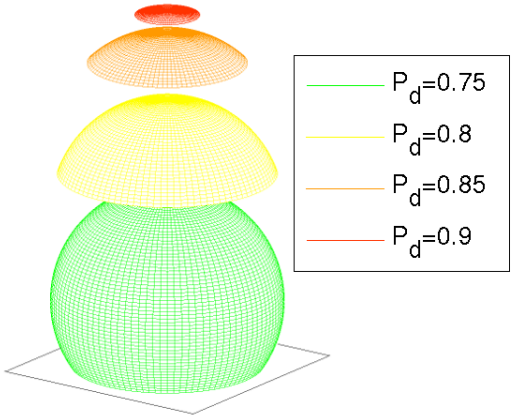


Figure 5.2: CWJ Theoretical workspace

angle α, azimuth angle β and the plunge distance P_d. It is therefore a displacement that is an implicit function of the distal end orientation (measured in body frame). The plunge distance refers to the distance from a fictitious plane of symmetry between the plates to the upper or lower plate, both being equivalent.

Potentiometers and encoders respectively provide direct and indirect measurements of joint angles at the basal plate for feedback control. These quantities are propagated through the kinematic model to determine distal plate attitude. The device built for the micro gravity experiment uses off the shelf RC servos which have built-in position control drivers; with a Pulse Width Modulation (PWM) signal it is possible to control the absolute position of the servo angle.

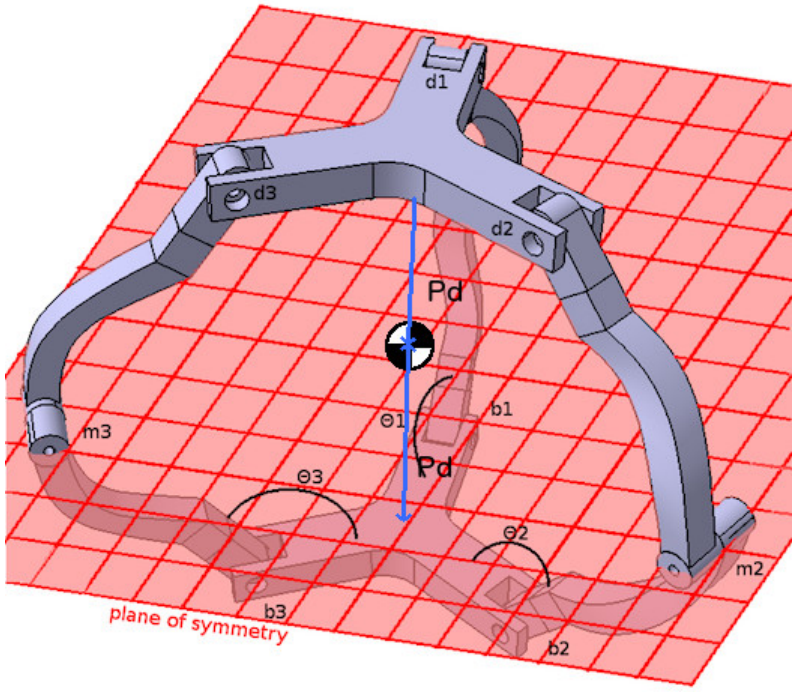


Figure 5.3: CWJ plane of symmetry

To solve the forward kinematic model, (i.e. obtain the distal plate attitude and plunge distance, given basal angles) the key feature exploited by Canfield is the fact that the joint posseses a symmetry plane between the basal and distal plate that intersects the middle revolute joints, as shown in Figure 5.3. With the location

of the basal revolutes and their corresponding angles known, the location of the mid-points (m_i) of each branch can be found using Equation (5.1). Here $\boldsymbol{R}_{[u_i,\theta_i]}$ represents the rotation of θ_i degrees about axis $\boldsymbol{u}_i$. $\boldsymbol{q}_i$ represent the direction of the branch. $\boldsymbol{b}_i$ gives the distance from the center of the basal plate to the revolute joint.

$$\mathbf{m_i} = \mathbf{R}_{[\mathbf{u_i},\theta_\mathbf{i}]} l\boldsymbol{q}_i + \boldsymbol{b}_i \quad i = 1...3 \tag{5.1}$$

Knowing the location of the m points, the mid-plane can be determined using Equation (5.22). Where A_m, B_m, C_m are the standard coefficients for a plane.

$$[A_m, B_m, C_m]^T = (\mathbf{m_2} - \mathbf{m_1}) \times (\mathbf{m_3} - \mathbf{m_2}) \tag{5.2}$$

We need to compute the distance from the $\boldsymbol{b}_i$ points to the mid-plane using Equation (5.3). The distance is given by $\boldsymbol{\delta}_i$. $\boldsymbol{N}_m$ represents the normal direction to the mid-plane.

$$\boldsymbol{\delta}_i = \boldsymbol{N}_m \cdot (\mathbf{m_i} - \mathbf{b_i}) \tag{5.3}$$

The distal revolute joints ($\boldsymbol{d}_i$) can be obtained exploiting the joint symmetry using Equation (5.4).

$$\boldsymbol{d}_i = 2\boldsymbol{\delta}_i \boldsymbol{N}_m + \boldsymbol{b}_i \tag{5.4}$$

Finally the center of the distal plate $\boldsymbol{C}_d$ can be obtained from the overage of the three distal revolute as shown in Equation (5.5). A similar approach can be

used to solve the reverse kinematic model.

$$\boldsymbol{C}_d = \frac{\boldsymbol{d}_1 + \boldsymbol{d}_2 + \boldsymbol{d}_3}{3} \tag{5.5}$$

5.3 CARPAL WRIST JOINT MECHANICAL DESIGN

In this section we present two CWJ built in the AR Laboratories. These devices have been used to test different controllers.

STRUCTURAL DESIGN AND PROTOTYPING

The first device was machined from an aluminum block using a CNC machine Figure 5.4. Shielded ball bearings and steel hardened shafts are used in every articulation of the joint to reduce the friction between parts with relative motion. The joint is driven by three DC motors. The motors have a planetary gearbox reduction in the front side reducing the speed and increasing the torque. An aluminum helical coupling is used to link the motor gearbox to the shaft driving the basal plates revolute joints. The rear shaft of the motors has no reduction.

ELECTRICAL DESIGN

The motors have a controller that uses hall effect sensors as feedback to achieve speed control. The desired speed is set with the use of a PWM signal. With the use of encoders, a position control is achieved in an outer loop that runs on the

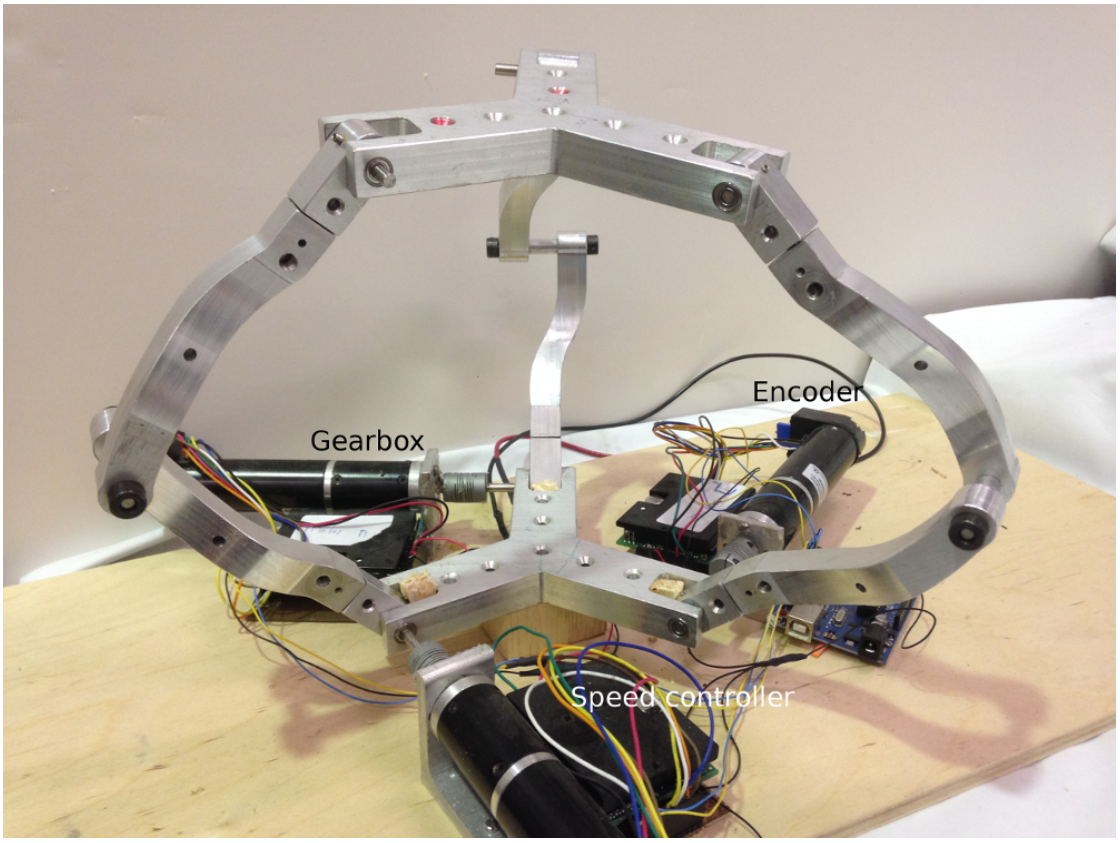


Figure 5.4: Carpal Wrist Joint Prototypes and Control design

control computer. A National Instruments 6351 data acquisition board is used to control the motors and read the encoders.

SENSOR SYSTEMS AND FEEDBACK CONTROL

The realization of the SAI architecture was utilized to control the prototypes is shown in Figure 5.5. While providing a high accuracy for precision pointing, this realization was formed to have low actuation bandwidth owing to joint metrics.

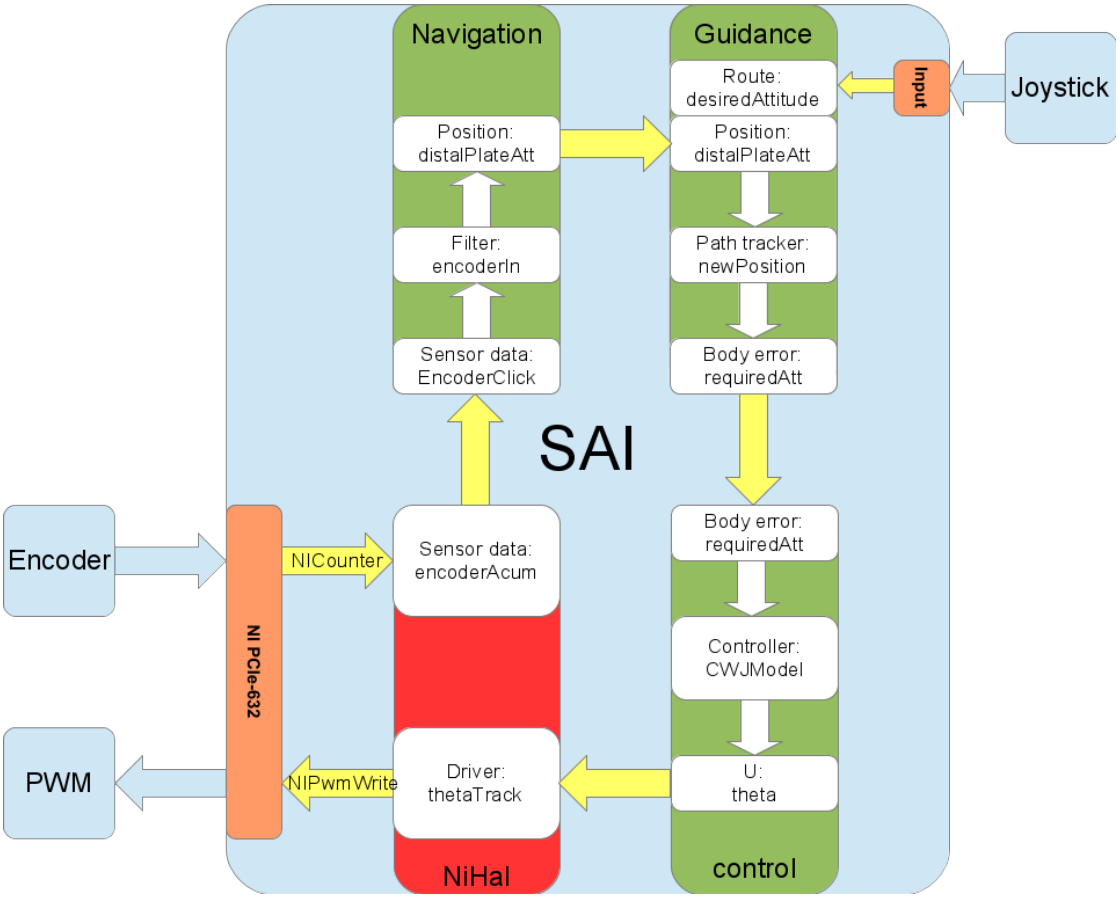


Figure 5.5: satellite controller software layout

The navigation module converts encoder clicks to distal plate position and orientation. The encoders measurements are integrated to obtain the absolute angular position of the basal angles. The basal angles are converted to radians using the pulse per revolution constant of the encoder. The forward kinematic model is used to estimate the distal plate position. The guidance module performs the distal plate attitude tracking. With the estimate provided by navigation and the input desired by the user the guidance module solves the next required attitude. The

controller module takes as input the error signal generated by Guidance and sets the required basal angles. The HAL is the block required to read encoders and write the motors position. For the National Instruments Boards (NiHal) implementation this modules integrates encoders based on a fixed clock. This block implements a model-free Proportional, Integral and Derivative (PID) control logic on the tracking error to achieve a precise angular position on the motors. In Embedded Multimedia Figure 5.6 a demo video of a manual control of the joint is shown.

Figure 5.6: CWJ joystick controlled

PROTOTYPE ATTITUDE MANIPULATOR SYSTEM

The prototype of a miniature joint for spacecraft attitude control applications is shown in Figure 5.7. This compact joint is driven by RC Servos. Steel shafts are used for the legs. The joint articulations are machined from a square rod and bearings are used to reduce friction. A inertia disk mounted on a brush-less motor attached to a speed controller is used as momentum wheel.

To demonstrate the flexibility and versatility of the proposed autonomous dynamical system controller, we now briefly expound upon the required software changes to control the prototype devices. Note that this device is conceptually completely different from the UAVs or other vehicle platforms.

The only software module that requires an update is the HAL module. For this design a set of RC Servos are used. The angular position is controlled with a pulse. The pulse width ranges from 1 ms to 2 ms. The HAL generates the required pulse based on the controller input. Also set the equivalent angular position for the navigation module. A video showing the control of the joint is present in Embedded Multimedia Figure 5.8. The small joint, in contrast to the large joint discussed earlier possesses high bandwidth while maintaining the precision achieved by the larger joint.

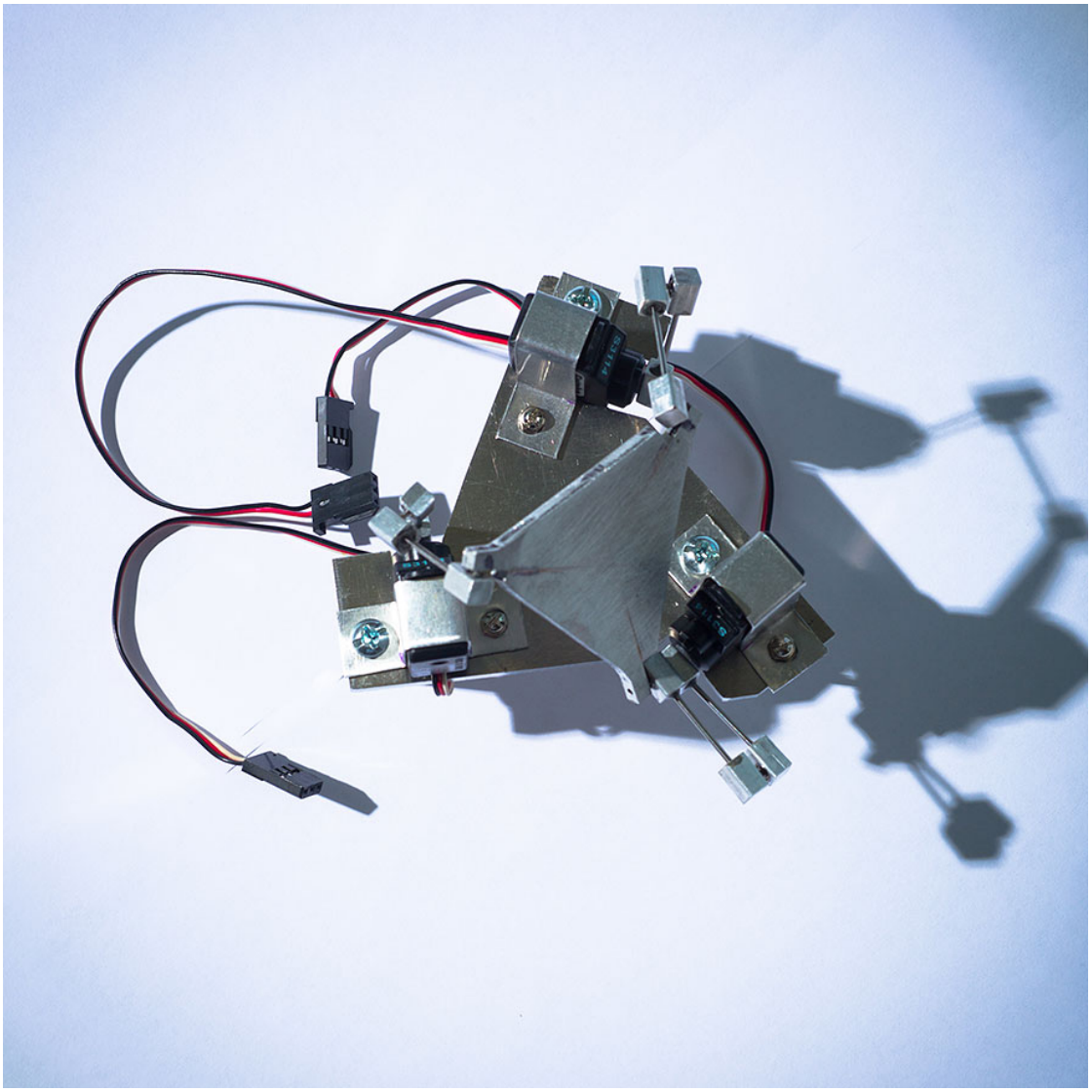


Figure 5.7: Attitude manipulator prototype

VISION BASED ATTITUDE ESTIMATION

Using feedback controller based on the revolute joints at the base leads to end effector pointing errors owing the mechanical play and slip in the intermediate joints.

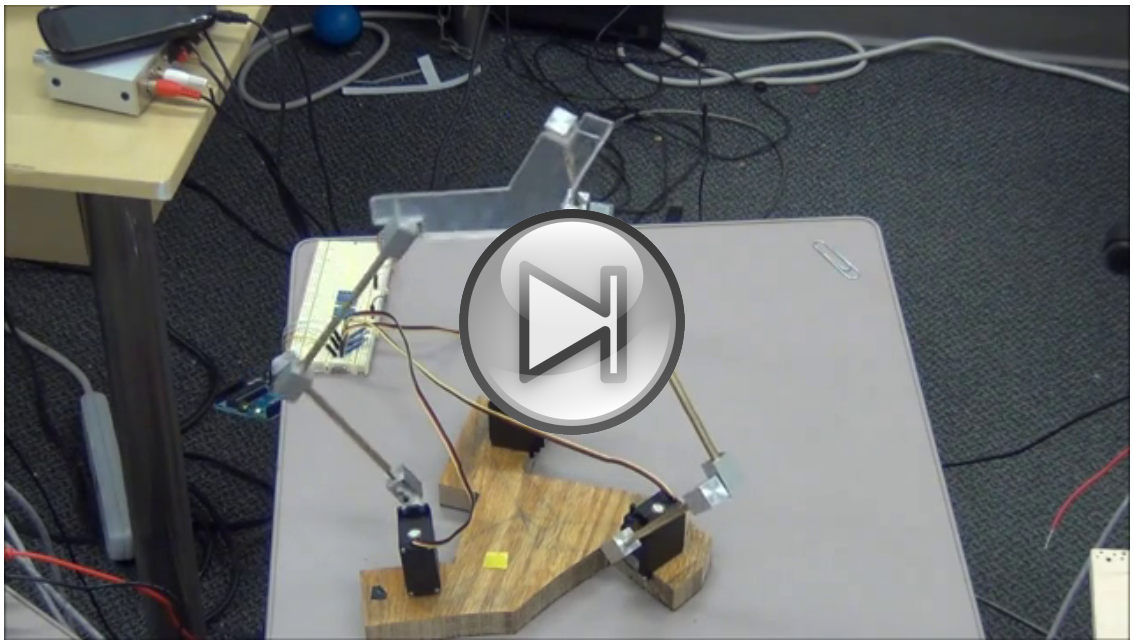


Figure 5.8: CWJ actuated by servos

The kinematic drift associated with mechanical misalignments is unavailable for measurements for compensation by the control system. The error propagation is nonlinear owing to the parallel kinematics of the joint. This leads to the large deviations of the estimated distal plate orientation from the true value, if left unaccounted for. To evaluate how this orientation error affects the pointing precision, a vision based method[134] is used to measure the upper plate attitude . This measurements are then compared to the encoder measurements to understand the error from the revolute joint and the actual distal plate position. Using image processing techniques, known references points lying on the distal plane are identified. Solving the homography problem yields the space location of the references points. Using algebraic manipulations the point distribution is used to obtain the distal

plate attitude. The image processing techniques are presented in Appendix B. The homography problem and solution is discussed in Appendix C.

KALMAN FILTER

In the previous section we presented the approach to estimate the pointing direction of the joint using an image processing technique for feature extraction. Although this method gives us a position estimate at each time step, we can improve this estimate by using popular filtering techniques.

If the dynamic model of the joint is known,[127] and measurement uncertainties can be characterized, these two pieces of information can be leveraged to derive an optimal position estimate, along with other states of interest. To achieve this estimation, an extended Kalman[140] filter is implemented. For this particular application we use the approach presented by Crassidis and Junkins.[69;149] The joint is considered as a body under constant acceleration. Its assumed that the attitude can be directly measured. In Appendix D An Extended Kalman filter and its implementation are discussed. The results for a sample trajectory are shown in Figures 5.9 and 5.10

GENERALIZED KINEMATIC MODEL

The kinematic model presented in the previous section, although providing a closed-form solution that is easy to implement, has two main deficiencies: first, it does not account for the system mass and inertia, and hence cannot be used to esti-

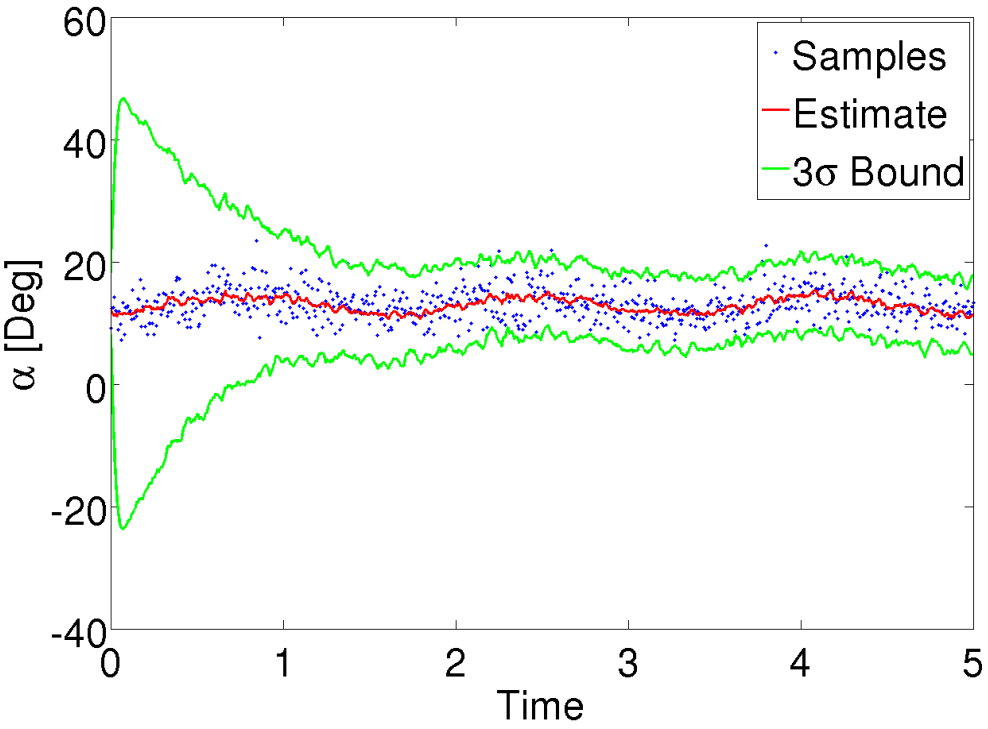


Figure 5.9: Kalman filter elevation estimate

mate system dynamics or its dynamic response; second, it requires the joint to be symmetric on the legs and angles which limits the design envelop for the joint. A more general and flexible approach is presented in this section, however it lacks the analytical appeal of the former approach, in that a closed-form solution cannot be found. The presented method is widely used in the analysis of complex closed-chain parallel manipulators.[128;150] Each leg can be modeled as a series manipulator,[151] i.e. a chain where each link is rotated over only one angle from the previous link. Each link position then is modeled as a rotation and translation over the $\boldsymbol{k}_i$ axis as shown in Figure 5.11. For each of the legs of the joint the position of each of the

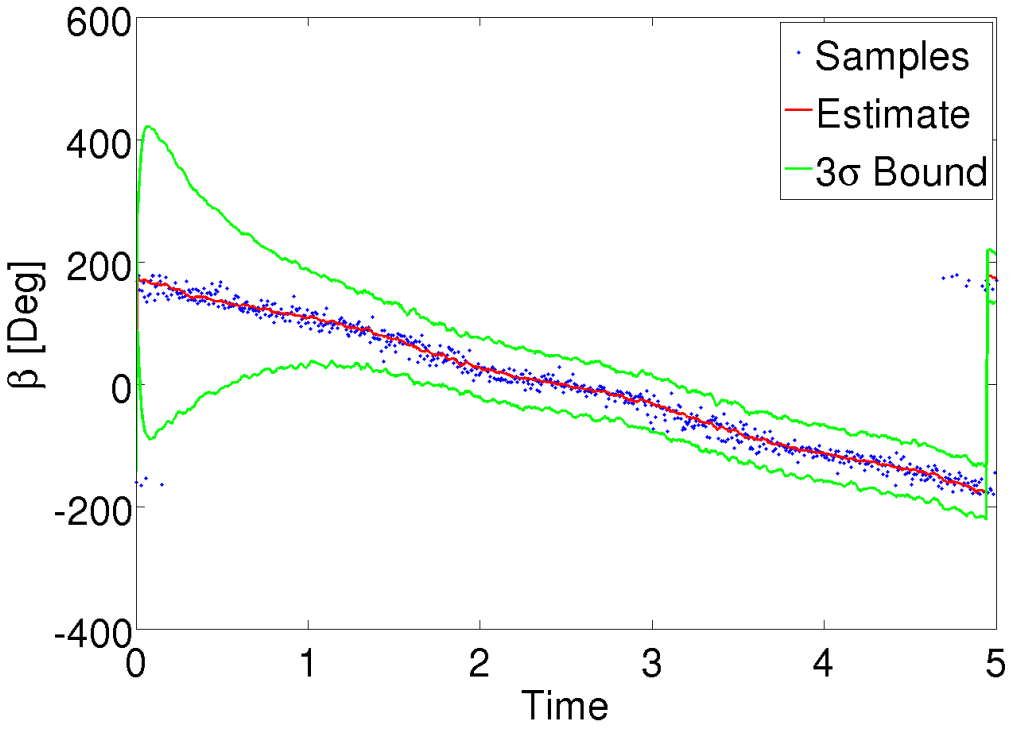


Figure 5.10: Kalman filter azimuth estimate

links is given by

$$\boldsymbol{e}_i = \left(\prod_{j=0}^{i} [R_{\theta_j}] \right) \boldsymbol{e}_j \tag{5.6}$$

$$\boldsymbol{r}_i = \sum_{j=0}^{i} l_j \boldsymbol{e}_j \tag{5.7}$$

where $\boldsymbol{r}_i$ represent the length of each link and $[R_{\theta_j}]$ represent the corresponding rotation matrix. From Equation (5.7) it can be seen that with this model each link component can have an arbitrary length is required. Also θ_0, represents the

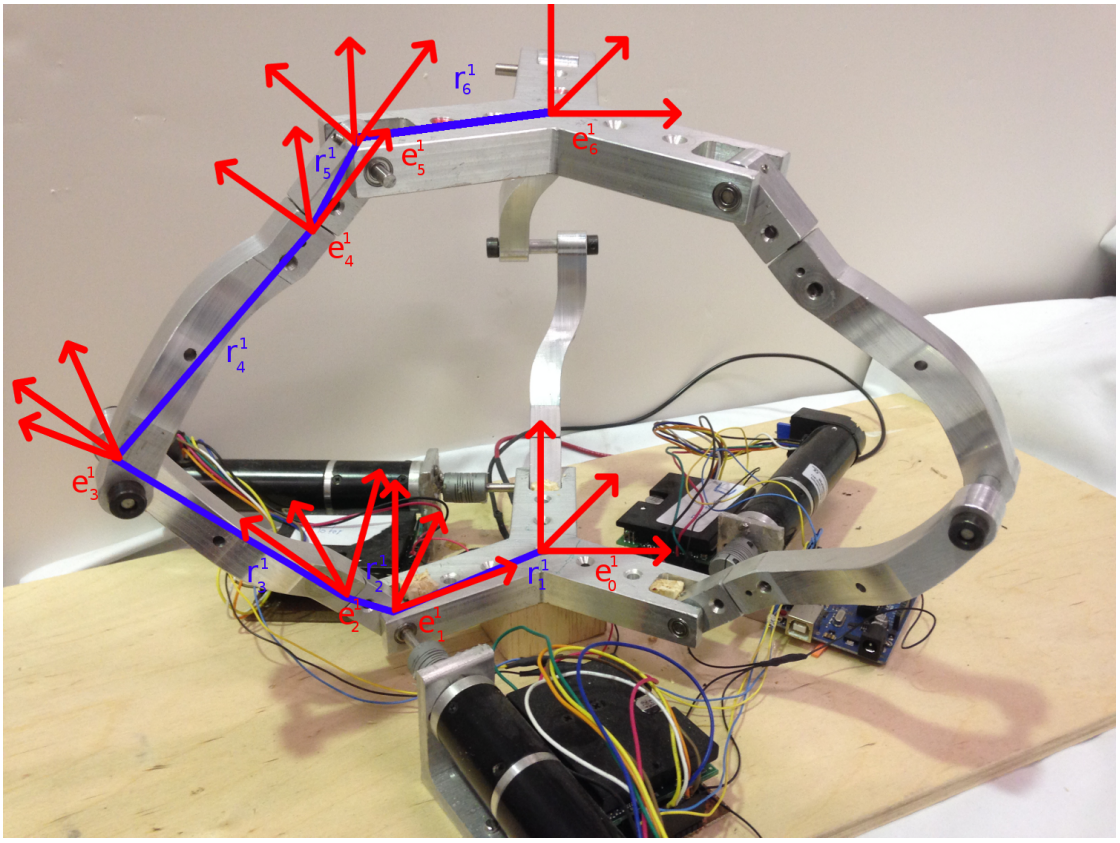


Figure 5.11: Link frames of the CWJ

rotation on the base plate, which gives the designer the ability to work with a non symmetric joint. Repeating the previous procedure for each leg, we define the entire system. We continue assuming three legs with a relative angle of $\frac{2}{3}\pi$ between each, however this is not a requirement, and any amount of legs and any relative angle can be used.

Since the joint is driven by three set of angles, the system should be defined with these three angles as the control inputs. Because of the multilink structure of each leg, constraint equations are required to enforce the geometric limitations of

the system. The constraints are then, the position of the last link of all the chains (i.e. each leg) should be the same Equation (5.8), the three last links should be coplanar Equation (5.9) and the angle between each of the last links should be $\frac{2}{3}\pi$ Equations (5.10) to (5.12)

$$\boldsymbol{r}_6^1 - \boldsymbol{r}_6^2 = \mathbf{0} \quad and \quad \boldsymbol{r}_6^1 - \boldsymbol{r}_6^3 = \mathbf{0} \tag{5.8}$$

$$\boldsymbol{e}_6^1 \cdot (\boldsymbol{e}_6^2 \times \boldsymbol{e}_6^3) = 0 \tag{5.9}$$

$$\cos(\phi_{1,2}) - \boldsymbol{e}_6^1 \cdot \boldsymbol{e}_6^2 = 0 \tag{5.10}$$

$$\cos(\phi_{1,3}) - \boldsymbol{e}_6^1 \cdot \boldsymbol{e}_6^3 = 0 \tag{5.11}$$

$$\cos(\phi_{3,2}) - \boldsymbol{e}_6^3 \cdot \boldsymbol{e}_6^2 = 0 \tag{5.12}$$

The previous set of equations is used to obtain the internal angles after the basal revolute joints have been set. The output of the pointing device can be considered the normal of the distal plate that can be obtained with the cross product of any two of the upper links Equation (5.13).

$$\boldsymbol{n} = \boldsymbol{e}_6^2 \times \boldsymbol{e}_6^3 \tag{5.13}$$

The normal can be used to compute the elevation angle α solving the dot product of said normal with the reference as shown in Equation (5.14). To compute the azimuth β we first need to project the normal to a reference plane using Equation (5.15). n_x and n_y represent the x and y components of the vector. The azimuth

is then computed with Equation (5.16)

$$\alpha = \arccos(\boldsymbol{n} \cdot [0, 0, 1]) \tag{5.14}$$

$$\boldsymbol{n}_{x,y} = \frac{[n_x, n_y]^T}{|[n_x, n_y]|} \tag{5.15}$$

$$\beta = \arccos(\boldsymbol{n}_{x,y} \cdot [1, 0]^T) \tag{5.16}$$

The set of equations presented in this section not only can be used to design non symmetric joints but also can be used to solve the dynamics of the plant. In the next section, we use this system to perform an uncertainty analysis.

UNCERTAINTY ANALYSIS

All mechanical manipulators have multiple sources of uncertainty.[152] Most importantly, researchers focus on uncertainties in design,[128;144;145;147;152] linearizations of the kinematics equation[153;154] and singular configurations.[131;132;141;142;153;155] These studies are focused on system analysis and only few controllers have been proposed.[150]

In the particular case of the CWJ, we are interested in uncertainty in the pointing precision due to uncertainty in the base angles. To perform this analysis, we use a statistical approach.[154–156] We assume that the error in the revolute joint positions are normally distributed, with mean zero and known standard deviation.

We propagate this distribution through the system using kinematic equations to obtain angular uncertainty in the normal vector to the distal plate. Since the system has not only nonlinear kinematics but also implicit constraints, the uncertainty propagation is not trivial. First we try an analytical approach, and then compare the results with the straightforward UT,[148] and finally we validate by using linear error theory the previous analyseswith a Montecarlo simulation.

The pointing direction of the distal plate $\boldsymbol{n}$ is a function of the revolute angles of the basal plate denoted by $\boldsymbol{\theta}$, and the internal angles of legs denoted by $\boldsymbol{\beta}$ Equation (5.17).

$$\boldsymbol{n} = \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\beta}) \tag{5.17}$$

The sensitivity of the normal $\boldsymbol{n}$ to perturbations in $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ is given by Equation (5.18), $\boldsymbol{\delta}n$ represents the displacement to the pointing direction due to the perturbations on $\boldsymbol{\theta}$ given by $\boldsymbol{\delta}\theta$ and the perturbations in $\boldsymbol{\beta}$ given by$\boldsymbol{\delta}\beta$. The nominal condition is given by $\boldsymbol{\theta}_0$ and $\boldsymbol{\beta}_0$. The first term of the RHS can be expanded using Taylor series Equation (5.19).The final relation can be obtained discarding the Higher Order Terms (HOT) Equation (5.20)

$$\boldsymbol{\delta}n = \boldsymbol{f}(\boldsymbol{\theta}_0 + \boldsymbol{\delta}\theta, \boldsymbol{\beta}_0 + \boldsymbol{\delta}\beta) - \boldsymbol{f}(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0) \tag{5.18}$$

$$\boldsymbol{\delta}n = \boldsymbol{f}(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0) + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \boldsymbol{\delta}\theta + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \boldsymbol{\delta}\beta + \boldsymbol{HOT}(\boldsymbol{\theta}, \boldsymbol{\beta}) - \boldsymbol{f}(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0) \tag{5.19}$$

$$\boldsymbol{\delta}n = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \boldsymbol{\delta}\theta + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \boldsymbol{\delta}\beta \tag{5.20}$$

The previous equation cannot be computed directly since $\boldsymbol{\beta}$ is not a set of independent variables; its elements are related by the constraints equations $\boldsymbol{c}$ as shown in Equation (5.21). Therefore, the sensitivity of the constraint equations with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ must be calculated as shown in Equation (5.22) to provide these relations.

$$\mathbf{0} = \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\beta}) \tag{5.21}$$

$$\boldsymbol{\delta} c = \mathbf{0} = \boldsymbol{c}(\boldsymbol{\theta}_0 + \boldsymbol{\delta}\theta, \boldsymbol{\beta}_0 + \boldsymbol{\delta}\beta) - \boldsymbol{c}(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0) \tag{5.22}$$

To obtain the sensitivity equation a Taylor series expansion, of the constraint equations is used. A special consideration must be made which is that the constraints must be satisfied at all times hence $\boldsymbol{\delta} c = \mathbf{0}$ which yields Equation (5.23). We discard HOT Equation (5.24) and then $\boldsymbol{\delta}\beta$ can be solved Equation (5.25).

$$0 = \boldsymbol{c}(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0) + \left.\frac{\partial \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \boldsymbol{\delta}\theta + \left.\frac{\partial \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \boldsymbol{\delta}\beta + \boldsymbol{HOT}(\boldsymbol{\theta}, \boldsymbol{\beta}) - \boldsymbol{c}(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0) \tag{5.23}$$

$$\mathbf{0} = \left.\frac{\partial \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \boldsymbol{\delta}\theta + \left.\frac{\partial \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \boldsymbol{\delta}\beta \tag{5.24}$$

$$\boldsymbol{\delta}\beta = \left[\left.\frac{\partial \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0}\right]^{-1} \left[\left.\frac{\partial \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0}\right] \boldsymbol{\delta}\theta \tag{5.25}$$

The newly found $\boldsymbol{\delta}\beta$ can be replaced on to Equation (5.20) to obtain the final expression that can be used to propagate the uncertainties in the basal input angles $\delta\boldsymbol{\theta}$ to the manipulator orientation $\delta\boldsymbol{n}$. The final equation given by Equation (5.26)

is a general expression that can be used to compute the uncertainty in any parallel manipulator that can be parameterized in terms of implicit constraints. This equation however is not simple to manipulate and symbolic mathematic software may be required. A simpler method is to use the Unscented Transform to perform the uncertainty Analysis.

$$\boldsymbol{\delta} n = \left(\left[\frac{\partial \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \right] + \left[\frac{\partial \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \right] \left[\frac{\partial \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \right]^{-1} \left[\frac{\partial \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}_0, \boldsymbol{\beta}_0} \right] \right) \boldsymbol{\delta} \theta \quad (5.26)$$

UNSCENTED TRANSFORM

The Unscented Transformation (UT) is a mathematical function that enables the computation of the statistics of a transformed probability density function in terms of the statistics of the independent random variable. It is derived from the insight that it is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation.[148] The basic principle is illustrated in Figure 5.12 where the interest is in computing the statistics of a function of a random variable. UT uses a set of points (called sigma points) lying on the 1 σ contour of the probability density function of the random variable in the problem of interest. Weight functions associate a weight with all the sigma points from a partition of unity (i.e. $\sum w_i = 1, \forall i = 0, ..., n$, with n being the number of sigma points). This point distribution is transformed using the nonlinear function of interest.

Let $\mathbf{x}$ be a random variable whose probability density function is known and let $\mathbf{z} = \mathbf{f}(\mathbf{x})$ be the nonlinear function of the random variable. The unscented transform enables the computation of the mean and covariance of $\mathbf{z}$. To this end, sigma points are chosen in the $\mathbf{x}$ space and are transformed using the known map. From the set of transformed points, the mean and covariance is computed using expressions given in Equation (5.27) and Equation (5.28).

$$\hat{\mathbf{z}} = \sum_{i=0}^{n} w_i \mathbf{z}_i = \sum_{i=0}^{n} w_i \mathbf{f}(\mathbf{x}_i) \tag{5.27}$$

$$\Sigma_{\mathbf{z}} = \sum_{i=0}^{n} w_i \left(\mathbf{z}_i - \hat{\mathbf{z}}\right)\left(\mathbf{z}_i - \hat{\mathbf{z}}\right)^T \tag{5.28}$$

where $\hat{\mathbf{z}}$ denotes the mean value of the distribution and $\Sigma_{\mathbf{z}}$ denotes the covariance matrix. Recent research shows that by modifying the values of individual weights and the number of weight functions, high order moment matching can also be achieved.[157]

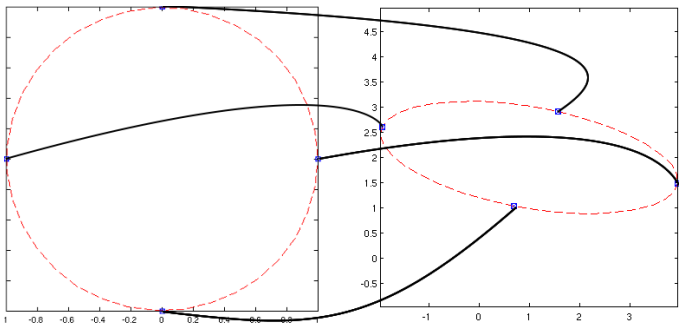


Figure 5.12: Graphical Illustration of the Unscented Transformation (plot on the left denotes sigma points in $\mathbf{x}$ space and the plot on the right denotes the transformed sigma points)

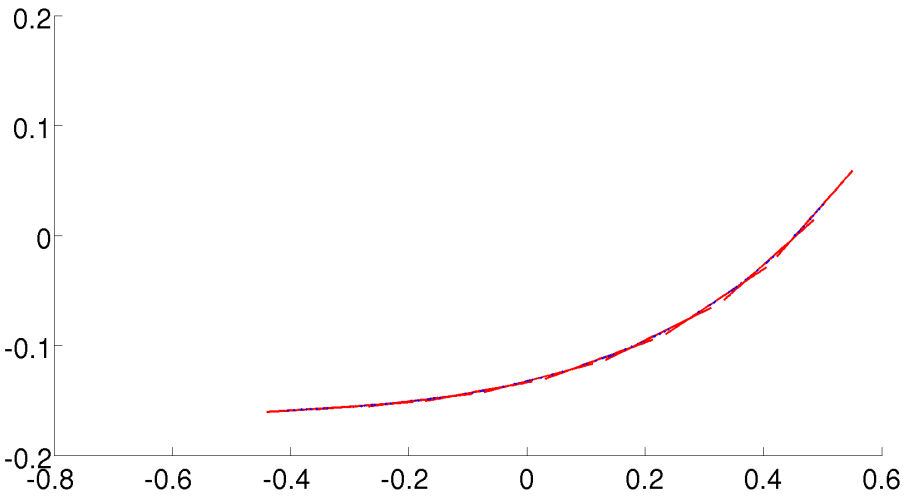


Figure 5.13: u_1 with uncertainty and u_1 updated to generate new positions

We use the unscented transform alongside with a Monte-carlo simulation to validate the effects of the input errors on the pointing direction of the distal plate. For this experiment, we assume that the joints have an input noise given by $\mathcal{N}(0, \sigma_\pi)$ with $\sigma_\pi = 3[Deg]$ and for the Montecarlo Simulation 1000 samples are used. The analysis is repeated multiple times for multiple configurations, and the results are plotted on a 3D sphere representing the workspace, and on a 2D plane to evaluate how the sigma contours estimated with the use of the UT catch the sample points. First we assume noise in only one of the inputs and we generate multiple positions by updating this input. The results are presented in Figures 5.13 and 5.14.

Then we keep noise in only one inputs and generate new positions by updating an input that has no noise, the result is shown in Figures 5.15 and 5.16. In the second scenario, we consider two of the inputs to be affected by noise and again we generate multiple trajectories by updating one of these inputs, the result is shown

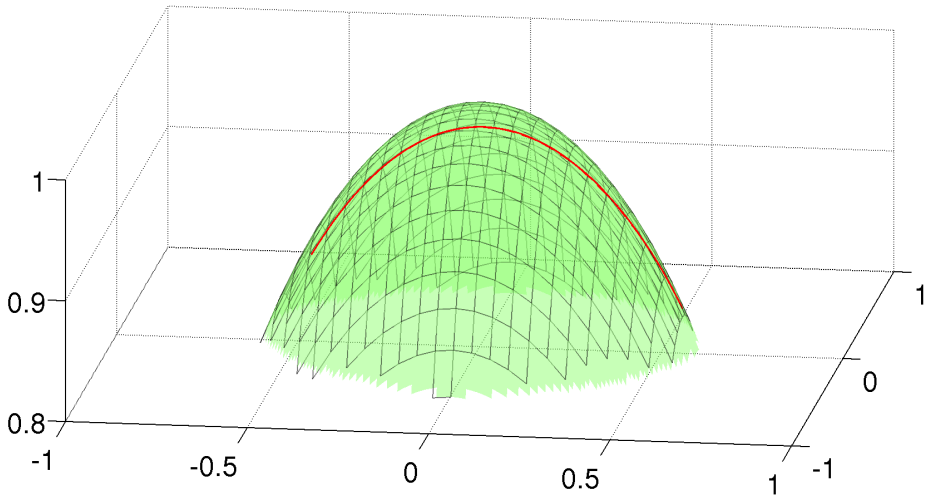


Figure 5.14: u_1 with uncertainty and u_1 updated to generate new positions

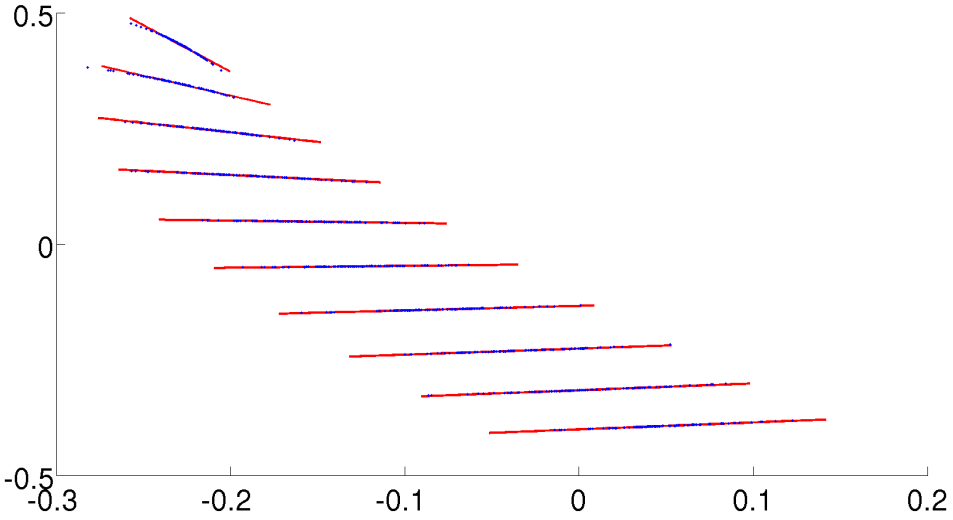


Figure 5.15: u_1 with uncertainty and u_2 updated to generate new positions

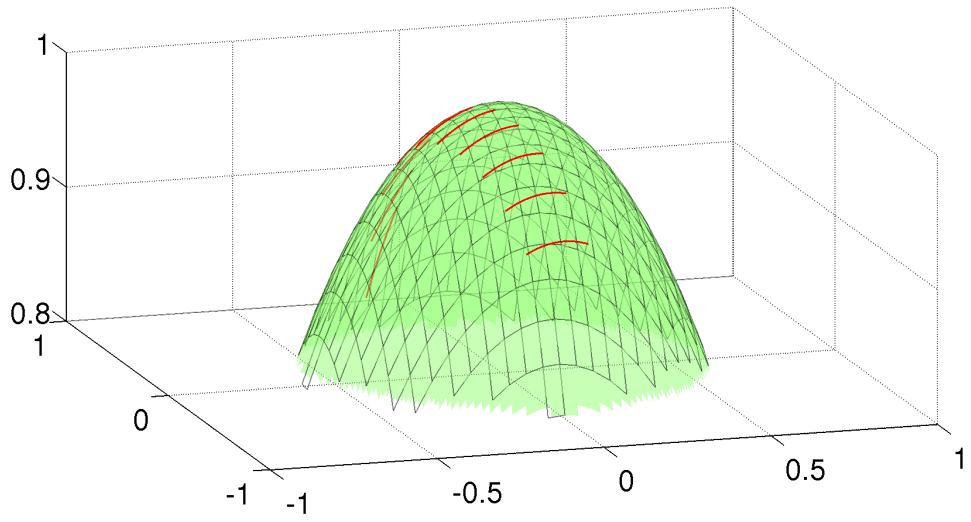


Figure 5.16: u_1 with uncertainty and u_2 updated to generate new positions

in Figures 5.17 and 5.18.

We consider two of the inputs to be affected by noise and we generate trajectories by updating the input not affected by noise. The sigma contours for this case are shown in Figures 5.19 and 5.20.

Finally, all three inputs are consider to be affected by noise and multiple configurations are tested updating only one of the joints Figures 5.21 and 5.22.

The most realistic scenario is when the joint have uncertainty in all of the three inputs then a uncertainty map is generated then for this configuration. To generate this map a Monte-Carlo Simulation and a UT are perform on multiple configurations from the entire workspace and the results are superimposed over a sphere Figure 5.24 and are projected over a plane Figure 5.24.

As seen from the previous plots when the uncertainty on the inputs is considered

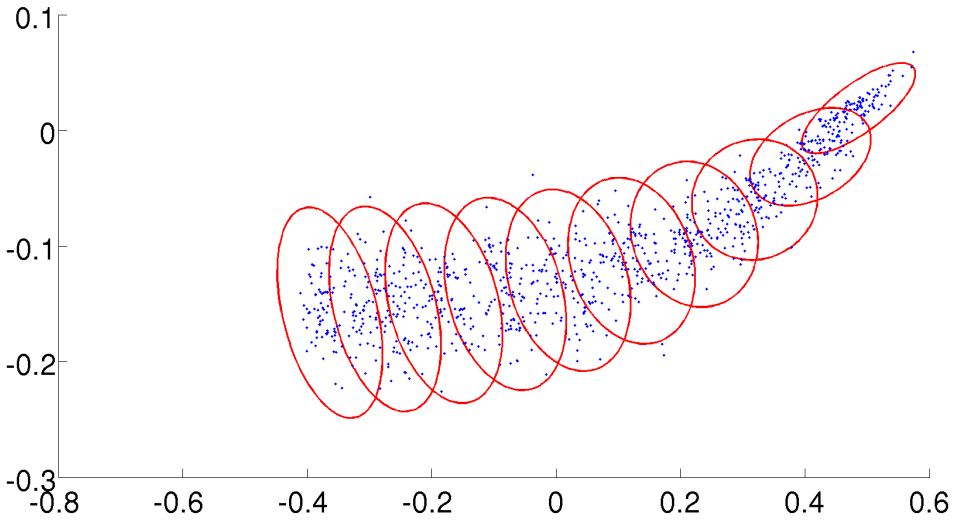


Figure 5.17: u_1 and u_2 with uncertainty and u_1 updated to generate new positions

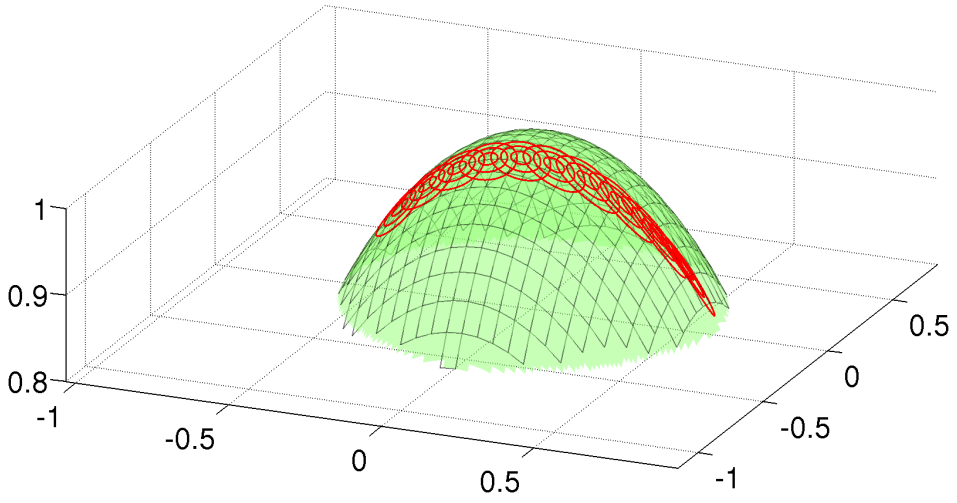


Figure 5.18: u_1 and u_2 with uncertainty and u_1 updated to generate new positions

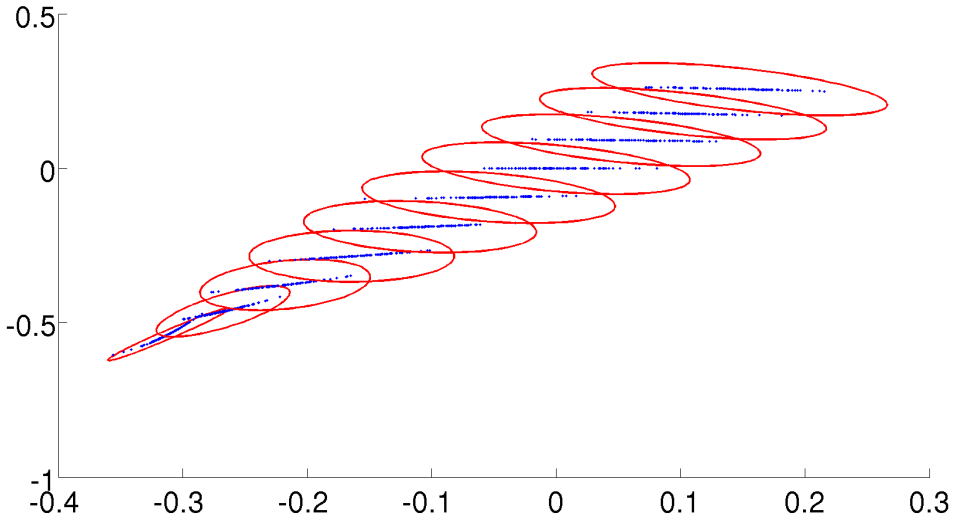


Figure 5.19: u_1 and u_2 with uncertainty and u_2 updated to generate new positions

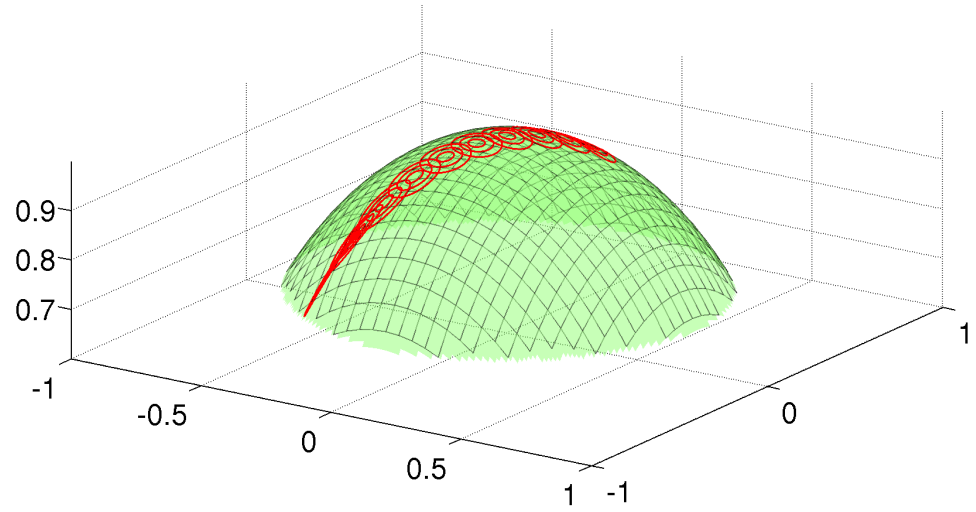


Figure 5.20: u_1 and u_2 with uncertainty and u_2 updated to generate new positions

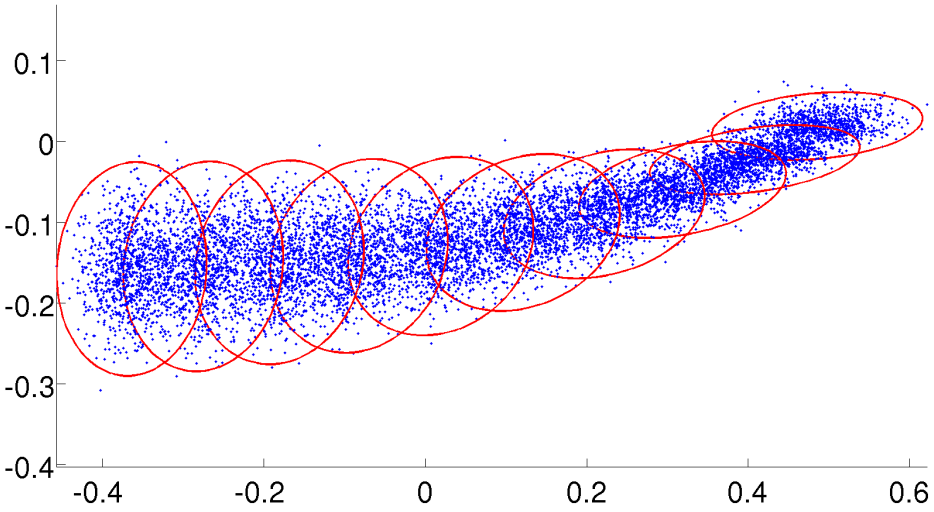


Figure 5.21: u_1, u_2 and u_3 with uncertainty and u_1 updated to generate new positions

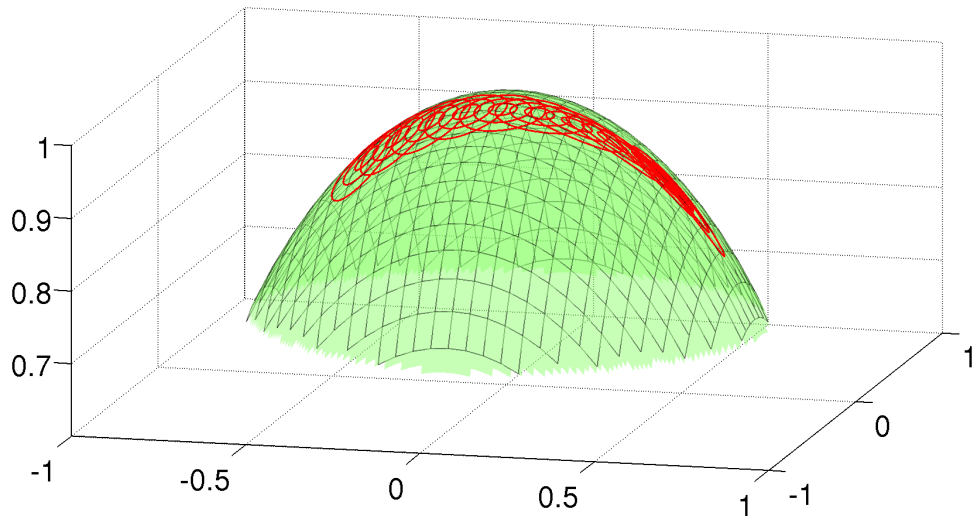


Figure 5.22: u_1, u_2 and u_3 with uncertainty and u_1 updated to generate new positions

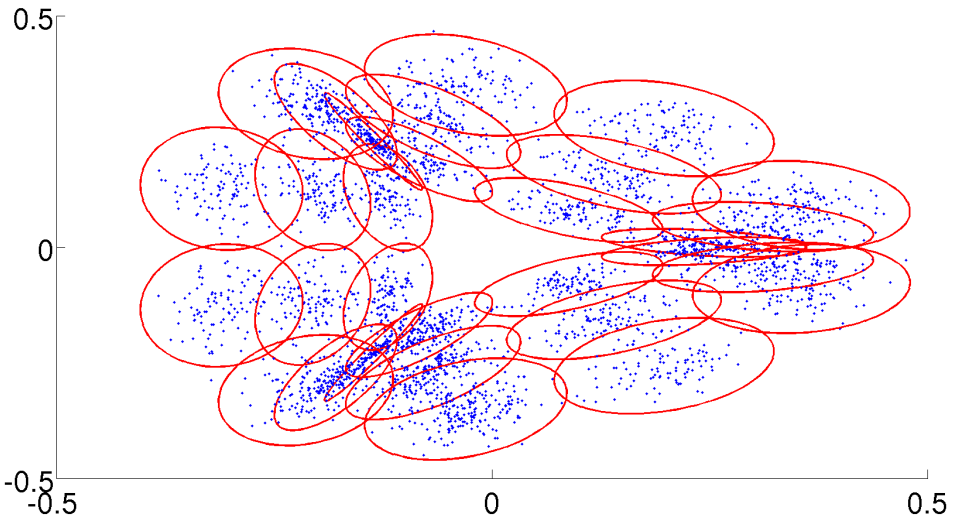


Figure 5.23: u_1, u_2 and u_3 with uncertainty and u_1, u_2 and u_3 updated to generate new positions

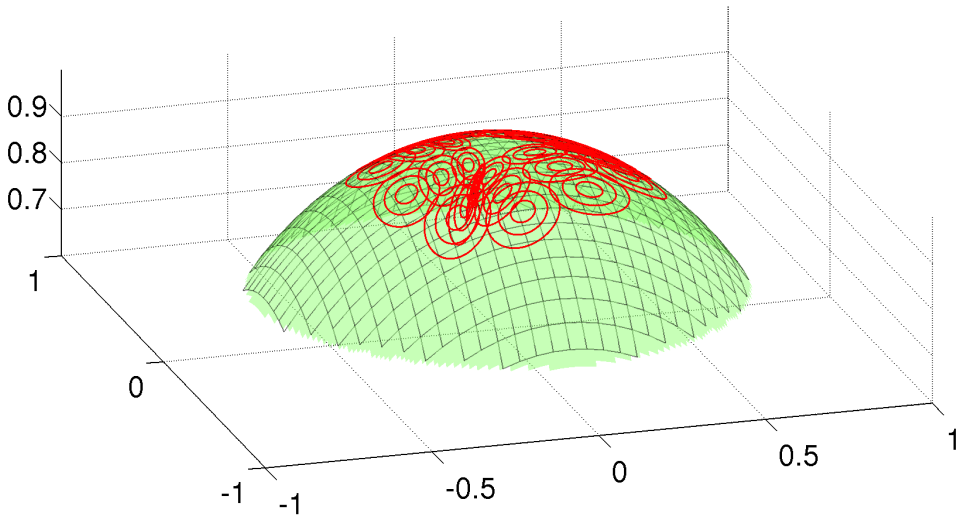


Figure 5.24: u_1, u_2 and u_3 with uncertainty and u_1, u_2 and u_3 updated to generate new positions

to be affecting only one of the angles that drive the joint the output uncertainty is highly directional predominantly on the direction of the uncertain revolute joint. It can be observed that the uncertainty grows larger when the vertical pointing direction is departed being at its largest when the pointing direction is horizontal. When multiple joints are affected by uncertainty, the directionality of the output uncertainty depends on the joint configuration. When the orientation aligns with one of the driving inputs the directionality of the uncertainty gets aligned as well. In all these cases the UT can catch not only the magnitude but also the directionality proving to be a great tool to be used in uncertainty analysis for this type of manipulators.

AZIMUTH ELEVATION COMPARISON

The CWJ can be used as a redundant pointing device, where three inputs are used to control only two degrees of freedom. In this section we perform a side by side comparison with a traditional azimuth/elevation pointing device. The main focus of the study is to identify the sensitivity on the pointing error for both devices. To this end, a pointing error metric is required. The precision in pointing can be measured by the error angle γ between the reference direction $\boldsymbol{r}$ and the actual direction $\boldsymbol{n}$ as shown in Figure 5.25. Since the cosine and the arccosine functions are nonlinear, a better metric is given by $\cos(\gamma)$; however, the result is counterintuitive since the no error condition is indicated by 1 rather than by 0, hence this quantity

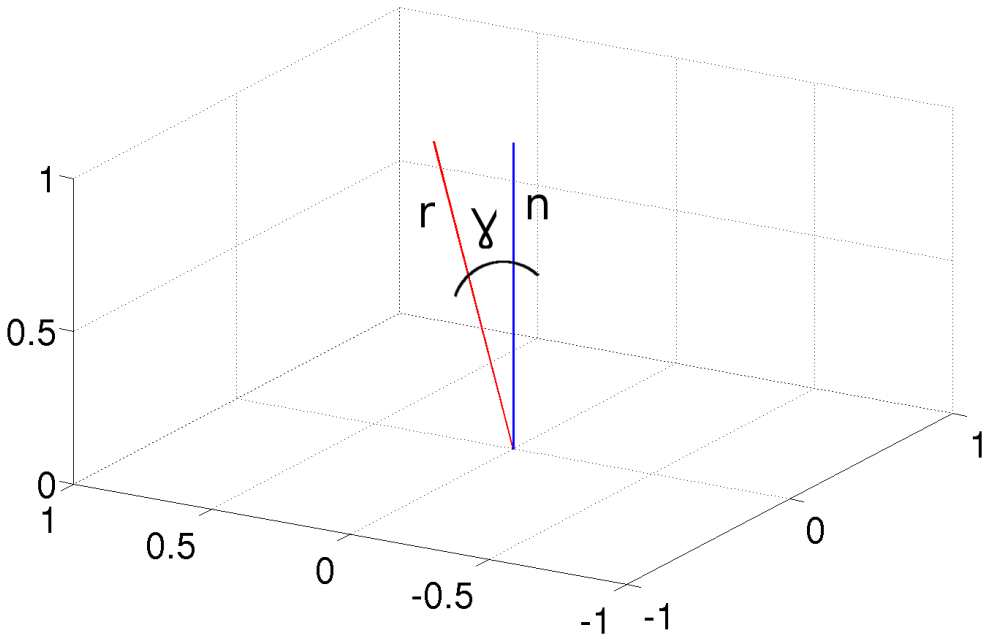


Figure 5.25: Error metric in pointing devices

is subtracted to give the final metric as

$$\epsilon = 1 - \cos(\gamma) \tag{5.29}$$

For an azimuth elevation joint the pointing direction depends only on two angles, elevation α and azimuth β, and is given by Equation (5.30) where $\boldsymbol{\theta} = [\alpha \;\; \beta]^T$

$$\boldsymbol{n} = \boldsymbol{h}(\boldsymbol{\theta}) = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta) & \sin(\alpha) \end{bmatrix}^T \tag{5.30}$$

To find the sensitivity of the pointing error due to the uncertainty of the driving

angles, the error metric Equation (5.31) can be expanded using Taylor series

$$\epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) = 1 - \cos(\gamma) = 1 - \boldsymbol{h}(\boldsymbol{\theta})^T \cdot \boldsymbol{h}(\boldsymbol{\theta}) \tag{5.31}$$

$$\epsilon = \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta})|_{\boldsymbol{\theta}_0} + \frac{\partial}{\partial \boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) \bigg|_{\boldsymbol{\theta}_0} \delta\boldsymbol{\theta} + HOT \tag{5.32}$$

$$\epsilon = \frac{\partial}{\partial \boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) \bigg|_{\boldsymbol{\theta}_0} \delta\boldsymbol{\theta} \tag{5.33}$$

where $\boldsymbol{\theta}_0$ is the nominal configuration and $\boldsymbol{\theta}$ is a random variable with given statistical properties $\mathcal{N}(\boldsymbol{\theta}_0, \sigma_\theta)$. The expected value of the error metric can be computed using $E(\epsilon^2)$, replacing the error ϵ with the identity found using the Taylor series expansion

$$E(\epsilon^2) = E\left(\left[\frac{\partial}{\partial \boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) \bigg|_{\boldsymbol{\theta}_0} \right] \delta\boldsymbol{\theta} \left[\frac{\partial}{\partial \boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) \bigg|_{\boldsymbol{\theta}_0} \right] \delta\boldsymbol{\theta} \right) \tag{5.34}$$

$$E(\epsilon^2) = \left[\frac{\partial}{\partial \boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) \bigg|_{\boldsymbol{\theta}_0} \right] E\left(\delta\boldsymbol{\theta} \delta\boldsymbol{\theta}^T \right) \left[\frac{\partial}{\partial \boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) \bigg|_{\boldsymbol{\theta}_0} \right]^T \tag{5.35}$$

$$E(\epsilon^2) = \left[\frac{\partial}{\partial \boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) \bigg|_{\boldsymbol{\theta}_0} \right] [\sigma_\theta] \left[\frac{\partial}{\partial \boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) \bigg|_{\boldsymbol{\theta}_0} \right]^T \tag{5.36}$$

The previous equation can be developed a little bit further without loss of generalization. The partial derivative of $\boldsymbol{h}(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is defined as shown in Equation (5.37). Using this identity, the final expression for the expected value is given by Equation (5.39) where n is the number of control inputs.

$$H = \frac{\partial}{\partial \boldsymbol{\theta}} h(\boldsymbol{\theta}) \tag{5.37}$$

$$\frac{\partial}{\partial \boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}_0, \boldsymbol{\theta}) = -\boldsymbol{h}(\boldsymbol{\theta}_0)^T \cdot \boldsymbol{H}(\boldsymbol{\theta}) \tag{5.38}$$

$$E(\epsilon^2) = \left[\boldsymbol{h}(\boldsymbol{\theta}_0)^T \cdot \boldsymbol{H}(\boldsymbol{\theta}_0)\right]_{1\times n} [\sigma_\theta]_{n\times n} \left[\boldsymbol{H}(\boldsymbol{\theta}_0)^T \cdot \boldsymbol{h}(\boldsymbol{\theta}_0)\right]_{n\times 1} \tag{5.39}$$

Equation (5.39) is a general expression that can be incorporated with any manipulator to evaluate analytically the pointing error due to the error in the control inputs. For the azimuth elevation joint the solution can be easily obtained by evaluation of Equation (5.30) and its derivatives, and plug them in Equation (5.33), which yields Equation (5.40). On the other hand for the CWJ the solution is not straightforward since implicit equations must be solved, hence the use of a symbolic toolbox is required.

$$E(\epsilon^2) = 4\sigma_\theta^2 \sin^4(\frac{\delta_\alpha}{2}) \sin^2(2\alpha + \delta\beta)) \tag{5.40}$$

For the simple AEJ an analytical expression was found, however for the CWJ or more complex joints an analytical expression for the required transformation may not exist. For redundant mechanical devices, solving the kinematic input-output relation requires the solution of complex nonlinear implicit functions. To this extent using a Monte-Carlo Simulation to evaluate the pointing error is computationally expensive, even more if its taken into account that the Monte-Carlo simulation needs to be re-run for each configuration. To get an accurate quantification of the error distribution over workspace hundred of thousand of samples on each of

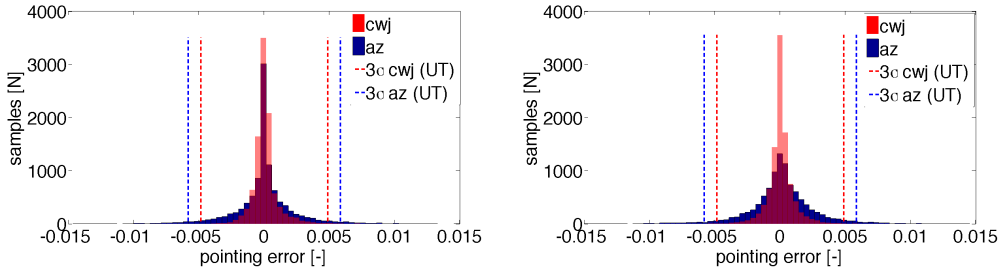
Configuration		CWJ			Azimuth-Elevation		
α Deg	β Deg	UT	MC	Analytical	UT	MC	Analytical
60	85	0.0009	0.0010	0.0012	0.0019	0.0024	0.0029
120	60	0.0006	0.0010	0.0012	0.0020	0.0027	0.0032
-120	60	0.0006	0.0010	0.0011	0.0020	0.0026	0.0030
90	90	0.0006	0.0008	0.0010	0.0019	0.0024	0.0029

Table 5.1: Uncertainty comparison for the CWJ and a azimuth-elevation joint

probably hundreds of configurations have to be evaluated. On the other hand, the UT drastically reduces the amount of required function evaluations since only two evaluations of the kinematic equations per input are required, hence for a 3 inputs joint the functions evaluation required are 6 and not thousands.

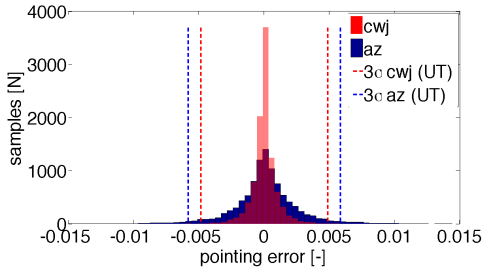
To validate the solution obtained with the linearization of the equations and the UT approach, a comparison between these two and a Monte-Carlo simulation for multiple points in the workspace is presented. In Table 5.1, a comparison of the results is shown and in Figures 5.26(a) to 5.26(d) the histograms showing the comparison of the error estimations and the Montecarlo simulation results are presented. Monte-Carlo

It can be seen from the results that the uncertainty in the CWJ is smaller for all the configurations. Being a closed kinematic chain the error in pointing get reduced by the physical constraints. The error in the inputs does not get systematically added since the constraints have to be satisfied for all configurations. It can be seen also that the solution obtained with the linearization method overestimates the error. Since the error is a cosine function and the linearization can only capture the linear component it is expected to have larger errors, which will increase when the linearization boundary also increase. This error overestimation is another reason

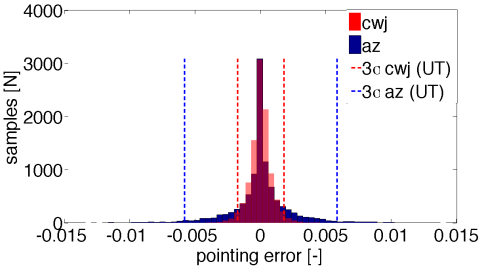


(a) $\alpha = 60\ Deg\ \beta = 85\ Deg$

(b) $\alpha = 120\ Deg\ \beta = 60\ Deg$



(c) $\alpha = -120\ Deg\ \beta = 60\ Deg$



(d) $\alpha = 90\ Deg\ \beta = 90\ Deg$

Figure 5.26: Error distribution comparison for a CWJ and a AEJ

to prefer the UT over the analytical solution. From the results of the experiment it can be seen that the UT can perform faster than the MC simulation, but also better than the linearization, which makes it a perfect tool to perform uncertainty analysis. As a future work can be consider to expand the analysis to other types of uncertainties such as the ones due to mechanisation, assemble and design.

5.4 CARPAL WRIST JOINT AS ATTITUDE CONTROLLER EXPERIMENTAL LAYOUT

In this section, the experimental apparatus designed to test the CWJ controller is presented and its functionality is described. The devices is shown in Figure 5.27. A

joint that allows two degree of freedom, machined from an aluminum block, is used to connect two square beams to obtain an articulated L-shaped pendulum. The measurements of the two angles which are required to perform control actions are acquired with two encoders mounted on the mechanized joint sides. The CWJ is mounted on a plate that is attached to one of the ends of the pendulum, as shown in. All the wires required for the connections are routed from the plate through a hole in the center of the joint and all others are gathered and routed at the side of the pendulum. Any loose cable is left on the joint to allow the relative motion of the aluminum arms.

An aluminum box is constructed to contain the experiment and to adjust the mounting requirements of the space ship. To prevent any injuries to personnel or damage to surrounding equipment in case of a failure, plexiglass sheets are used to conceal the experiment. Since the pendulum can swing freely in any direction a hoop is added to restrict the maximum allowable displacement.

The experiment has to be fully autonomous. Two linear actuators are added to generate perturbations and to give initial conditions to the pendulum. The actuators are placed with a relative angle of 90 degrees to allow each servo to excite one of the degrees of freedom. When both servos are fully extended, the pendulum locks against the hoop to prevent the pendulum from moving around during takeoff and landing.

The CWJ used for the microgravity experiment is show in Figure 5.28. Three RC servos are used to control the basal angles, and a brushless motor and a plexiglass disk are used to build the momentum wheel that is mounted on the distal

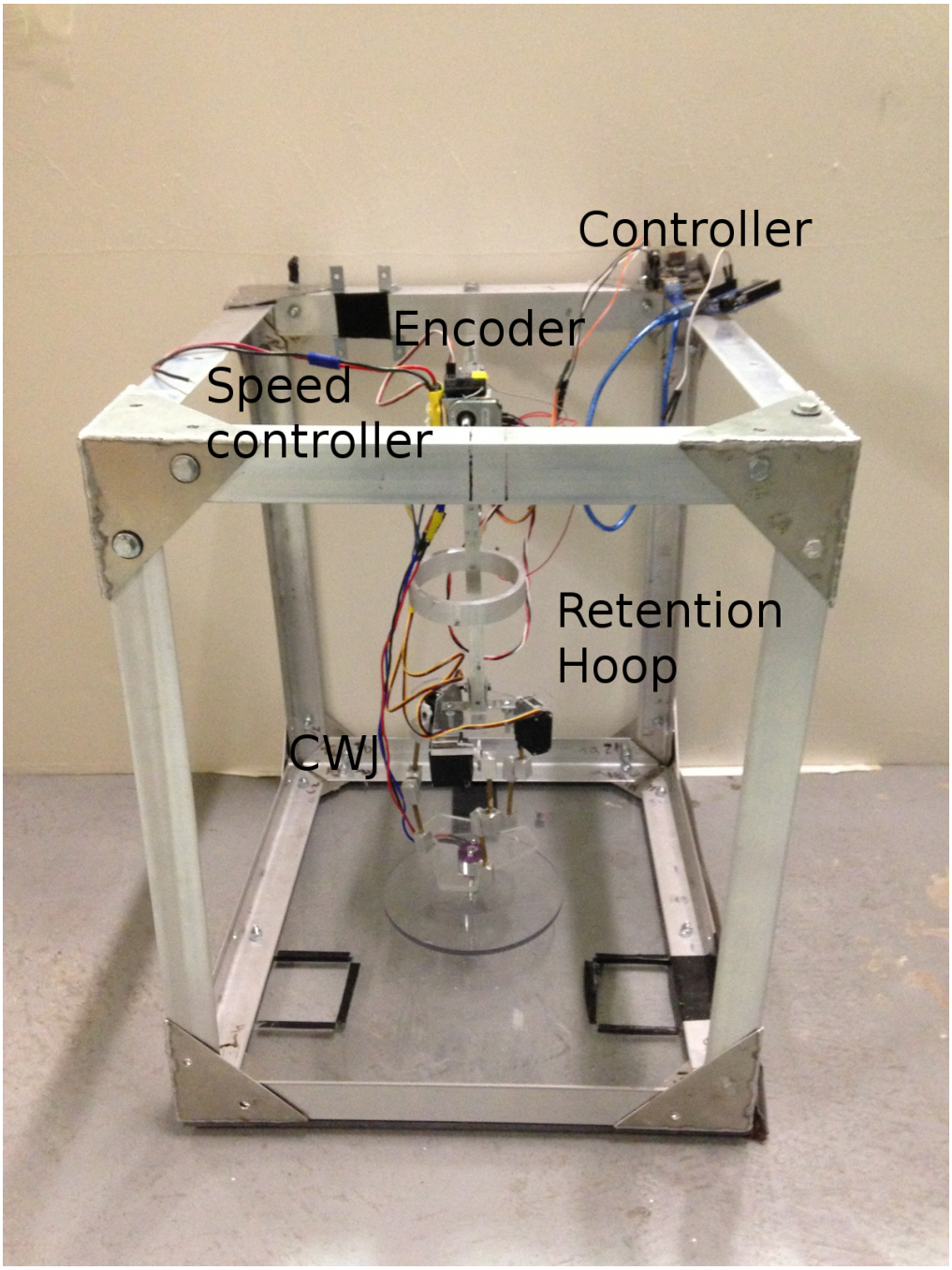


Figure 5.27: Apparatus design for microgravity experiment

plate. To control the servos and the motor speed controller a Beagle Bone Black single board computer running the SAI is used. The entire setup is shown in Figure 5.27.

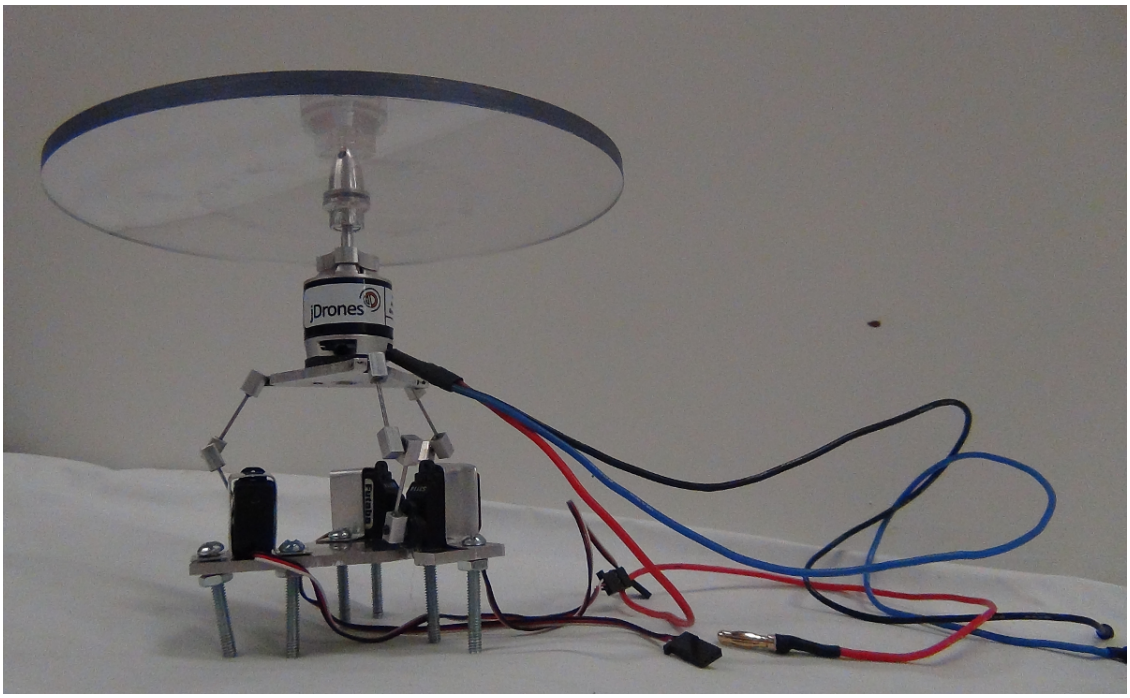


Figure 5.28: CWJ used for the microgravity experiment

The system runs three control loops. One control loop stabilizes the pendulum where the inputs are the encoder measurements and the outputs are the attitude α and β angles required by the pointing mechanism. A second control loop takes as inputs the reference angles and outputs the corresponding servo angles. With the use of the kinematic or reverse-kinematic equations it is possible to obtain the required servos angle to achieve the upper plate attitude. To assure the proper angles are achieved with the PWM signal, the control loop relies on the position

controller embedded on the servo. With a simple calibration an open loop signal from the main computer controls the servo within the required precision. A third control loop at the hardware level with the PWM driver generates a square analog signal in the required range to achieve the pulse requested.

Since the experiment is intended to be used in a microgravity environment, to test controllers on the laboratory a counter weight system is added Figure 5.29. This system consists of an extension which has at one end a mechanism to attach to the pendulum and at the other end a canister for storing mass. These devices give the ability to trim and balance the pendulum, allowing equivalent behavior to a 0 g environment. When the canister is removed prior to flight the inertia of the system must be recomputed.

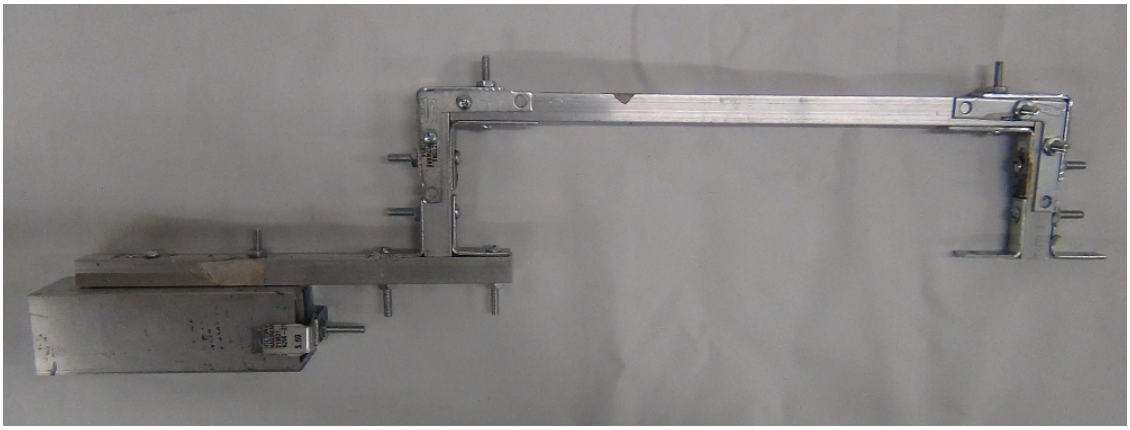


Figure 5.29: Counter weight system used to counter act gravity

The systems needs to be able to identify the microgravity period to activate itself since the system should be fully autonomous. Using an IMU the gravity is sensed and compared to the expected flight profile to identify the microgravity

period. When the system is on the expected fly path the system will trigger the experiment following a predefined sequence. The first step is to spin up the wheel. Then, the linear servos retract to unlock the pendulum. Once the pendulum is free the controller forces it back to the vertical position. Once the vertical position is achieved the system generates a new desired position and the controller performs set point regulation on that position. When the microgravity period is over and the gravity starts to increase again, the linear actuators extend to lock the pendulum once again. Finally, the current to the motor is cut and viscous damping eventually stops the wheel, preparing the system for the reentry and landing phase.

EQUATION OF MOTION ONE DEGREE OF FREEDOM

In this section, we present the first analysis case, where one angle of the pendulum is locked, reducing the system to a single degree of freedom. The equations of motion for the pendulum are derived using Lagrangian Mechanics. For a a single degree of freedom, the system is modeled as shown in Figure 5.30, where the variable to control is the angular position of the pendulum θ and the control variable is the torque angle α. Both angles are measured from the vertical position. The reference frame used to obtain the velocity and position of the pendulum is formed by two axes, $\boldsymbol{e}_\theta$ and $\boldsymbol{e}_r$. The position and velocity are given in Equation (5.41) and can be used to compute the kinetic energyEquation (5.42), potential energy Equation (5.43), and the Lagrangian Equation (5.44). To obtain the equation of motion we take the derivatives of the Lagrangian with respect to the pendulum

angle as shown in Equation (5.45) and solve for $\ddot{\theta}$. The equation of motion is finally given by Equation (5.46).

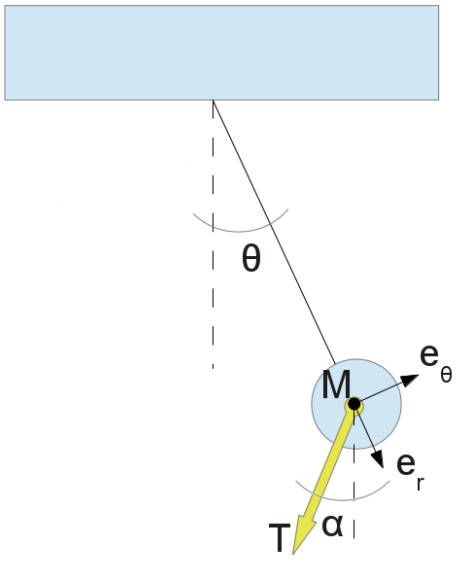


Figure 5.30: Experimental layout 1 DOF model

$$\begin{aligned} \boldsymbol{R} &= r\boldsymbol{e}_r \\ \dot{\boldsymbol{R}} &= r\boldsymbol{\omega} \times \boldsymbol{e}_r = r\dot{\theta}\boldsymbol{e}_\theta \end{aligned} \tag{5.41}$$

$$T = \frac{1}{2} m r^2 \dot{\theta}^2 \tag{5.42}$$

$$V = -mgr\cos(\theta) \tag{5.43}$$

$$L = T - V = \frac{1}{2} m r^2 \dot{\theta}^2 + mgr\cos(\theta) \tag{5.44}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = T_\alpha \tag{5.45}$$

$$\ddot{\theta} = \frac{1}{mr^2} T \sin(\alpha) - \frac{g}{r}\sin(\theta) \tag{5.46}$$

The rather simple equation of motion given byEquation (5.46) is the typical EOM for a pendulum, with an extra term due to the control torque. Since the experiment takes place in a microgravity environment the gravity term can be neglected. Thus, the nonlinearity of the equation is due to the sine function in the control action, however the torque can be parametrized in terms of α, which yields a linear system.

The open loop transfer function for the 1 DOF pendulum in a microgravity environment is given by Equation (5.47) which is equivalent to the satellite attitude EOM on each degree of freedom. It can be seen from the transfer function the lack of damping on the system, which represents the key challenge in the design of a stabilization controller.

$$G(s) = \frac{1}{s^2} \tag{5.47}$$

A general premise for spacecraft dynamics and control is that all movements should be carried out at slow speed. However due to the time constraints imposed

by the flight profile we need to develop fast controllers. High gain, low damping, fast controllers are highly sensitive to noise and error; moreover, for this particular case we have a time delay imposed by the actuator that has a negative impact on the plant dynamics. The mechanical joint has been constructed using off-the-shelf RC servos with a time constant of approximately $40[\frac{Deg}{s}]$ without any load, hence we use a slower time constant to take this and the pointer mechanism dynamics into account.

$$G(s) = \frac{\tau}{s^3 + \tau s} \tag{5.48}$$

The procedure introduced in the previous section is used to design a controller for this model. The main control input is the torque parameterized in terms of the control angle α. The second input is δ which models the hardware constraints. The system to control is given by the following transfer function matrix.

$$y = \begin{bmatrix} \frac{G(s)G_c(s)}{1+G(s)G_c(s)} & \frac{G(s)}{1+G(s)G_c(s)} \end{bmatrix} \begin{Bmatrix} r \\ \delta \end{Bmatrix} \tag{5.49}$$

Our first concern is to guarantee the stability due to the hardware constraints in this implementation. The limitations are due to the low quality of the PWM signal. Although the digital communication is done using PWM which allows a discretization in terms of a microsecond or smaller, the duty cycle for the servos is 2 ms (theoretically a base frequency of 500 Hz). However, this number is far too optimistic; our experience with this type of servo suggests that they fail to respond when the update frequency is faster than 50 Hz. The embedded controller

on the servo should be able to track the position required. However, once again, experience shows that due to the gearbox, load, and quality of inexpensive materials used, this servo has a rather large quantization error. For a more realistic model, we take the most conservative situation and assume a 10% increment interval on position, this value is used to scale the step input on the δ loop. The requirement for this loop is to reject the servo frequencies $0 - 50[Hz]$ and the error due to quantization. To achieve this design goal, we can work with the Bode plot,[158] in Figure 5.31. We show the magnitude part of the bode plot for the δ loop, where we can see that it behaves much like a low pass filter, which does not meet our design requirements. To reduce the gain margin on low frequencies we add poles through feedback control. Although one pole seems to reduce the gain, the reduction is not sufficient hence, two poles are added. Figure 5.32 clearly shows that the magnitude peak at $1[Hz]$ has a negative amplitude and rejects any signal with low or high frequency.

Now we focus on the control for the plant response to the reference input. From the root locus, clearly it can be seen that we need to add some zeros to the RHP to give stability to the system. Because we introduced two poles to the controller on the previous step, we can add two zeros and still have a proper controller, which is a critical condition for a real time hardware implementation. We repeat the root locus and frequency response analysis after adding the zeros to verify that the requirements are still met; the controller final shape is given by Equation (5.50). We choose the gain that achieves the highest damping ratio to avoid unnecessary oscillations, however it must be noted that because the system now has zeros, the

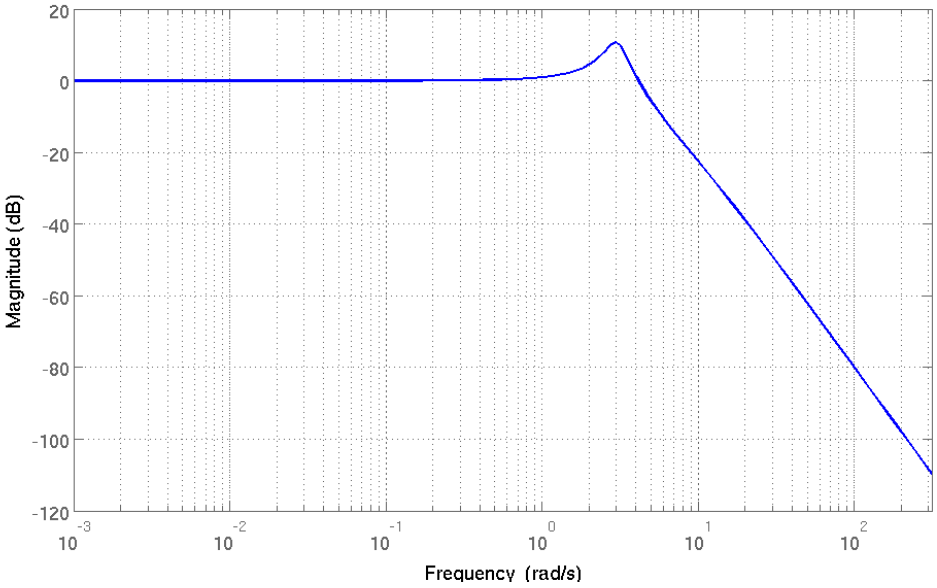


Figure 5.31: Plant with unitary closed loop bode plot

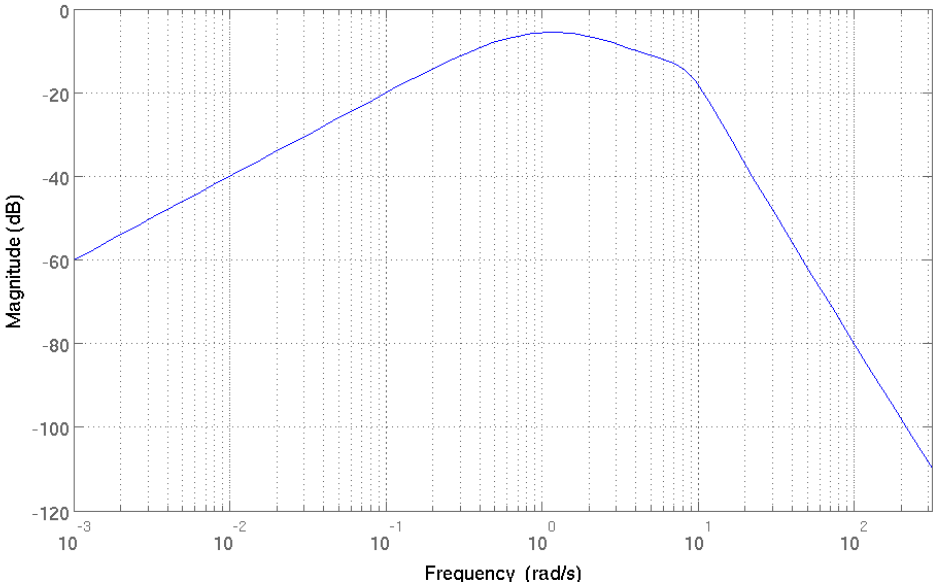
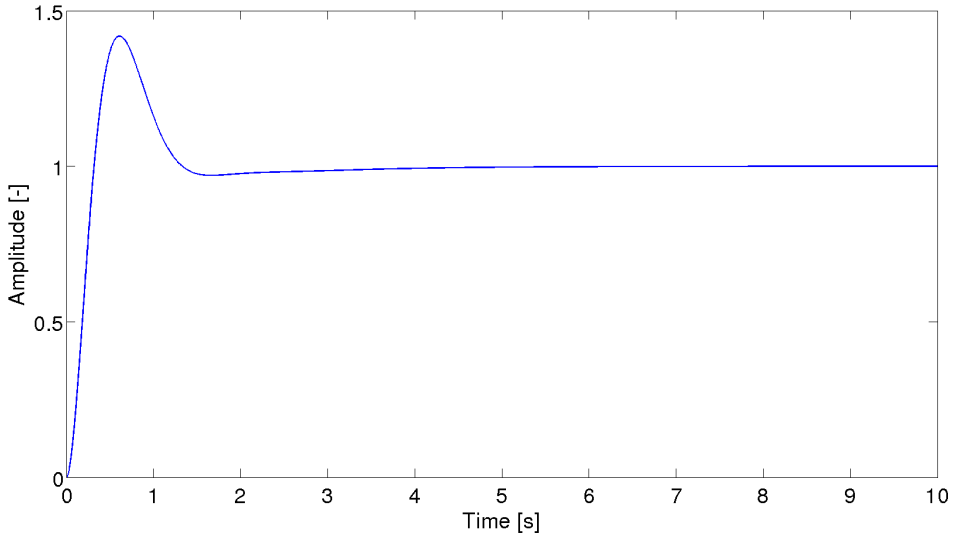


Figure 5.32: Plant with unitary closed loop bode plot



system will always have oscillations.

$$G(s) = \frac{s^2 + 2s + 1}{s^2 + 1000s} \tag{5.50}$$

We can see from the step response that the plant achieves a fast settling time with a small overshoot. The derived controller performs favorably despite the quantization and lag of the actuator.

EQUATION OF MOTION TWO DEGREE OF FREEDOM

To take a more realistic approach toward modeling the pendulum, instead of considering a point mass, a rigid body is used to model the pendulum. The approach presented in the previous section can be reused here taking into account that the length of the rod can now be replaced with the distance from the rotational center

to the center of mass. The total mass and inertia of the system are found using an experimental approach. Once the system is built, it is easy to estimate the weight with a scale and determine the center of mass with a CG machine (commonly used to find cg in airplanes). Once the system is fully assembled, applying a small perturbation to the pendulum and measuring the period, the total inertia of the system with respect to to the rotational center can be easily found using Equation (5.51).

$$T = 2\pi\sqrt{\frac{I}{mgL}} \rightarrow I = (\frac{T}{2\pi})^2 mgL \tag{5.51}$$

The model considering two degrees of freedom and the rigid body dynamics is shown inFigure 5.33 where two angles are used to describe the pendulum movements. These two angles are obtained when the pendulum displacement is projected onto two orthogonal planes. Taking into account the system symmetry, the inertia matrix becomes diagonal Equation (5.52) for the 2 DoF considered in the projection. This yields two decoupled degrees of freedom that can be model with the same EOM.

$$I = \begin{bmatrix} I_{zx} & 0 \\ 0 & I_{zy} \end{bmatrix} \tag{5.52}$$

$$I_{zx} = I_{zy} = I_{eq} \tag{5.53}$$

We repeat the analysis of the previous section, replacing the point mass with a rigid body with known physical properties. For each degree of freedom, the system

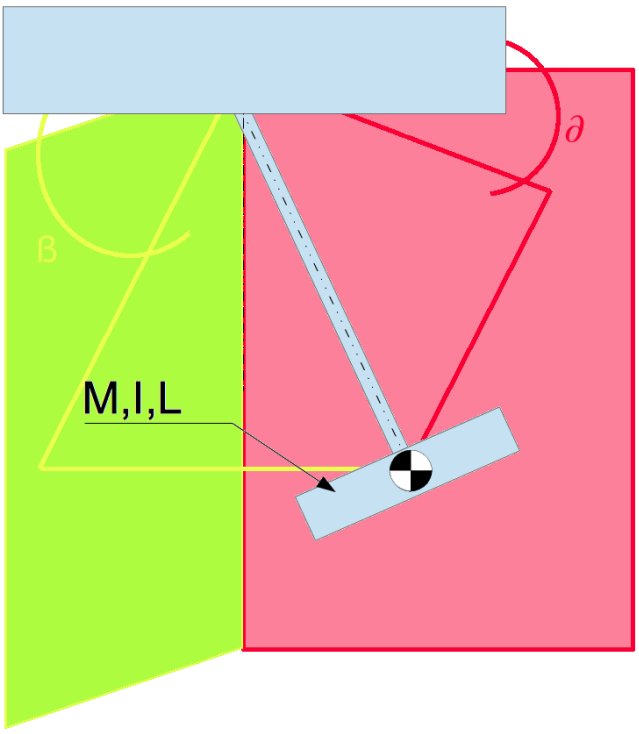


Figure 5.33: Experimental layout with inertia model

is modeled as shown in Figure 5.34, where the variable to control is the angular position of the pendulum θ and the control variable is the torque angle α. Both angles are measured from the vertical position

The reference frame used to obtain the velocity and position of the pendu-

lum is formed by two axes, $\boldsymbol{e}_\theta$ and $\boldsymbol{e}_r$. The position and velocity are given in Equation (5.54) and can be used to compute the kinetic energyEquation (5.55), potential energy Equation (5.56), and the Lagrangian Equation (5.57). To obtain the equation of motion we take the derivatives of the Lagrangian with respect to the pendulum angle as shown in Equation (5.58) and solve for $\ddot{\theta}$. The equation of motion is finally given by Equation (5.59).

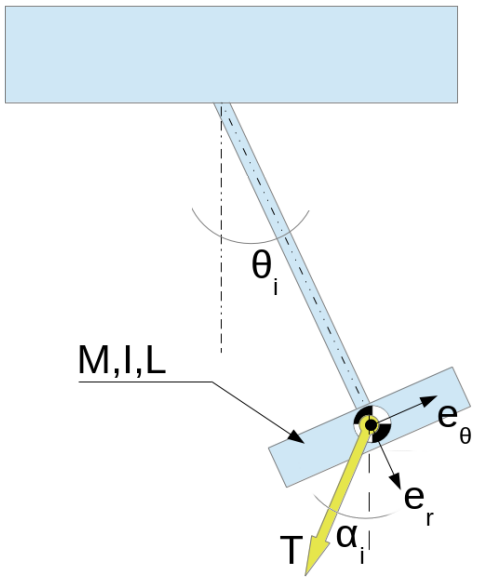


Figure 5.34: Experimental layout with inertia model

$$\begin{aligned} \boldsymbol{R} &= r\boldsymbol{e}_r \\ \dot{\boldsymbol{R}} &= r\boldsymbol{\omega} \times \boldsymbol{e}_r = r\dot{\theta}\boldsymbol{e}_\theta \end{aligned} \tag{5.54}$$

$$T = \frac{1}{2} I_{eq} \dot{\theta}^2 \tag{5.55}$$

$$V = -mgr\cos(\theta) \tag{5.56}$$

$$L = T - V = \frac{1}{2} I_{eq} \dot{\theta}^2 + mgr\cos(\theta) \tag{5.57}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = T_\alpha \tag{5.58}$$

$$\ddot{\theta} = \frac{1}{I_{eq}} T \sin(\alpha) - \frac{g}{r}\sin(\theta) \tag{5.59}$$

The EOM obtained is equivalent to the one from the previous section, where the coefficients are dependent on the system inertia rather than on the mass. The control technique is repeated to develop the controllers. To validate the entire setup, a initial external perturbation is applied to the pendulum. This perturbation, drives on a fixed time interval, the pendulum from and equilibrium point at 0 Deg to another unstable point at a 40 Deg. It can be seen from the time response of the pendulum in Figure 5.35 how the controller, compensate the system dynamics driving back the pendulum to the initial condition.

5.5 CONCLUSIONS

In this chapter, a novel device with redundant control inputs for satellite attitude control is presented. The closed-form solution for the kinematic input-output map is given. Additionally, a more general and flexible approach to develop said map that can be used also for the dynamic model is developed.

A vision feedback controller based in passive features and a planar homography

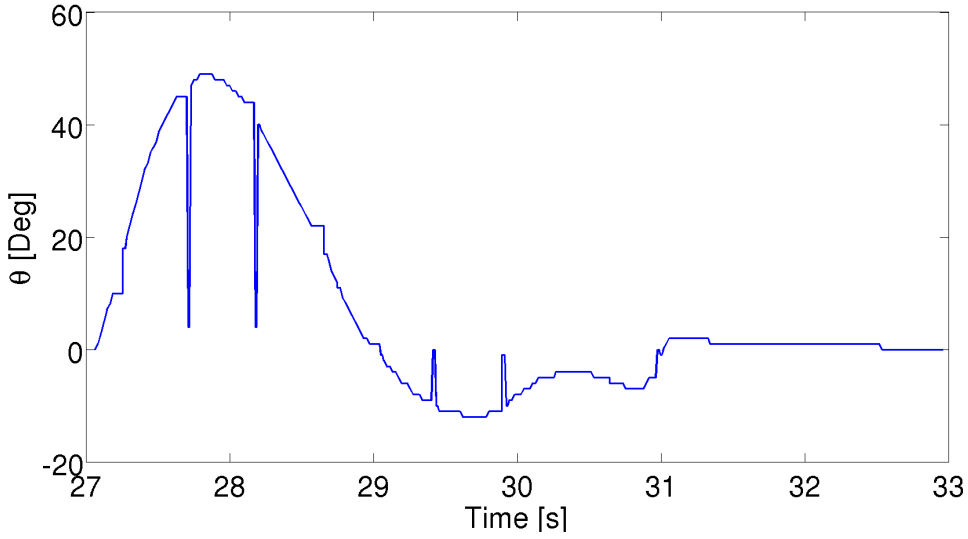


Figure 5.35: Pendulum return to initial condition

is developed. The least squares solution is generalized and a Kalman filter is implemented to improve the controller trajectory tracking performance. Kinematic controllers based on the models developed in the chapter are implemented, and their efficiencies are compared.

An extensive uncertainty analysis is performed to show how the closed kinematic chain of the CWJ helps reduce pointing error as opposed to an open chain serial manipulator. General analytical expression for both type of manipulators are developed and discussed. Results from Monte Carlo simulations and the unscented transform validate the analysis.

The design of a novel apparatus used to simulate satellite attitude dynamics in a laboratory and on a micro gravity vehicle is discussed. The Implications of the test flight are analyzed and the features required for a safe operation are present.

Finally a set of controllers for the plant are synthesized and its performance is

evaluated on a ground test and on a ground test connected to a simulator. The outcome of these experiments validates the work presented in this chapter

Chapter 06

OTHER APPLICATIONS OF THE SMARTAI SYSTEM

A robot must obey the orders given[...]

Isaac Asimov, Runaround

6.1 INTRODUCTION

In the present chapter a set of experiments to showcase the GCS capabilities is presented. These experiments are performed in highly heterogeneous conditions with different types of vehicles. The vehicles that participate are a quad-rotor, a simple rover and a plane. Indoors and outdoors experiments are shown. Some experiments are with a single vehicle and some others use a multiple vehicle configuration. Each experiment has its own unique features such as mission objective, sensors and scenario. The main objective of these experiments is to demonstrate the versatile capabilities of the GCS and how easily it can be adapted and reused. Also this section presents the different layouts and configurations required.

In the first experiment presented, an off-the-shelf quad-rotor that integrates a

look-down camera is used to perform visual navigation and auto-landing. This vehicle is not capable of image processing, hence these computationally expensive tasks are performed on the GCS.

In the second experiment, a rover and a quad-rotor are used on a cooperative schema. The rover follows a route and the quad-rotor escorts it. This "follow-me" mission replicates a wide set of scenarios where a vehicle is following a route and is being tracked by a helicopter.

The final experiment presents a popular autopilot on a surveillance mission. For this mission the airplane is equipped not only with the autopilot and radio link, but also with a video system. The video system consists of a camera and a radio link. Although the autopilot can not update the route while flying, thanks to the GCS the plane can be rerouted from the ground.

6.2 ARDRONE PARROT VISUAL LANDING

A key issues that need to be addressed on the automation of any air vehicle is the take off and landing phases of the flight. Multiple approaches exist to handle the autonomous landing capabilities on vehicles. Vision based autonomous landing keeps gaining popularity among researches.[159–162] The research not only focused on aircraft but also on space vehicles.[159] Extensive research on visual landing related to UAV is focused on rotor crafts.[160–162] Yu et. al.[162] presented a method to land a helicopter using a stereoscopic camera. Yakimenko and others[163] describe a mechanism used to land on shipboard with the use of infrared features. The method proposed on[164] also requires the use of active features. The approach presented by

Merz[160] requires a custom pattern to extract passive features. These methods that rely on features works solving the ego-motion problem, namely by identifying how the known features on a picture have move the movement of the camera can be estimated.

In this section we present a new approach where the pattern is extracted using the Canny edge detector[165] to extract the image edges, then the Hough transform[166] to obtain the relative position of known features in the image. With this features is possible to use the homography approach presented in Appendix C. For this experiment an ArDrone Parrot flies inside a known workspace, and a Vicon Camera System is used to obtain the absolute position of the vehicle. The objective of the mission is to fly the Parrot in a search route to identify a landing area using the on-board camera, once the area is identified the Parrot aligns itself to it and land.

IMPLEMENTATION

The image processing techniques used for this experiment are discussed in Appendix B. First the edge extraction is presented and then in Appendix B.0.5 the Hough transform practical implementation is discussed. The Hough transform is used to search for a know shape, which means will only detect a predefined shapes. For our experiment the landing pattern is defined and completely determined by the use of the industry standard[167] hence it is possible to use this pattern extraction technique. The H shape is decomposed in a set of lines that then can be identified. Once all lines have been identified the size which is related to the distance to the

objective and orientation can be estimated. If the identified objective value is below a reference threshold then the identification processes can be considered as a failure and the image is discarded. The identified pattern in the image can be related to an object or in this case to a mark in the physical word by solving the homography problem. Using a filtering technique such as the one presented in Section 5.3.5 its possible to extract a distance vector from the vehicle to the landing area

The vehicle used for this experiment is an ArDrone Parrot quad-rotor. This vehicle has a simple on-board autopilot that can only perform attitude tracking. By default the Parrot will remain on a stable horizontal position. The commands that can receive are roll, pitch, yaw rate and vertical speed. The vehicle has no meanings of position tracking. Even more, due to the sensors drift the vehicle will not hold the position even when the attitude is set to 0 roll, 0 pitch and 0 yaw rate. The vehicle is controlled over a wifi network using a UDP port.

In Figure 6.1 the computer and software distribution is presented. The hardware used for the experiment is presented in light-blue. Four computers are used, one that runs the MP, one for the PRC, one for the Vicon system, and another one for the SS. Al computers are connected to a local network via a router. The vicon computer access the required cameras trough the ViconMX a proprietary hardware. Since the Vicon protocol are proprietary and have several requirements and constrains a broadcaster (SLBRoadcaster) is required. The broadcaster uses the Vicon API to acquire the Parrot position and broadcast it to the network on a simple UDP package. The PRC runs in a dedicated single board computer with two network interfaces. The wireless interface is used to control the parrot over a

UDP port. The wired interface is used to connect to the mission planner using the GCS internal protocol. The SS requires two network cards as well, the first one is used to connect to the MP and the wireless is used to connect to the parrot video port. The SS processes the video stream, generates images and performs the image processing required for the pattern extraction. The MP reads the SLBRoadcaster data and the SS data, decides if the objective have been identified and send a new route to the PRC.

For this experiment, several reference frames are used as shown in Figure 6.2. First a inertial reference frame is considered to be on the center of the workspace. A body reference frame is considered to be attached to the quad-copter center of gravity. The camera reference frame is considered to have the center in the camera lens. In Figure 6.2 an image is presented demonstrating the calibration process. The red vector indicates the position of the objective in the inertial reference frame. In green its represented the Inertial position of the vehicle. In Blue the relative distance between the vehicle and the objective is presented as observed from the camera.

The Parrot carries a low resolution 640×480 RGB camera that broadcast, a video stream at approximately $10Hz$ over a UDP port.[168] The Sensor Station (SS) will receive this images and process them accordingly to identify the landing pattern. The Sensor sub-module of the SS process the video stream. When enough data has been received to generate a frame the module will convert the video format into a RGB matrix with dimensions $640 \times 480 \times 3$ where each elements represent the intensity of the red, green or blue color for each pixel. The Data Processor

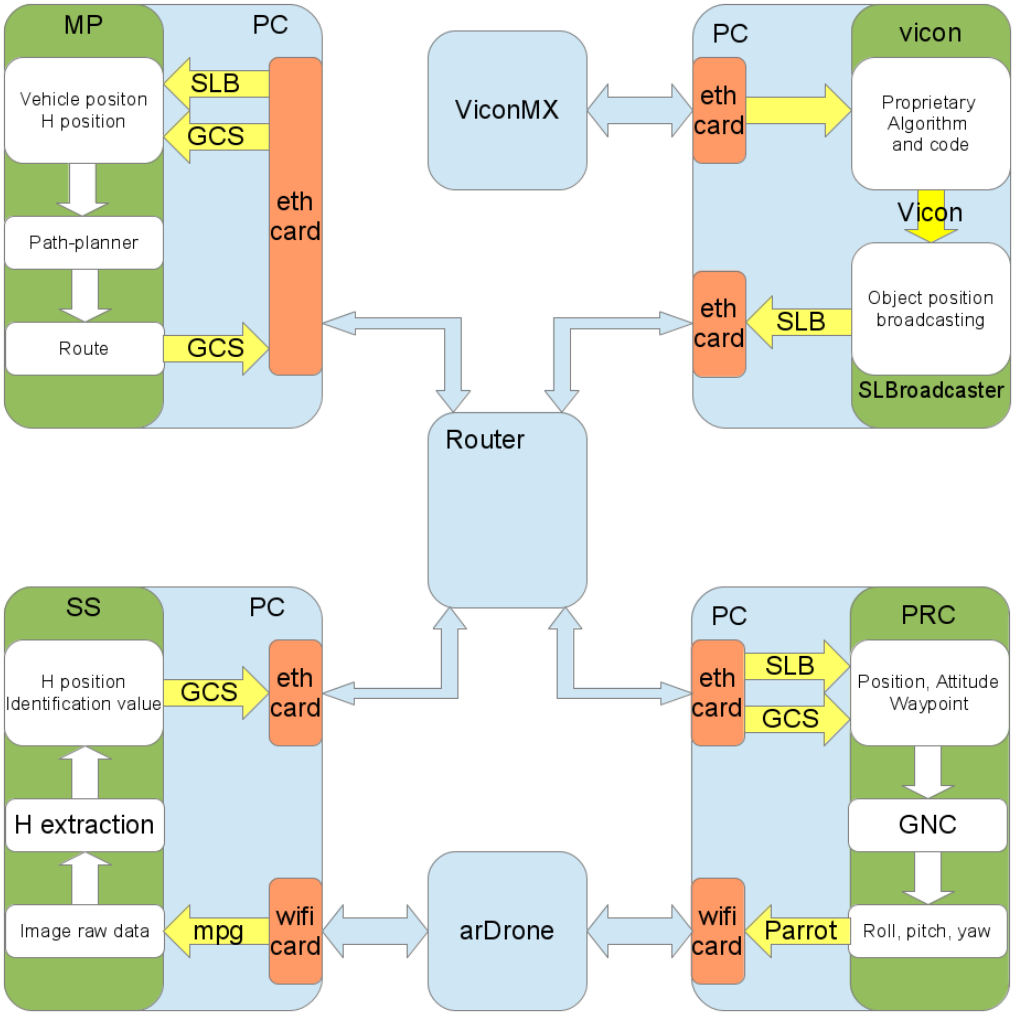


Figure 6.1: Hardware and Software distribution used in the auto-landing experiment

(DP) sub-module will take the RGB matrix as input and will perform all image processing techniques. The first step is to convert the RGB matrix to a gray-scale intensity image using Equation (6.1), sample images are shown in Figure 6.3. Using

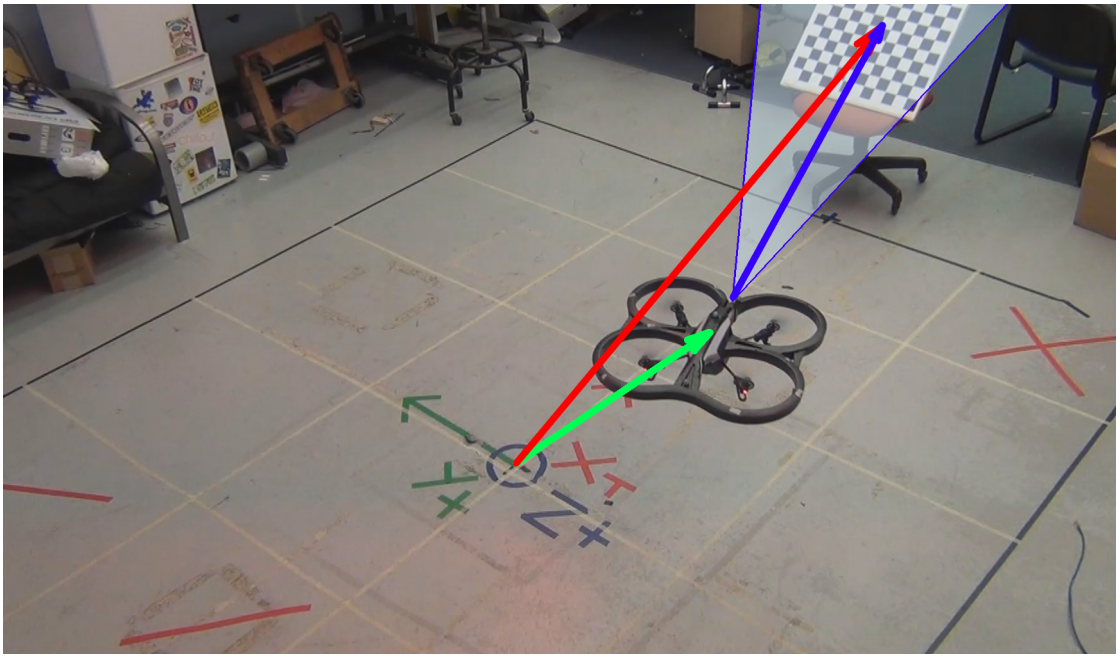


Figure 6.2: References frames used for Parrot camera calibration

standard c++ functions the canny edge detector is implemented. A sample image as acquired with the parrot and the outcome of the shape identifications is shown in Figure 6.4.

$$I_{i,j} = 0.2989 \times R_{i,j} + 0.5870 \times G_{i,j} + 0.1140 \times B_{i,j} \tag{6.1}$$

The Canny edge detector technique implemented in this section is highly sensitive to the σ value chosen for the Gaussian filter since a small values will not filter out all camera noise, and a high value my blur in excess the edge and difficult the identification process. For this implementation, due to the low resolution of the parrot camera the blur filter can be kept to a low dimension. The key components in the algorithm for this type of edge detector is the thresholding, set its value

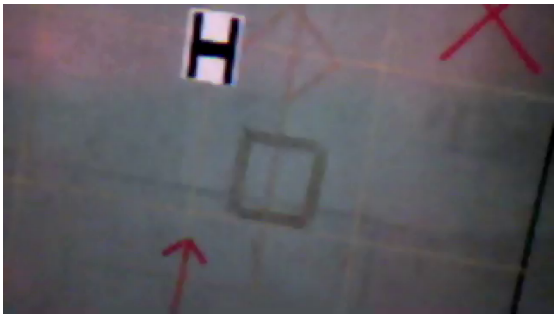


Figure 6.3: Parrot bottom camera sample

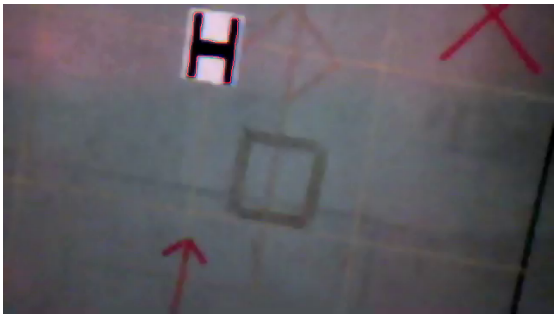


Figure 6.4: Edge detection result superimposed with original image

is not trivial and clearly, for different values of the thresholding the edges will be different. In this implementation the thresholding is done using visual feedback calibration, but since the light conditions inside the laboratory remain sensibly constant throughout all day and the floor is kept clean the algorithm gains can be reliably reused. All the calibration gains are set in a XML configuration file that on runtime is parsed by the ConfigParser component.

The identification result from the DP is sent to the MP. The message contains not only the relative vector obtained from the homography solution but also the identification confidence value which is given by the accumulator used in the Hough pattern search. With this data the MP can decide wherever to reroute the vehicle or not based on the confidence of the identification.

Since the MP knowns the position of the vehicle on a real time basis, it can obtain an absolute position for the landing path (which was unknown at the beginning of the experiment) by adding the relative position of the vehicle and the distance to the landing pattern. As the mission develops over time the identification

indicator keeps increasing and the absolute position converges to the real position, this behavior helps eliminating spurious response. Once the identification index reaches a maximum and the distance is vertical, it serves as an indicator that the Parrot is over the landing pattern, with the use of the yaw control can be aligned before landing.

This Algorithm is really expensive computationally due to the nature of the gradient operator which demands numerical derivatives to be computed two times. The dimension of the pictures is kept to a minimum size which help to reduce the computational burden. The hysteresis filtering method requires a iterative process with a kernel that operates at each pixel until it converge to a final solution. However since the black over white high contrast of the H shape offers a high magnitude edge, setting the threshold to a high level helps reduce by several order of magnitudes the iterations required by the hysteresis process.

EXPERIMENT OUTCOME

The methodology presented in this section proves the importance in object recognition and enhancement of pictures in the aerospace industry. These methods can be used to aid a pilot on low visibility condition and even can be used to land autonomous vehicles. Since it can recognize objects partially covered by others, with missing parts, in low quality pictures the vehicle can identify the object even before the entire landing pattern is inside the field of view. The applications for this method are numerous. However, most of the image processing techniques are

computational expensive and require, most of the time, a calibration procedure or feedback from a user. It, is important to notice that the computational cost of this method will require an equally capable computer severely limiting its application in embedded devices. Since the calibration varies based on time of day, lighting condition, camera, pattern just to mention the more important, the need of a configuration mechanism is evident.

This experiment also demonstrates some of the most powerful features of the GCS, the ability to act as a distributed system. The image processing is at low frequency limited up to a maximum of $10Hz$ due to the limitations imposed by the ArDrone camera, also the Parrot platform imposes a minimum frequency on the control loop of at least 1000 Hz. Clearly, there is a incompatibility in the frequency to be excecuted on a single thread. Using a main thread with a frequency divisor will impact the precision of the frequency control. It may seem that a multiple thread application could be enough, however, to maintain a constant frequency with a variable processor load as the once imposed by the image processing its hardly stable without the use of real time implementation tools, which impose restriction and limitations on the program design. On the other hand, using the distribute capabilities each module can be run in separated computers allowing the image processing part stress the micro processor while the real time highly sensitive code is implemented in a different computer.

The second important feature that should be brought into attention is the configuration utility. The config parser allows to recalibrate the entire experiment on the fly by the use of rather simple configurations files. This property may sound

trivial, however it saves the developer a considerable amount of time by eliminating the need of code recompilation.

6.3 ARDRONE PARROT OBJECT TRACKING

The objective of this experiment is to show a cooperative interaction exploiting the capabilities of the GCS. As mentioned in the previous section, the quad-rotor used in this experiments has only attitude control. The other vehicle that takes part in this experiment is a UGV with two degrees of freedom. The first degree is the linear displacement velocity. The second degree of freedom is the yaw rate control. This vehicle, as opposed to the quad-rotor does not have an attitude control, it can not adjust its bearing autonomously. To enhance the vehicles capabilities both are connected to its own PRC. The experimental hardware and software layout is shown in Figure 6.5.

In this mission the MP generates a route for the UGV and sends it to the PRC that controls it. The PRC uses the position and attitude from the Vicon stream to track the required waypoints. To track the waypoints the PRC generates a set of commands to assure the vehicle reaches the required position. Based on the route sent to the UGV the mission planner sends routes to the parrot to follow the UGV. The generated routes compensates for different velocities, and different position. On this experiment the parrot lags behind the vehicle to emulate a follow-me behavior. In Figure 6.6 an image of the experiment is presented. On the image can be seen the absolute and relative position of the vehicles. A video showing the quad-rotor tracking the ground vehicle can be seen in Figure 6.7.

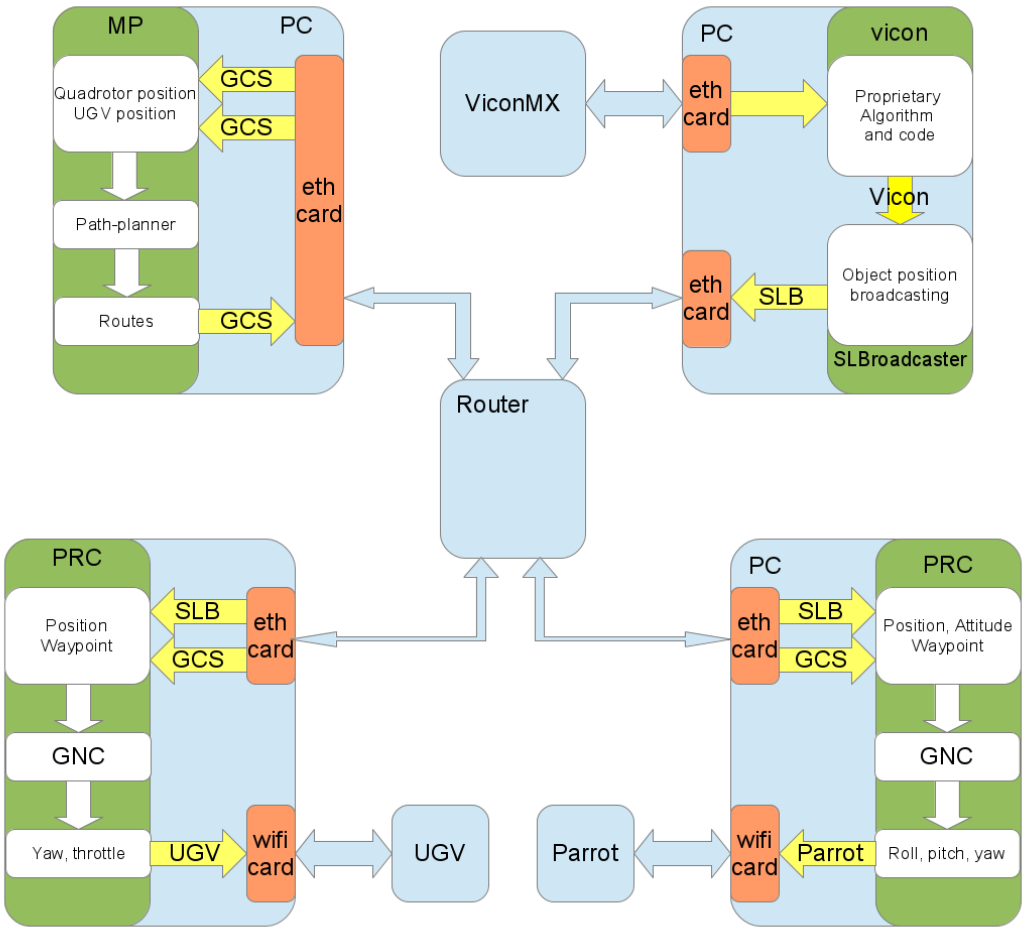


Figure 6.5: Computer layout and connections diagram

EXPERIMENT OUTCOME

The simple experiment presented on this section showcase the several features of the GCS. It can be seen first, how the system can integrate seamlessly with different types of vehicles. In this experiment the GCS is not only used to perform

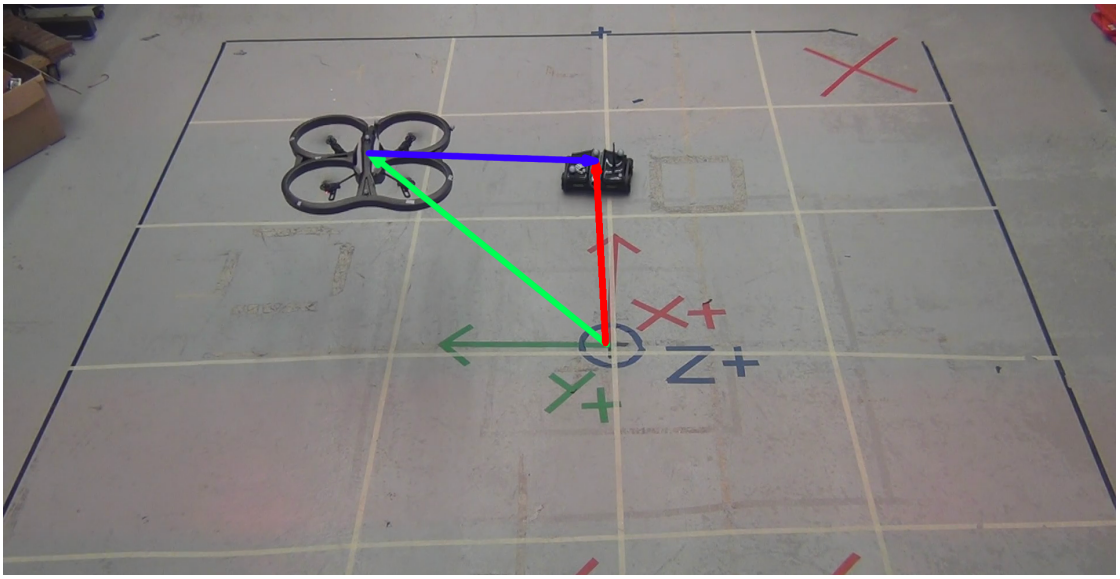


Figure 6.6: References frames used for cooperative route tracking

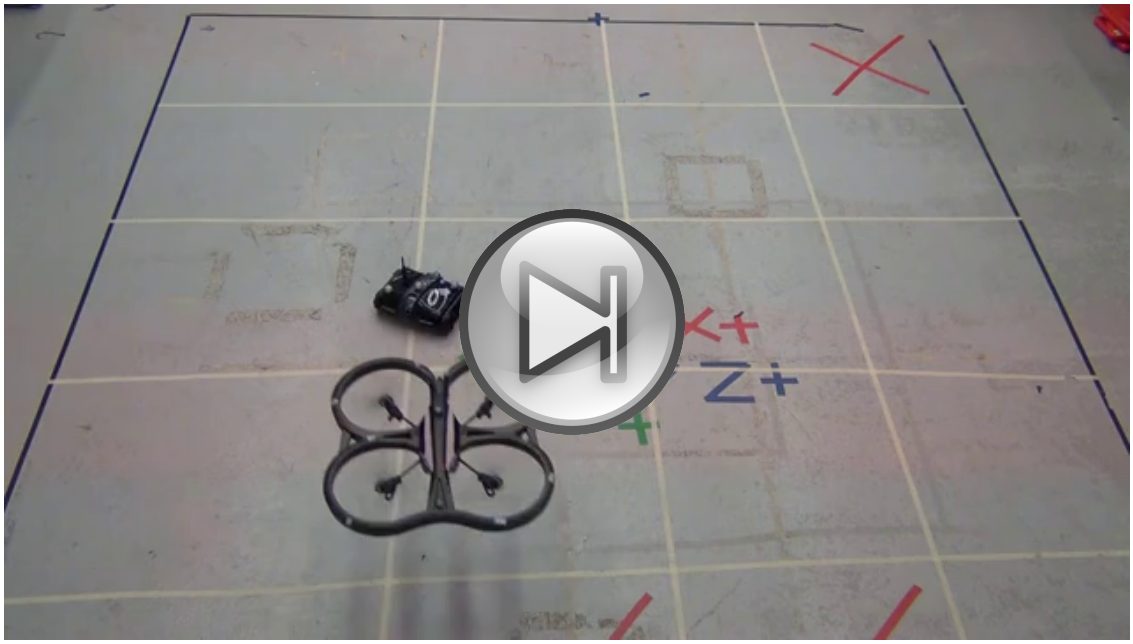


Figure 6.7: Parrot tracking UGV

interactions based on multiple vehicles but also to enhance the vehicles autopilots. This experiments works with two vehicles that are not capable of track waypoints, however with the proper configuration and the GCS the route tracking capability was granted. All the algorithms that require position information are integrated with the Vicon, which shows how simple is to add external sensors that can be shared by multiple devices.

6.4 VIDEO SURVEILLANCE PLANE

To further demonstrate the flexibility of the GCS we use an off-the-shelf autopilot to control a Bixler Plane. The selected autopilot is the arduplane 2.6. This autopilot has the capability of waypoint tracking, and can store only one route on board. This route can not be updated on real time. However, the route can be overridden with a single waypoint. When a waypoint is used to override the route the vehicle will enter a waiting circular pattern when the waypoint is reach. For this experiment the GCS performs the route tracking on the PRC and sends one waypoint at the time. With this configuration the route can be changed on the fly since the autopilot does not need to track the entire route but just the current waypoint.

For this experiment the MP, ATC and PRC only are used. To reduce the deployment burden the three components run on the same computer as shown in Figure 6.8. The computer uses a usb radio modem to establish a wireless communication with the autopilot. To assure the communication is reliable the MAVLink protocol is used. The PRC receives the position as reported by the autopilot which carries a GPS receiver. The PRC forward the position to the ATC and the ATC

forwards it to the MP. Based on the current position the MP generates a route that is sent to the ATC. The ATC performs the routes checks and validations and once is approved it is send to the PRC. The PRC will decide whenever a waypoint is reached based on the vehicle position. Once a waypoint is reach will forward the next waypoint on the queue. In case that the autopilot reports that the waypoint was reach the next waypoint on the queue is sent. When the last waypoint is reach a Return-To-Launch (RTL) command is sent to bring back the plane to home. The MP generates a route based on predefined rules. For this simple experiment a predefined route is sent to the PRC.

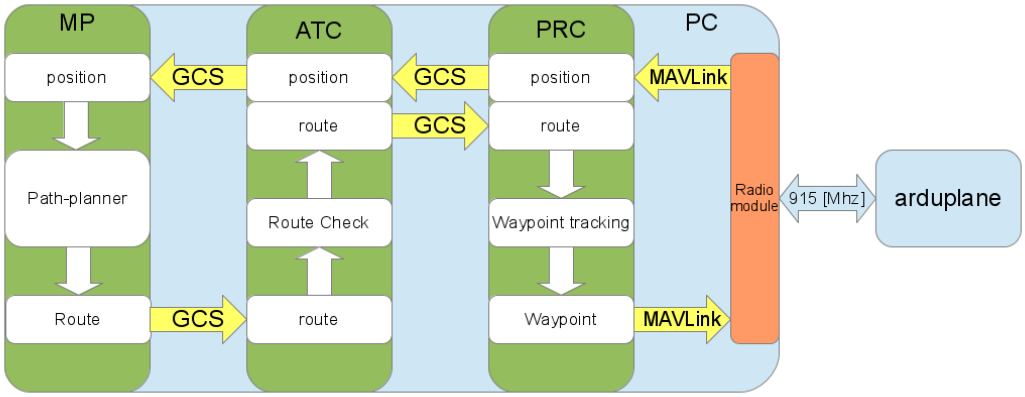


Figure 6.8: GCS layout required for arduplane waypoint tracking

For this test the plane carries also a small NTSC video camera. This camera is attached to a wireless transmitter. Basically the plane is configured to perform a surveillance tasks. Unlikely the previous missions on this experiment the camera is not connected to the GCS but rather to an external computer. If this vehicle is intended to be used for surveillance it may be required that the operator does

not have access to the video feed to ensure privacy is respected; also, showcase how the entire system can be reconfigure to address the need to carry a custom sensor which may not be possible to connect to the GCS. In this scenario external routes should be received by the IC and forwarded to the MP. A demo flight can be seen in Figure 6.9.

Figure 6.9: Bixler airframe with Arduplane on a waypoint tracking flight

EXPERIMENT OUTCOME

The route and the waypoint tracking is shown in Figure 6.10. Although this is a rather simple experiment is really important since demonstrate the flexibility of the GCS. Not only the GCS can operates indoors and outdoors, but also, the algorithms

can seamlessly be reused. Mission, ground avoidance algorithms among others can be reused and selected with a simple configuration file without the need to adapt the code to the particular vehicle being use. These features reduces the deployment time, and demonstrate how any vehicle, no matter how complex the autopilot may be can be integrated to the GCS with a simple wrapper. An importante remark, to make is how the vehicle can carry any required sensor with out the need of a driver. Although the GCS can integrate to sensors to expand the capabilities of the vehicle and mission; as is demonstrated on this experiment, this is not required and the sensor can be treated as a blackbox.

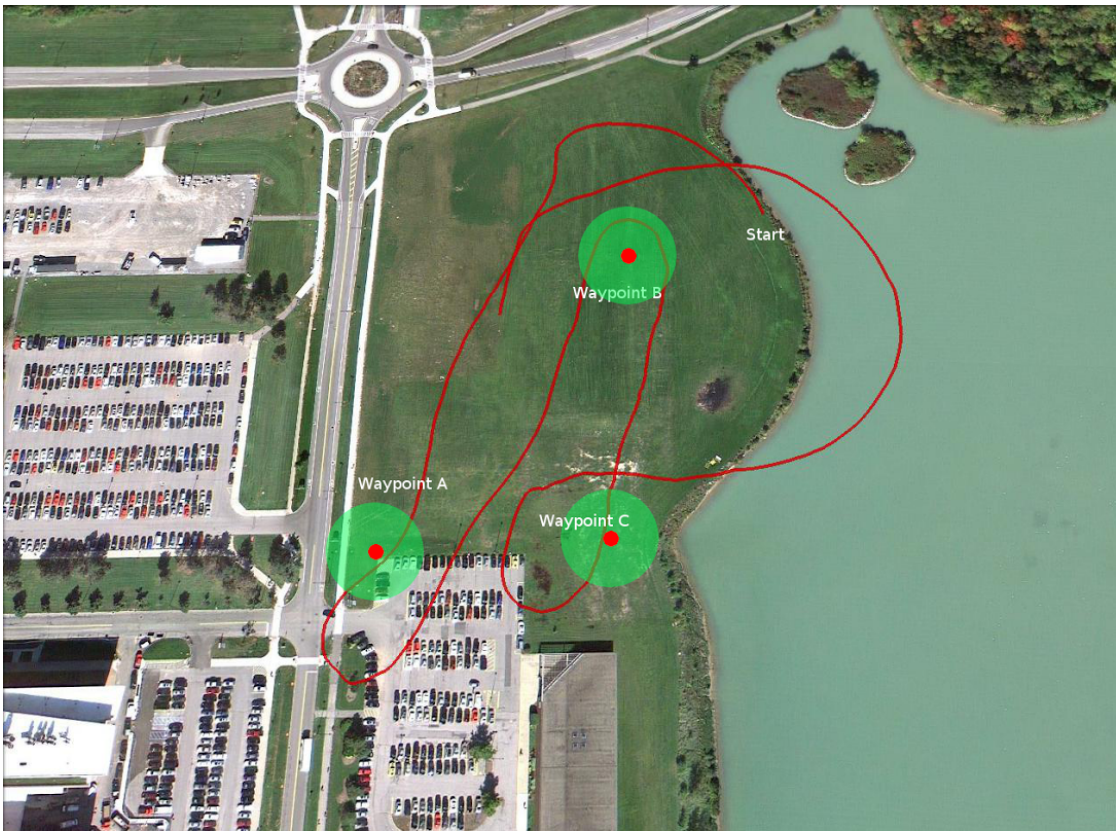


Figure 6.10: Bixler airframe with Arduplane tracking waypoint outcome

6.5 CONCLUSIONS

In the present chapter a set of experiments to showcase the GCS capabilities are presented. This experiments are performed in highly heterogeneous conditions with different types of vehicles. The vehicles that participate are a quad-rotor, a simple rover and a plane. These experiments take place indoors and outdoors. The outcome of the experiments demonstrate the flexibility of the system. The first experiment presents how image processing techniques can be used to perform autonomous navigation and landing. A second experiments demonstrate how the GCS can be integrate with multiple vehicles to generate a cooperative environment. The final experiment showcase the indoors-outdoors analogy used on the system. Algorithms can be reused in different scenarios with different coordinate systems with minimum modifications. On all the experiments presented the GCS is used not only to control the vehicles but also to enhance their capabilities.

Chapter 07

CONCLUSIONS

> If we knew what it was we were doing,
> it wouldn't be called 'research,' would it?
>
> Albert Einstein

7.1 SUMMARY

The main objective of the current dissertation is to provide a new abstraction layer between the data acquisition layer and the implemented algorithms. We present a systematic approach to decouple controller and filter design from hardware selection. The main goal of the SmartAI and Ground Control Station is to provide a simple, yet flexible configuration that can be scaled to match a variety of standard missions. This is made possible by utilizing state of the art software engineering techniques such as modular design, object oriented design and patterns. This approach also addresses the performance decay due to hardware limitations. The outcome of this work is an integrated environment to develop, validate, and test

algorithms. This framework is tested in different applications. As a demonstration a flight controller is implemented. All critical problems required for a safe flight, including manual override of autopilot commands, robust communication between the vehicle and the ground station, and low battery alerts have been addressed. Other problems such as input saturation or radio communication loss, that although impact flight quality but do not pose a threat to flight safety, are classified as non-critical. Methods to address the non-critical issues have been taken into account and considered on the design. These features, however, due to lack of time have not been implemented, leaving the implementation, testing and debugging as future work.

In Chapter 1 the system layout is introduced. The software design is presented first. The software design covers the Unified Modeling Language diagrams of the main components. Basic algorithms used for collision avoidance are presented when discussing the Air Traffic Module. Then, advantages, disadvantages, and limitations of the system are discussed. Several examples for different applications of each module are presented in this dissertation.

In Chapter 2 a new modeling technique for designing controllers in the presence of hardware constraints is introduced. Adding a new set of inputs that represents the discretization of the digital servos allows the designer to model the hardware constraints independently from the main plant while letting him analyze the controller effects.A mathematical nonlinear model of an off-the-shelf remote controlled airplane is presented; this model is then linearized. A LQR controller is designed and synthesized for the linear plant. The controller performance is then evaluated

with the linear model.

Chapter 3 presents the simulation capabilities of the proposed framework. The requirements for a hardware-in-the-loop simulation are discussed. The numerical solvers embedded into the framework are presented. A systematic approach for controller validation is presented. An example is presented for a satellite attitude controller. Finally the hardware-in-the-loop integration with FlightGear is introduced and the controller developed for the aircraft is validated.

Chapter 4 present the flight test results. Several implications regarding the implementation of the flight controller are discussed. Details on how the antennas have been mounted to reduce the interference is given. The technique used to shield the magnetometer to reduce the overall magnetic drift is presented. System initialization and data backup are also discussed. The route tracking algorithm and the results of a test performed on a ground vehicle are presented. Finally flight test results of the controller developed in Chapter 2 are presented.

Chapter 5 covers the design of a novel microsatellite attitude controller. The controller for the experimental layout is designed using the technique proposed in Chapter 2 and implemented with the smartAI. The Carpal Wrist Joint is first introduced, and its characteristics discussed. A visual filter to improve the pointing estimates is develop. An experimental apparatus to test the attitude manipulator in a 0G environment is discussed. Finally a counterweight system is introduced to perform benchtop experiments. Practical results are shown.

In Chapter 6 integration between the Ground Control Station and off-the-shelf vehicles and autopilots are discussed. First the integration with the ArDrone Par-

rot is presented. A set of experiments using the Parrot and the Vicon Tracking System are developed to validate the system integration. An experiment that performs image processing in real time is performed to showcase the system scalability. A cooperative route tracking algorithm between the Parrot and an autonomous ground vehicle showcase the flexibility of the system. Finally a flight test using an off-the-shelf autopilot mounted on a Bixler airframe carrying a wireless camera demonstrates a live video feed flight.

7.2 FUTURE WORK

The autopilot developed in this dissertation has been tested and used as a proof of concept. The reliability of the system has been demonstrated with several iterations of experimental procedures described in this dissertation. This vehicle can act as a reliable test bed to allow different controllers and filters algorithms to be tested without fear of compromising the platform. To develop model-based filters and controllers, an identification method can be used to obtain a more reliable model. Techniques such as OKID/ERA[169] can be used to generate a more accurate state-space representation. Using error propagation methods such as the Unscented Transform,[148] the uncertainty in the system identification can be obtained.[170] This uncertainty model can then be used to develop a robust controller using techniques such as μ-Synthesis. Extra sensors should be installed on the platform to provide angle of attack measurements, and sideslip angle measurements. Additionally, a pitot tube to measure air speed must be added to obtain airspeed tracking. Relative altitude sensors such as sonars can be installed to provide automatic takeoff and

landing capabilities. Development of more advanced controllers incorporating on-line system identification and gain tuning is among the future tasks. One of the most important features of the autopilots introduced in this dissertation is the capability to switch midair controllers. This allows an easy method for comparing the controllers, using time domain metrics such as steady state error, rise time, and settling time, among others.

Appendix A

AIRCRAFT MODEL

LINEAR MODEL

$$A = \begin{bmatrix}
-0.4367 & -0.0136 & 0.4431 & 0 & -8.3423 & 0 & 0 & -0.5146 & 0.0091 & 0 & 0.0132 \\
-0.0004 & -1.4591 & -0.0000 & 8.2418 & -0.0044 & 0 & 0.3620 & 0 & -16.9975 & 0 & 0 \\
-1.0030 & -0.0345 & -8.2183 & -4.5101 & -4.3260 & 0 & -0.0091 & 14.9024 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0000 & -0.0000 & 0 & 1.0000 & -0.0000 & -0.0213 & 0 & 0 \\
0 & 0 & 0 & 0.0000 & 0 & 0 & 0 & 1.0000 & -0.0010 & 0 & 0 \\
0 & 0 & 0 & -0.0000 & -0.0000 & 0 & 0 & 0.0010 & -1.0002 & 0 & 0 \\
0.0003 & -0.5649 & 0.0000 & 0 & 0 & 0 & -1.3964 & -0.0071 & 0.6664 & 0 & 0.0000 \\
0.0235 & -0.0000 & -1.0448 & 0 & 0 & 0 & -0.0971 & -3.9779 & 0.1471 & 0 & 0 \\
-0.0001 & 0.1308 & -0.0000 & 0 & 0 & 0 & -0.2229 & -0.0450 & -0.0825 & 0 & 0.0000 \\
-0.0213 & 0 & 0.9998 & 0 & 14.3094 & 0 & 0 & 0 & 0 & 0 & 0 \\
71.8047 & 2.4215 & 4.0911 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -7.4528
\end{bmatrix}$$

$$B = 10^3 \begin{bmatrix} 0 & -0.0004 & -0.0009 & 0 \\ 0 & 0 & 0.0057 & 0 \\ 0 & -0.0000 & -0.0000 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.0066 & 0 & -0.0006 & 0 \\ 0 & -0.0117 & 0 & 0 \\ 0.0006 & 0 & -0.0030 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8.0676 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.9998 & 0.0299 & 0.0506 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0012 & 0 & 0.0587 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0000 & 0.0587 & -0.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 \end{bmatrix}$$

LONGITUDINAL MODEL

$$A = \begin{bmatrix} -0.4367 & 0.4431 & -8.3423 & -0.5146 & 0 & 0.0132 \\ -1.0030 & -8.2183 & -4.3260 & 14.9024 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0.0235 & -1.0448 & 0 & -3.9779 & 0 & 0 \\ -0.0213 & 0.9998 & 14.3094 & 0 & 0 & 0 \\ 71.8047 & 4.0911 & 0 & 0 & 0 & -7.4528 \end{bmatrix}$$

$$B = 10^3 \begin{bmatrix} -0.0003 & 0 \\ -0.0000 & 0 \\ 0 & 0 \\ -0.0092 & 0 \\ 0 & 0 \\ 0 & 8.0676 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.9998 & 0.0506 & 0 & 0 & 0 & 0 \\ -0.0012 & 0.0587 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \end{bmatrix}$$

CONTROLLABILITY AND OBSERVABILITY MATRICES

$$Co = \begin{bmatrix} -0.31608 & 0 & 4.8675 & 106.74 & -5.0881 & -842.09 & 452.47 & 6348.9 & -5460.6 & -47801 & 44656 & 3.5959e+05 \\ -0.0067309 & 0 & -136.67 & 0 & 1703.2 & -107.05 & -14189 & 1761.7 & 89805 & -19633 & -3.9895e+05 & 1.7886e+05 \\ 0 & 0 & -9.1957 & 0 & 36.579 & 0 & -2.6069 & 2.5054 & -1769.2 & 82.114 & 21873 & -2018.2 \\ -9.1957 & 0 & 36.579 & 0 & -2.6069 & 2.5054 & -1769.2 & 82.114 & 21873 & -2018.2 & -1.8097e+05 & 27419 \\ 0 & 0 & -4.3311e-11 & 0 & -268.32 & -2.2724 & 2226.3 & -89.099 & -14233 & 1662 & 64584 & -17436 \\ 0 & 8067.6 & -22.724 & -60126 & -40.247 & 4.5577e+05 & 6902.5 & -3.4577e+06 & -77003 & 2.6232e+07 & 5.4919e+05 & -1.9902e+08 \end{bmatrix}$$

$$Ob = \begin{bmatrix} 0.99979 & 0.050604 & 0 & 0 & 0 & 0 & \\ -0.0011827 & 0.058665 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 1 & 0 & \\ & -0.48736 & 0.027077 & -8.5594 & 0.23959 & 0 & 0.013227 \\ -0.058322 & -0.48265 & -0.24392 & 0.87485 & 0 & -1.5648e-05 & \\ 0 & 0 & 0 & 1 & 0 & 0 & \\ & -0.02129 & 0.99977 & 14.309 & 0 & 0 & 0 \\ 1.1411 & -0.63466 & 3.9486 & -8.8581 & 0 & -0.10503 & \\ 0.52896 & 3.0266 & 2.5745 & -10.887 & 0 & -0.00065498 & \\ & 0.023473 & -1.0448 & 0 & -3.9779 & 0 & 0 \\ & -0.99344 & -8.2259 & -4.1474 & 29.219 & 0 & -0.00028167 \\ -7.6112 & 14.547 & -6.7737 & 29.14 & 0 & 0.79785 & \\ -3.5692 & -13.268 & -17.506 & 90.712 & 0 & 0.01188 & \\ 0.94426 & 12.753 & 4.324 & 0.2417 & 0 & 0.00031055 & \\ 9.3497 & 36.634 & 43.873 & -242.45 & 0 & -0.011044 & \\ 46.708 & -150.1 & 0.56583 & 98.006 & 0 & -6.0469 & \\ 17.848 & 12.734 & 87.173 & -574.24 & 0 & -0.13576 & \\ -13.175 & -104.64 & -63.047 & 192.93 & 0 & 0.010178 & \\ -47.31 & -43.66 & -236.48 & 1549.5 & 0 & 0.20601 & \\ -301.75 & 1127.2 & 259.7 & -2650.2 & 0 & 45.685 & \\ -43.793 & 502.67 & -203.98 & 2552 & 0 & 1.2479 & \\ 115.96 & 652.61 & 562.59 & -2383.1 & 0 & -0.25017 & \\ 115.61 & -1280.2 & 583.54 & -7026.3 & 0 & -2.1612 & \end{bmatrix}$$

LQR MATRICES

$$Q = \begin{bmatrix} 0.99971 & 0.043654 & 0 & 0 & 0 \\ 0.043654 & 0.34671 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.1}$$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{A.2}$$

$$K_L = \begin{bmatrix} 0.013473 & -0.10193 & -1.3501 & -0.46834 & 0 & 4.6644e-05 \\ 0.93709 & 0.04852 & -1.1925 & -0.073168 & 0 & 0.0010573 \end{bmatrix} \tag{A.3}$$

LATERAL MODEL

$$A = \begin{bmatrix} -1.4591 & 8.2418 & 0 & 0.36196 & -16.998 \\ 0 & 3.5826e-15 & 0 & 1 & -0.021295 \\ 0 & -1.6828e-13 & 0 & 0 & -1.0002 \\ -0.56487 & 0 & 0 & -1.3964 & 0.66635 \\ 0.1308 & 0 & 0 & -0.22287 & -0.082517 \end{bmatrix}$$

$$B = 10^3 \begin{bmatrix} 0 & 5.6951 \\ 0 & 0 \\ 0 & 0 \\ 6.621 & -0.64963 \\ 0.57219 & -2.9643 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.058749 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

OBSERVABILITY AND CONTROLLABILITY MATRICES

$$Co = \begin{bmatrix} -0.31608 & 0 & 4.8675 & 106.74 & -5.0881 & -842.09 & 452.47 & 6348.9 & -5460.6 & -47801 & 44656 & 3.5959e+05 \\ -0.0067309 & 0 & -136.67 & 0 & 1703.2 & -107.05 & -14189 & 1761.7 & 89805 & -19633 & -3.9895e+05 & 1.7886e+05 \\ 0 & 0 & -9.1957 & 0 & 36.579 & 0 & -2.6069 & 2.5054 & -1769.2 & 82.114 & 21873 & -2018.2 \\ -9.1957 & 0 & 36.579 & 0 & -2.6069 & 2.5054 & -1769.2 & 82.114 & 21873 & -2018.2 & -1.8097e+05 & 27419 \\ 0 & 0 & -4.3311e-11 & 0 & -268.32 & -2.2724 & 2226.3 & -89.099 & -14233 & 1662 & 64584 & -17436 \\ 0 & 8067.6 & -22.724 & -60126 & -40.247 & 4.5577e+05 & 6902.5 & -3.4577e+06 & -77003 & 2.6232e+07 & 5.4919e+05 & -1.9902e+08 \end{bmatrix}$$

$$Ob = \begin{bmatrix} 0.058749 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -0.085722 & 0.4842 & 0 & 0.021265 & -0.99859 \\ 0 & 3.5826e-15 & 0 & 1 & -0.021295 \\ 0 & -1.6828e-13 & 0 & 0 & -1.0002 \\ -0.017555 & -0.7065 & 0 & 0.64603 & 1.5433 \\ -0.56766 & 1.2835e-29 & 0 & -1.3917 & 0.66811 \\ -0.13083 & -6.0287e-28 & 0 & 0.22292 & 0.082536 \\ -0.13744 & -0.14469 & 0 & -1.9589 & 0.61657 \\ 1.7018 & -4.6785 & 0 & 1.589 & 8.6663 \\ 0.075777 & -1.0783 & 0 & -0.37704 & 2.3656 \\ 1.3877 & -1.1327 & 0 & 2.4037 & 0.98295 \\ -2.2471 & 14.026 & 0 & -8.2129 & -28.483 \\ 0.41184 & 0.62454 & 0 & -1.0516 & -1.7115 \end{bmatrix}$$

LQR MATRICES

$$Q = \begin{bmatrix} 0.0034515 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.4}$$

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{A.5}$$

$$K_L = \begin{bmatrix} -0.072229 & 9.9819 & 0.28362 & 1.5478 & -0.080349 \\ -0.026107 & -0.55264 & 9.996 & 0.036281 & -2.4798 \end{bmatrix} \tag{A.6}$$

Appendix B

IMAGE PROCESSING

FEATURE EXTRACTION

Most computer displays and image acquisition devices (e.g. cameras, scanners) represent an image as a matrix where each dot (pixel) contains three intensities values, one for each primary color: red, green and blue.[135] The resulting mixtures in RGB color space can reproduce a wide variety of colors; however, the relationship between the amounts of red, green, and blue and the resulting color is unintuitive. Furthermore, neither additive nor subtractive color models define color relationships the same way the human eye does; hence to overcome these issues new models were developed, one of these being the Hue, Saturation, Value (HSV) representation.[137] This is one of the most common cylindrical representations of points in a Red, Green, Blue (RGB) color model where **hue** is an attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors: red, yellow, green, and blue, or to a combination of two of them. **saturation** is the colorfulness of a stimulus relative to its own brightness and **value** is the total amount of light passing through a particular area. The transformation is defined

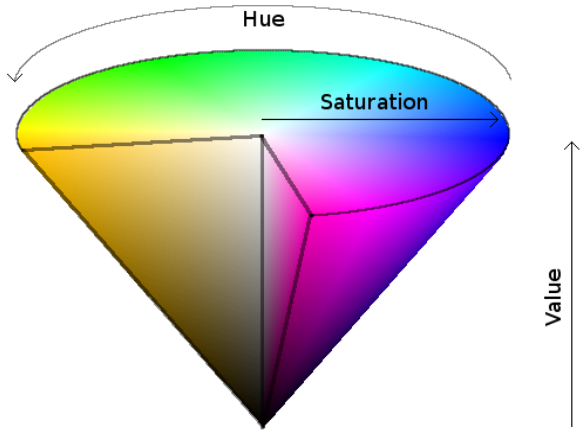


Figure B.1: Hue, Saturation, Value representation

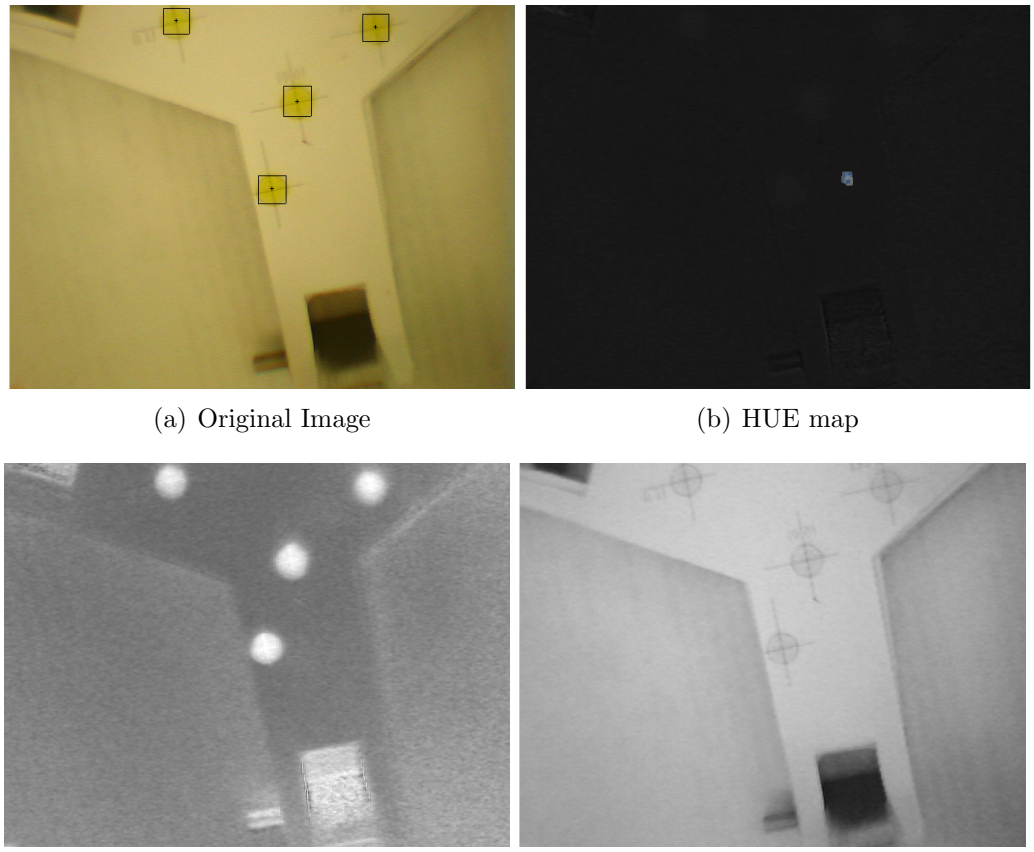
in Equation (B.1) and shown in Figure B.1.

$$
\begin{aligned}
V &= max(R, G, B) \\
S &= \frac{V - min(R, G, B)}{V} \quad if\ V \neq 0 \\
S &= 0 \quad if \quad V = 0 \\
H &= 60\frac{G - B}{S} \quad if\ V = R \\
H &= 120 + 60\frac{B - R}{S} \quad if\ V = R \\
H &= 240 + 60\frac{R - G}{S} \quad if\ V = R \\
H &= H + 360 \quad if\ H < 0
\end{aligned}
\tag{B.1}
$$

FEATURE EXTRACTION USING HSV TRANSFORMATION

With the use of the HSV transformation colors with high contrast are easily identifiable. If a pattern of high contrast color dots (i.e. red, green, yellow) is placed over a uniform background color a Hue, Saturation, Value image decomposition as shown in Figures B.2(a) to B.2(d) can be used to identify the centers of the dots. From the HSI decomposition sub image we can set calibrated thresholds where pixels laying outside a reference value are considered background and a 0 is assigned to the pixel. On the other hand the pixels laying in the predefined range are considered part of the reference circle and a 1 is assigned to each pixel. This process yields a set of three binary matrices (one per channel). Using an AND binary operator over the three matrices results in a frame with all zeros except for the zones where circles have been detected. An example can be seen in Figure B.3(a).

The identification obtained using the HUE decomposition usually yields a noisy and spurious response. To address this problem a region-growing method[136] to identify the correct areas can be used. The region-growing method proceeds by identifying clusters of neighboring 1's in the binary image matrix, searching in concentrically larger regions about a "1" until the cluster boundary is defined. A size threshold is defined to disregard small regions as "noise." Figure B.3(b). Once the spurious response is eliminated the centroid of each region can be computed and used as the circle center, i.e. the coordinates of the points as shown in Figure B.4. If the identification of the areas is repeated by changing the set of calibrated thresholds by which the binary matrices are constructed, multiple colors can be detected, allowing an easy matching between the 3D points and its equivalent in the reference

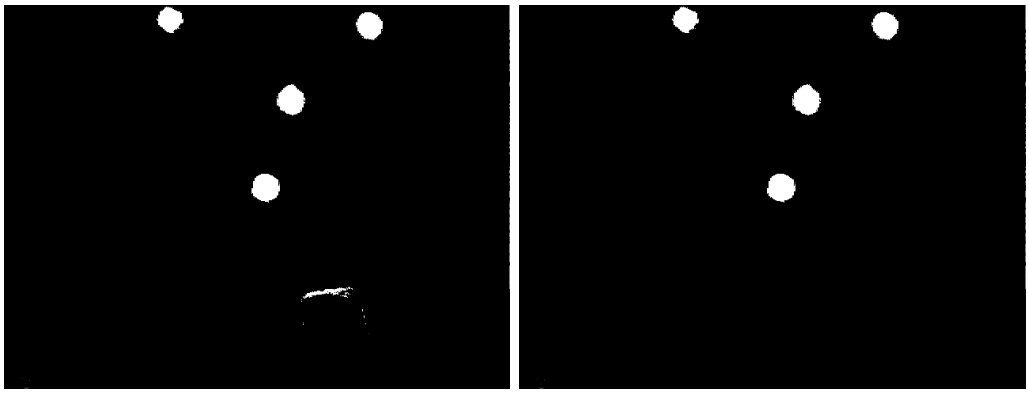


(a) Original Image (b) HUE map

(c) Saturation map (d) Intensity map

Figure B.2: Original Image and HSI decomposition

image of the object. Due to the simple operations involved on this feature extraction method, the computer power required is far less than that required by other techniques such as the one presented by Lowe,[139] or the one presented by Bay.[138] Furthermore, the image processing procedure presented in this section is easier to calibrate, requiring only the definition of a few thresholds.



(a) detected regions (b) detected regions filtered

Figure B.3: Feature extraction based on region size

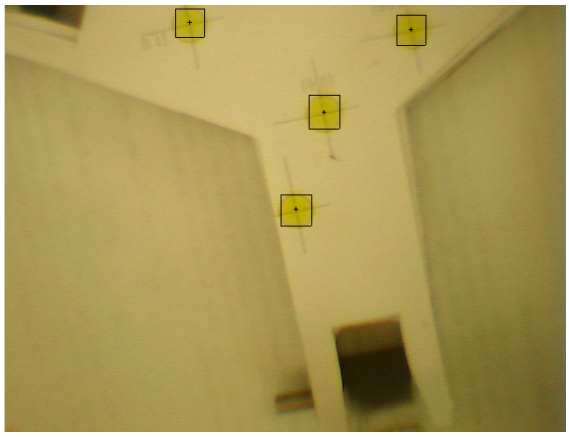


Figure B.4: Image captured with USB camera superimposed to the detected zones

EDGE EXTRACTION

Image processing techniques are intended to provide information of the real world based on a image. There are several approaches on how to recognize objects or shapes from an available image, however most of this techniques relies on the edges of the image.[166;171–173]

Before any pattern can be identified it is necessary to extract the edges of the image. Image edges are defined as pixels where the brightness changes abruptly. The intensity change can be expressed mathematically using the gradient of the pixel, which is composed of two parts, *magnitude* and *direction* as shown in Equations (B.2) and (B.3). Compute these quantities generally is expensive and the result is highly sensitive to noise. To address these issues many researchers such as Canny[165], Roberts, Sobel, Prewitt[135], Kirsch[174] among others have developed methods to estimate the gradient.

$$|Grad(x,y)| = \sqrt{\frac{\partial g}{\partial x}^2 + \frac{\partial g}{\partial y}^2} \tag{B.2}$$

$$\phi = Tan^{-1}(\frac{\frac{\partial g}{\partial y}}{\frac{\partial g}{\partial x}}) \tag{B.3}$$

Roberts Operator was among the first operators used due to its very low computational cost. This method uses a $2x2$ kernel as shown in Equation (B.4), since the dimension of the kernel and as a consequence the neighborhood is kept to a minimum the result is highly sensitive to noise. A much more robust method to estimate the magnitude of the gradient is given by the Laplace operator that estimates the second derivative as shown in Equation (B.5).

$$H_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{B.4}$$

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{B.5}$$

The Compass operators approximate the first derivative by performing a convolution between the image with 8 different mask, the one with greater magnitude will be consider as the edge direction. This method is computationally expensive since multiple masks have to be evaluated.

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, H = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} ... \tag{B.6}$$

Canny[165] proposed a new method based on calculus of variations. This approach focuses on three main objectives: No edge should be missed, the edge should be identified as closely as possible to the real location, and spurious response should be kept to a minimum. On the original approach presented by Canny the edge detection was considered to be a 1D problem and calculus of variations was used to solve it, later the solution was extended to 2D problems. The objectives are mathematically written in the form of constrains.

- Detection criteria:

  This criterion establish that every edge should be detected and spurious response should be discarded. This can be expressed mathematically as shown

in equation B.7.

$$SNR = \frac{|\int_{-w}^{w} G(-x)f(x)dx|}{n_0\sqrt{\int_{-w}^{w} f^2(x)dx}} \tag{B.7}$$

- Localization criteria: The location of the edge should be close enough to the real edge. This can be expressed mathematically as shown in equation B.8.

$$Loc = \frac{|\int_{-w}^{w} G'(-x)f'(x)dx|}{n_0\sqrt{\int_{-w}^{w} f'^2(x)dx}} \tag{B.8}$$

- Eliminating multiple response: Each edge should only give one response and not multiple. This can be expressed mathematically as shown in equation B.9.

$$x_{max}(f) = 2x_{zc}(f) = kWwww. \tag{B.9}$$

Using numerical optimization techniques an optimal detector can be synthesized. This method, which will yield optimal edges, can be resumed as follows: first a Gaussian Filter is used to reduce the effects of white noise, then the edge can be found using Equation (B.10). With a proper threshold the sharpest edges can be detected, finally using an hysteresis approach spurious response can be eliminated. In the next subsection the numerical implementation of this edge extraction technique is presented

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * f = 0 \tag{B.10}$$

EDGE DETECTION NUMERICAL IMPLEMENTATION

Since the Canny edge detector is highly sensitive to white noise the first step is to apply a *Gaussian filter* to blur the image, which is a mask that is convoluted with the image. Since the white noise is assumed to affect each pixel with out any correlation it can be reduced by averaging all the pixels in the neighborhood. The filter performs a wighted average in a desired neighborhood with size σ, assuming that the further away the pixel the less the weight should be. Any element laying at a distance greater than 3σ is considered to have no influence at all, hence weight is 0. The kernel will have a size $6\sigma+1 \times 6\sigma+1$ where each element is computed using Equation (B.11), then the matrix is normalized such that the sum of all elements is 1. A $3D$ plot of the filter is shown in Figure B.5, the sample is shown in Figure B.6, and the blured image in Figure B.7.

$$G(x,y) = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{B.11}$$

From[165] we know that the edges of the image will be located at the local maximum of the image convolved with the operator G_n, where this operator is nothing but the derivative of the filter G in the direction $\mathbf{n}$. To approximate this maximum Equation (B.12) can be used.

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * f = 0 \tag{B.12}$$

The normal to the edge is considered to be the edge direction. To numerically

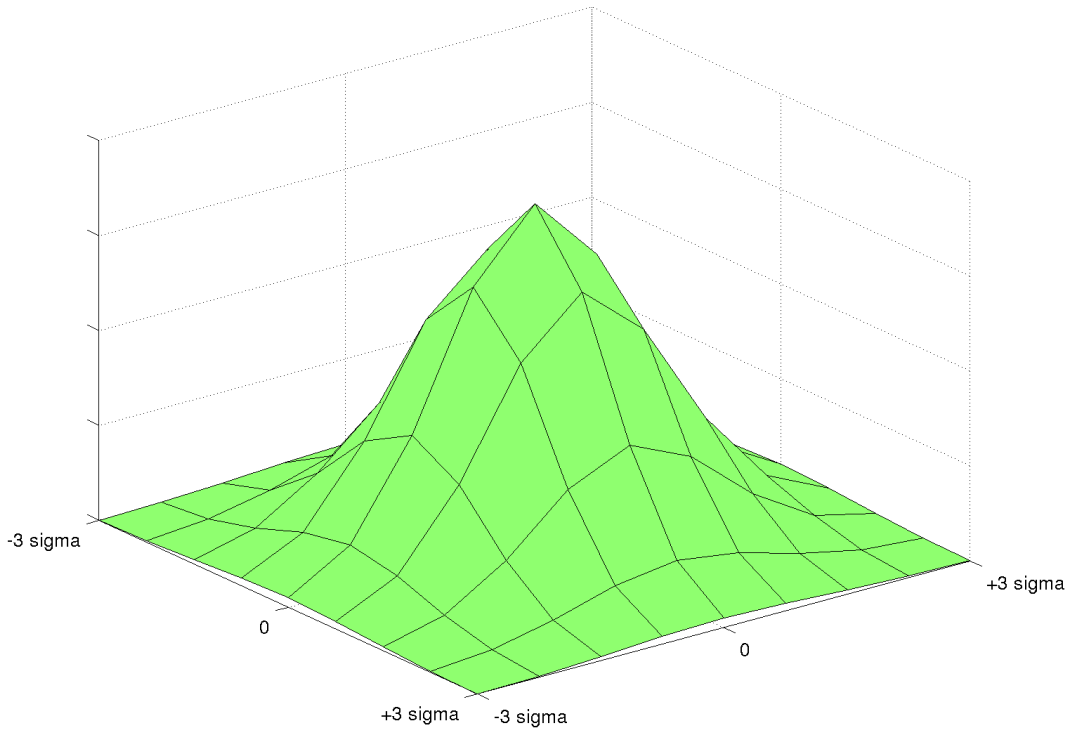


Figure B.5: Shape of the Gaussian filter

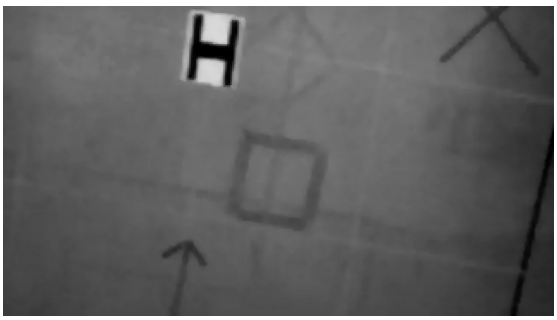


Figure B.6: Image in gray scale

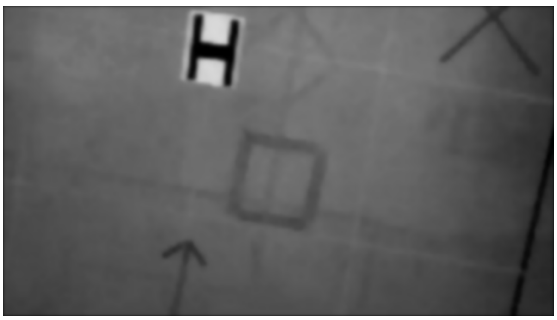


Figure B.7: Filtered image

Figure B.8: edge in X direction

Figure B.9: Edge in Y direction

estimate the directional derivative first we need to estimate the edge direction $\boldsymbol{n}$ using Equation (B.13). Where f is the image and G is the filter computed as described above. The discrete gradient can be easily computed as $\nabla r(i,j) = [r(i+1,j) - r(i,j); r(i,j+1) - r(i,j)]$. The results can be seen in Figures B.8 to B.10.

$$\mathbf{n} = \frac{\nabla(G * f)}{|\nabla(G * f)|} \tag{B.13}$$

The edge direction is used to estimate the second derivative. First the gradient is computed, then the dot product with the normal is computed. The dot product yields the first directional derivative. The second derivative is found by repeating this procedure twice. If a threshold (i.e only consider the pixels with certain magnitude as part of the edges) is used then a first set of edges can be found Figure B.10. As can be seen in the image the detection not only gives the edges but also some spurious response. To reduce the spurious response an hysteresis algorithm can be

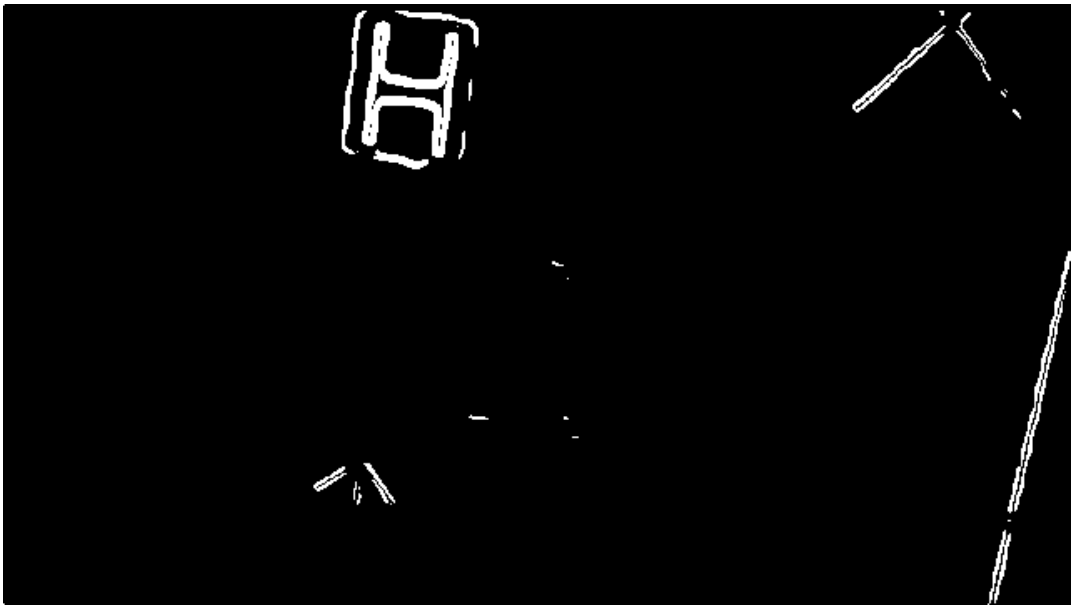


Figure B.10: Detected edge

used.

The hysteresis algorithm is a iterative process. The first step is to select a threshold value t_1 and any edge greater than that can be considered as edge. For the edges laying in the interval $[t_0 : t_1]$ mark only as real edges the ones that are next to a edge already mark as valid. This operation should be repeated until no change take place. In Figure B.11 is shown the final edge identified using edge with hysteresis. It can be seen from the image that the edge has a width of several pixels, however this is not desired. Using dilation and erosion the width of the image edge can be reduced to only one pixel width as shown in Figure B.12.

Figure B.11: Thresholding with hysteresis

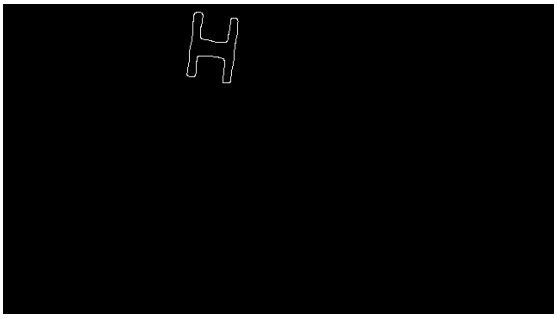


Figure B.12: edge detection result

EDGE BASED PATTERN MATCH

Once the edges have been detected we need to extract the shape of the objects on the image. Hough[166] developed a method that can be used to identify simple shapes such as lines and circles. One of the key features of this method is that can identify partially covered shapes. The basic principle is to parametrize the shape in terms of its main characteristics, for instance, a circle will be parametrized in terms of its radius. A family of shapes is generated varying each parameter in a predefined set of limits. In our example a family of circles with radius $r \in [r_m; r_M]$ will be generated as tests patterns. These patterns are compared against the found edges. Each pixel identified as edge is used as a center for the circle. A weight with initial value of 0 is assigned to each pixel. For every pixel that lays on the contour of the circle its corresponding weight gets its value increased by one. Repeating this procedure for all the test object over the entire identified edge will generate a matrix of weights. The highest weight is considered to be the position of the identified object. The outcome of a test image with several circular patterns is

shown in Figure B.13.

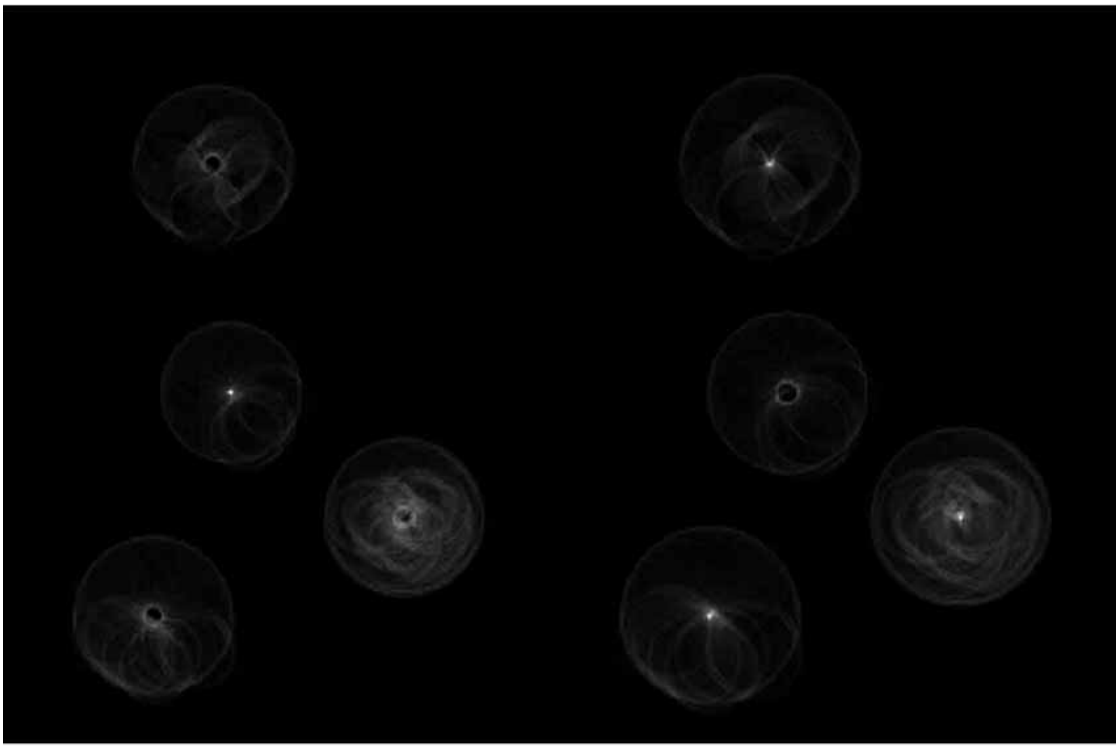


Figure B.13: Circular patterns identified with the Hough transform

Appendix C

HOMOGRAPHY

This section presents the method used to estimate the position and orientation of an object using reference features extracted from an image. The camera can be modeled as a pinhole[175](Figure C.1) where the relationship between the augmented 3D point denoted by $[\alpha_{1_i}, \alpha_{2_i}, \alpha_{3_i}, 1]^T$ and its image projection denoted by $\tilde{\boldsymbol{m}}_i = [u_i, v_i, 1]^T$ is given by

$$\lambda[u_i, v_i, 1]^T = A[R|\boldsymbol{t}][\alpha_{1_i}, \alpha_{2_i}, \alpha_{3_i}, 1]^T \tag{C.1}$$

where λ is a scale factor, R and $\boldsymbol{t}$ are the extrinsic parameters, namely rotation matrix and translation vector. The camera intrinsic matrix A is given by

$$A = \begin{bmatrix} \delta & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.2}$$

where (u_0, v_0) are the principal point coordinates, δ and β are the scaling factors

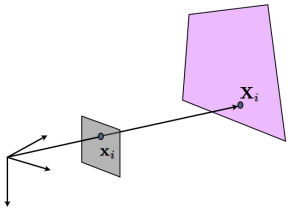


Figure C.1: Pin hole camera model

for the image, and γ the skewness of the axis. These parameters can be determined following a standard calibration procedure as the one presented by Zhang.[176]

Without loss in generality, all points can be chosen in the plane $\alpha_{3_i} = 0$ in the object space. Then, we can reduce the projection equations to

$$\begin{aligned} \lambda \tilde{\boldsymbol{m}}_i &= A[\boldsymbol{r_1}|\boldsymbol{r_2}|\boldsymbol{r_3}|\boldsymbol{t}] \begin{bmatrix} \alpha_{1_i} \\ \alpha_{2_i} \\ 0 \end{bmatrix} \\ \lambda \tilde{\boldsymbol{m}}_i &= A[\boldsymbol{r_1}|\boldsymbol{r_2}|\boldsymbol{t}] \begin{bmatrix} \alpha_{1_i} \\ \alpha_{2_i} \end{bmatrix} \\ \lambda \tilde{\boldsymbol{m}}_i &= H\boldsymbol{M}_i \end{aligned} \tag{C.3}$$

where $\boldsymbol{M}_i = [\alpha_{1_i}, \alpha_{2_i}, 1]^T$ and $H = A[\boldsymbol{r_1}|\boldsymbol{r_2}|\boldsymbol{t}]$. Knowing $\boldsymbol{M}$ and measuring $\hat{\boldsymbol{m}}$ we can proceed to estimate the homography between the plane and the image, namely extract the rotation matrix and translation vector. This problem can be presented as a minimization problem:

$$\sum_{i=0}^{n} ||\hat{\boldsymbol{m}}_i - \tilde{\boldsymbol{m}}_i||^2 \tag{C.4}$$

where $\hat{\boldsymbol{m}}_i = \frac{1}{\boldsymbol{h_3 M}_i}[\boldsymbol{h_1 M}_i, \boldsymbol{h_2 M}_i]^T$ and $\boldsymbol{h_i}$ is the i^{th} row of H. Equipped with the first two columns of the rotation matrix, the third column can be determined using the cross product.

Appendix D

KALMAN FILTER

When a physical system is completely known and correctly modeled, knowing the system inputs it is possible to estimate the system response along with its internal states. However, perfect system knowledge is never the case. When the system is modeled and later linearized, errors are incurred. Further, measurements are always accompanied by noise. To overcome these limitations, different kind of filters and states estimators are commonly used. The estimators not only reduce the noise error in the quantities measured but also provide estimates for the quantities that are not directly measured. In this section a popular filtering technique is presented. The quantities used throughout the remainder of this section are defined. All quantities in ***bold*** font represent vectorial quantities

measured value	=	real value	+	measurement error
$\tilde{\boldsymbol{x}}$	=	$\boldsymbol{x}$	+	$\boldsymbol{\nu}$

estimation error	=	estimated value	-	measured value
$\boldsymbol{e}$	=	$\hat{\boldsymbol{x}}$	-	$\tilde{\boldsymbol{x}}$

We start by assuming that the plant is completely know and that the time

response to the inputs is linear. Following this assumptions, the EOM of the system are given by Equation (D.1) where F represents the state matrix, B represents the control matrix, and H is the measurement matrix.

$$\begin{cases} \dot{\boldsymbol{x}} = F\boldsymbol{x} + B\boldsymbol{u} \\ \boldsymbol{y} = H\boldsymbol{x} \end{cases} \tag{D.1}$$

The previous equation assumes that there are no measurement errors, but as described before, this is not true. Usually the measurement error can be considered to be independent from the plant dynamics. The data acquisition equation is given by $\tilde{\boldsymbol{y}} = H\boldsymbol{x} + \boldsymbol{\nu}$, where $\boldsymbol{\nu}$ represents the measurement error.

A popular state estimator technique is to use the mathematical model of the system to propagate the inputs to the outputs. At each time-step, the error between the estimation and the measurements are fed to the estimator with a gain K as shown in Equation (D.2). If the matrix K is properly chosen the error will be reduced in each time-step and the estimate will converge to the real value.

$$\begin{cases} \tilde{\boldsymbol{y}} = H\boldsymbol{x} + \boldsymbol{\nu} \\ \dot{\hat{\boldsymbol{x}}} = F\hat{\boldsymbol{x}} + B\boldsymbol{u} + K[\tilde{\boldsymbol{y}} - H\hat{\boldsymbol{x}}] \\ \hat{\boldsymbol{y}} = H\hat{\boldsymbol{x}} \end{cases} \tag{D.2}$$

We can rearrange the previous equation in terms of the estimate error dynamics $\dot{\tilde{\boldsymbol{x}}}$ such that

$$\dot{\tilde{\boldsymbol{x}}} = (F - KH)\tilde{\boldsymbol{x}} + \boldsymbol{\nu}$$

From the previous equation, observe that the error dynamics are independent of the system inputs $\boldsymbol{u}$. The state estimator is reduced to obtain a feedback gain K. The gain must be large enough, such that the error converges rapidly to zero. However, it must be small enough to assure the filter stability. The gain matrix, is conditioned by the plant dynamics. Several techniques, have been developed to chose the matrix K that best fits the system. Methods such as "pole-placement" [93] or Ackermans[68] method require the designer to propose desired plant dynamisc. A deep knowledge of the plant and vast experience is required. The gains obtained using these methods are ad-hoc and cannot be reused for other problems.

Kalman,[140] proposed a rigorous procedure based on stochastic processes described by the measurement error. This approach requires knowledge of the plant dynamics, measurement equations, and the noise statistical characteristics. This represents a great advance over the other methods as they require the definition of desired dynamics for the estimator. Since this dissertation is heavily implementation oriented, we focus our studies on the discrete time version of the filter. The discrete time state-space model is given by Equation (D.3) where $\boldsymbol{v}_k$ y $\boldsymbol{w}_k$ are assumed to be normally distributed with zero mean and known standard deviation without correlation.

$$\begin{cases} \boldsymbol{x}_{k+1} = \Phi \boldsymbol{x}_k + \Gamma_k \boldsymbol{u}_k + \gamma_k \boldsymbol{w}_k \\ \tilde{\boldsymbol{y}}_k = H_k \boldsymbol{x}_k + \boldsymbol{v}_k \end{cases} \tag{D.3}$$

We assume that the state estimator presented early in the chapter is valid for

discrete systems, then we have

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1}^{-} &= \Phi_k \hat{\boldsymbol{x}}_k^{+} + \Gamma_k \boldsymbol{u}_k \\ \hat{\boldsymbol{x}}_k^{+} &= \hat{\boldsymbol{x}}_k^{-} + K_k[\tilde{\boldsymbol{y}}_k - H_k \hat{\boldsymbol{x}}_k^{-}] \end{aligned} \tag{D.4}$$

The error distribution is assumed normal, hence the distribution is completely characterized with the covariance matrix. All covariance matrices are given in Equation (D.5) where $\tilde{\boldsymbol{x}}$ represents the error in the estimate.

$$\begin{aligned} P_k^{-} &\equiv E\{\tilde{\boldsymbol{x}}_k^{-}\tilde{\boldsymbol{x}}_k^{-T}\} \quad P_{k+1}^{-} \equiv E\{\tilde{\boldsymbol{x}}_{k+1}^{-}\tilde{\boldsymbol{x}}_{k+1}^{-T}\} \\ P_k^{+} &\equiv E\{\tilde{\boldsymbol{x}}_k^{+}\tilde{\boldsymbol{x}}_k^{+T}\} \quad P_{k+1}^{+} \equiv E\{\tilde{\boldsymbol{x}}_{k+1}^{+}\tilde{\boldsymbol{x}}_{k+1}^{+T}\} \end{aligned} \tag{D.5}$$

We define the propagation error as $\tilde{\boldsymbol{x}}_{k+1}^{+} \equiv \hat{\boldsymbol{x}}_{k+1}^{+} - \boldsymbol{x}_{k+1}$. Replacing $\tilde{\boldsymbol{x}}_{k+1}^{+}$ in Equation (D.4) it can be shown that

$$\tilde{\boldsymbol{x}}_{k+1}^{-} = \Phi_k \tilde{\boldsymbol{x}}_k^{+} - \gamma \boldsymbol{w}_k$$

Since all quantities are known, we replace $\tilde{\boldsymbol{x}}_{k+1}^{-}$ in the covariance definition to obtain an expression to propagate the covariance matrix.

$$P_{k+1}^{-} \equiv E\{\tilde{\boldsymbol{x}}_{k+1}^{-}\tilde{\boldsymbol{x}}_{k+1}^{-T}\} \tag{D.6}$$

$$P_{k+1}^{-} = E\{\Phi_k \tilde{\boldsymbol{x}}_k^{+}\tilde{\boldsymbol{x}}_k^{+T}\Phi_k^{T}\} - E\{\Phi_k \tilde{\boldsymbol{x}}_k^{+}\boldsymbol{w}_k^{T}\gamma_k^{T}\} - E\{\gamma \boldsymbol{w}_k \tilde{\boldsymbol{x}}_k^{+T}\Phi_k^{T}\} + E\{\gamma \boldsymbol{w}_k \boldsymbol{w}_k^{T}\gamma_k^{T}\} \tag{D.7}$$

Taking into account that there is no correlation between $\boldsymbol{w}_k$ and $\tilde{\boldsymbol{x}}_k^+$ the equation can be simplified. We replace the equivalent quantities from Equation (D.5) which yields

$$\boxed{P_{k+1}^- = \Phi P_k^+ \Phi^T + \Gamma_k Q_k \Gamma_k^T} \tag{D.8}$$

With some algebraic operations it can be shown that

$$\tilde{\boldsymbol{x}}_k^+ = (I - K_k H_k)\tilde{\boldsymbol{x}}_k^- + K_k \boldsymbol{v}_k$$

Where the covariance is given by

$$\begin{aligned}
P_k^+ \equiv & E\{\tilde{\boldsymbol{x}}_k^+ \tilde{\boldsymbol{x}}_k^{+T}\} \\
= & E\{(I - K_k H_k)\tilde{\boldsymbol{x}}_k^- + K_k \boldsymbol{v}_k\} \\
= & E\{(I - K_k H_k)\tilde{\boldsymbol{x}}_k^- \tilde{\boldsymbol{x}}_k^{-T}(I - K_k H_k)^T\} + E\{(I - K_k H_k)\tilde{\boldsymbol{x}}_k^- \boldsymbol{v}_k^T K_k^T\} \\
& + E\{K_k \boldsymbol{v}_k \tilde{\boldsymbol{x}}_k^{-T}(I - K_k H_k)^T\} + E\{K_k \boldsymbol{v}_k \boldsymbol{v}_k^T K_k^T\}
\end{aligned}$$

Since there is no correlation between $\boldsymbol{w}_k$ and $\tilde{\boldsymbol{x}}_k^+$ and we know the equivalencies from Equation (D.5) the covariance expression reduces to

$$P_k^+ = [I - K_k H_k] P_k^- [I - K_k H_k]^T + K_k R_k K_k^T \tag{D.9}$$

The main diagonal of the matrix P_k^+ provides a measurement of the total error in the state estimation. Since the filter objective is to reduce the error in the estima-

tion, we search for a K matrix that minimize the trace of P_k^+. This minimization, implies that the maximum total error is reduced to a minimum. Knowing that the P_k^+ matrix is symmetric the minimization problem reduces to

$$min \quad J(K_k) = Tr(P_k^+)$$

Knowing that R_k and P_k^- are symmetric, and using trace properties we have

$$\frac{\partial J}{\partial K_k} = 0 = -2(I - K_k H_k) P_k^- H_k^T + 2K_k R_k$$

Solving for K we obtain the optimum gain for the filter

$$\boxed{K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1}}$$

The obtained gain can be plugged in Equation (D.5) to obtain an expression for the covariance matrix.

$$\boxed{P_k^+ = [I - K_k H_k] P_k^-}$$

As a summary all equations required to solve the filter are presented in Table D.1

Model	$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}_k \boldsymbol{x}_k + \Gamma_k \boldsymbol{u}_k + \gamma \boldsymbol{w}_k, \;\; \boldsymbol{w}_k \sim N(0, Q_k)$ $\hat{\boldsymbol{y}}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{\nu}_k, \quad \boldsymbol{\nu}_k \sim N(0, R_k)$
Gain	$\boldsymbol{K}_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1}$
Update	$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + K_k[\tilde{\boldsymbol{y}}_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_k^-]$ $P_k^+ = [I - K_k H_k] P_k^-$
Propagation	$\hat{\boldsymbol{x}}_{k+1}^- = \boldsymbol{\Phi}_k \hat{\boldsymbol{x}}_k^+ + \Gamma_k \boldsymbol{u}_k$ $\boldsymbol{P}_{k+1}^- = \boldsymbol{\Phi}_k \boldsymbol{P}_k^+ \boldsymbol{\Phi}_k^T + \boldsymbol{\gamma}_k \boldsymbol{Q}_k \boldsymbol{\gamma}_k^T$

Table D.1: Kalman filter for discrete time systems

ATTITUDE DIRECT MEASUREMENT FILTER

In this section we introduce the filter used for the CWJ on Chapter 5. For this particular application we use the approach presented by Crassidis and Junkins.[69;149] In this application, it is assumed that the attitude is directly measured. In the original work, Crassidis assumes that the attitude is directly measured with a star tracker. In our application however, the direct measurements are achieved using the image processing technique presented in the previous sections. The measurements obtained using the image processing techniques are equivalent to the ones given by the star tracker. The measurements equations are given by Equation (D.10). The model for the CWJ dynamics is given in Section 5.2. The summary of the Kalman filter is given in Table D.2.

$$\tilde{\boldsymbol{y}}_k = \left.\begin{bmatrix} \lambda_1^{-1} A[\boldsymbol{r_1}(\boldsymbol{q})|\boldsymbol{r_2}(\boldsymbol{q})|\boldsymbol{t}]\boldsymbol{M}_1 \\ \vdots \\ \lambda_i^{-1} A[\boldsymbol{r_1}(\boldsymbol{q})|\boldsymbol{r_2}(\boldsymbol{q})|\boldsymbol{t}]\boldsymbol{M}_i \\ \vdots \\ \lambda_n^{-1} A[\boldsymbol{r_1}(\boldsymbol{q})|\boldsymbol{r_2}(\boldsymbol{q})|\boldsymbol{t}]\boldsymbol{M}_n \end{bmatrix}\right|_{tk} + \left.\begin{bmatrix} \nu_1 \\ \vdots \\ \nu_i \\ \vdots \\ \nu_n \end{bmatrix}\right|_{tk} = \boldsymbol{h}_k(\hat{\boldsymbol{x}}) + \boldsymbol{\nu}_k \tag{D.10}$$

Model	$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}_k(\boldsymbol{x}_k) + G(t)\boldsymbol{w}_k, \quad \boldsymbol{w}_k \sim N(0, Q_k)$ $\tilde{\boldsymbol{y}}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{\nu}_k, \quad \boldsymbol{\nu}_k \sim N(0, R_k)$
Gain	$H_k(\hat{\boldsymbol{x}}_k^-) = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\big\vert_{\hat{\boldsymbol{x}}_k^-}$ $\boldsymbol{K}_k = P_k^- H_k^T(\hat{\boldsymbol{x}}_k^-)[H_k(\hat{\boldsymbol{x}}_k^-)P_k^- H_k^t(\hat{\boldsymbol{x}}_k^-) + R_k]^{-1}$
Update	$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + K_k[\tilde{\boldsymbol{y}}_k - \boldsymbol{h}(\hat{\boldsymbol{x}})]$ $P_k^+ = [I - K_k H_k(\hat{\boldsymbol{x}}_k^-)]P_k^-$
Propagation	$\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{\Phi}_k(\hat{\boldsymbol{x}}_k)$ $F_k = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\big\vert_{\hat{\boldsymbol{x}}_k}$ $P_{k+1} = F_k P_k + P_k F_k^T + G_k Q_k G_k$

Table D.2: Kalman filter for nonlinear discrete time systems

BIBLIOGRAPHY

[1] D.S. Bernstein. Feedback control: an invisible thread in the history of technology. *IEEE Control Systems*, 22(2):53–68, 2002.

[2] K.H. Lundberg. The history of analog computing: introduction to the special section. *IEEE Control Systems*, 25(3):22–25, 2005.

[3] H. L. Hazen. Theory of servo-mechanisms. *Journal of the Franklin Institute*, 218(3), 1934.

[4] V. Bush and S. H. Caldwell. A new type of differential analyzer. *Journal of the Franklin Institute*, 240(4):255–326, 1945.

[5] John R. Ragazzini, R.H. Randall, and F.A. Russell. Analysis of problems in dynamics by electronic circuits. *Proceedings of the IRE*, 35(5):444–452, 1947.

[6] George A Philbrick. Designing industrial controllers by analog. *Electronics*, 21(6):108–111, 1948.

[7] Thos. D. Truitt. Hybrid computation... what is it?... who needs it?... In *Proceedings of the April 21-23, 1964, Spring Joint Computer Conference*, AFIPS '64 (Spring), pages 249–269, New York, NY, USA, 1964. ACM.

[8] E.L. Zuch. Where and when to use which data converter: A broad shopping list of monolithic, hybrid, and discrete-component devices is available; the author helps select the most appropriate. *IEEE Spectrum*, 14(6):39–43, 1977.

[9] Richard H. Battin. Some funny things happened on the way to the moon. *Journal of Guidance, Control, and Dynamics*, 25(1):1–7, 2002.

[10] Community. Paparazzi autopilot. `http://wiki.paparazziuav.org/`, 2014. [Online; accessed 04-November-2014].

[11] Community. Apm:plane. `http://plane.ardupilot.com/`, 2014. [Online; accessed 04-November-2014].

[12] HaiYang Chao, YongCan Cao, and YangQuan Chen. Autopilots for small unmanned aerial vehicles: A survey. *International Journal of Control, Automation and Systems*, 8(1):36–44, 2010.

[13] MicroPilot. Mp2128. `http://www.micropilot.com/products-mp2028-autopilots.htm`, 2014. [Online; accessed 30-December-2014].

[14] Rong Zhu, Dong Sun, Zhaoying Zhou, and Dingqu Wang. A linear fusion algorithm for attitude determination using low cost MEMS-based sensors. *Measurement*, 40(3):322?328, 2007.

[15] Yiqi Kang and Mei Yuan. Software design for mini-type ground control station of uav. In *Electronic Measurement Instruments, 2009. ICEMI '09. 9th International Conference on*, pages 4–737–4–740, 2009.

[16] Bryan E Walter, Jared S Knutzon, Adrian V Sannier, and James H Oliver. Virtual uav ground control station. In *AIAA 3rd Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004.

[17] Francesca De Crescenzio, Giovanni Miranda, Franco Persiani, and Tiziano Bombardi. Advanced interface for uav (unmanned aerial vehicle) ground control station. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit*, volume 1, pages 486–494, 2007.

[18] UAV Factory. Portable ground control station, 2014.

[19] UAV Solutions. Tactical ground control station, 2014.

[20] Unmanned System Group. Ground control station, 2014.

[21] Community. Mission planner. `http://planner.ardupilot.com/`, 2014. [Online; accessed 04-November-2014].

[22] Lorenz Meier et al. Qground control. `http://www.qgroundcontrol.org`, 2014. [Online; accessed 04-November-2014].

[23] Luo Jun, Xie Shaorong, Gong Zhenbang, and Rao Jinjun. Subminiature unmanned surveillance aircraft and its ground control station for security. In *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International*, pages 116–119, 2005.

[24] Richard D Glover. Aircraft interrogation and display system: A ground support equipment for digital flight systems. *NASA TM-81370*, 1982.

[25] Dwain A Deets, V Michael DeAngelis, and David P Lux. *Himat flight program: test results and program assessment overview*, volume 86725. National Aeronautics and Space Administration, Scientific and Technical Information Branch, 1986.

[26] Eugene L. Duke. V amp;v of flight and mission-critical software. *IEEE Software*, 6(3):39–45, 1989.

[27] H. Jin Kim and David H. Shim. A flight control system for aerial robots: algorithms and experiments. *Control Engineering Practice*, 11(12):1389–1400, 2003.

[28] Christopher Alexander, S Ishikawa, and M Silverstein. Pattern languages. *Center for Environmental Structure*, 2, 1977.

[29] John Vlissides, R Helm, R Johnson, and E the. Design patterns: Elements of reusable object-oriented software. *Reading: Addison-Wesley*, 49:120, 1995.

[30] Daniel Sebastian Monserrat. Modelos de analisis orientado a objetos aplicados en el dominio aeronautico. Magister en ingenieria de software, Universidad Nacional De la Plata, 2005.

[31] Martin Fowler. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[32] Bjarne Stroustrup. *The C++ programming language*. Pearson Education, 2013.

[33] IETF. Transmission control protocol. RFC 793, The Internet Engineering Task Force, 1981.

[34] IETF. Internet protocol. RFC 791, The Internet Engineering Task Force, 1981.

[35] Lorenz Meier. Micro air vehicle communication protocol. standard, ETH Zurich, 2009.

[36] J. B. Knowles and R. Edwards. Effect of a finite-word-length computer in a sampled-data feedback system. *Proceedings of the Institution of Electrical Engineers*, 112(6):1197–1207, June 1965.

[37] H. Hanselmann. Implementation of digital controllersA survey. *Automatica*, 23(1):7–32, January 1987.

[38] J. E. Bertram. The effect of quantization in sampled-feedback systems. *American Institute of Electrical Engineers, Part II: Applications and Industry, Transactions of the*, 77(4):177–182, September 1958.

[39] Thomas A. Brubaker and William Loendorf. Implementation of digital controllers. *Computers & Electrical Engineering*, 1(3):401–413, December 1973.

[40] K. J. strm. Limitations on control system performance. *European Journal of Control*, 6(1):2–20, 2000.

[41] David F. Delchamps. Stabilizing a linear system with quantized state feedback. *IEEE Transactions on Automatic Control*, 35(8):916–924, August 1990.

[42] R.K. Miller, M.S. Mousa, and A.N. Michel. Quantization and overflow effects in digital implementations of linear dynamic controllers. *IEEE Transactions on Automatic Control*, 33(7):698–704, July 1988.

[43] Paul Moroney, A.S. Willsky, and P. Houpt. The digital implementation of control compensators: The coefficient wordlength issue. *IEEE Transactions on Automatic Control*, 25(4):621–630, August 1980.

[44] Bo Hu and A.N. Michel. Some qualitative properties of multirate digital control systems. *IEEE Transactions on Automatic Control*, 44(4):765–770, April 1999.

[45] A. Marigo A Bicchi. Quantized control systems and discrete nonholonomy. *IEEE Trans. on Automatic Control*, 2001.

[46] J. Slaughter. Quantization errors in digital control systems. *IEEE Transactions on Automatic Control*, 9(1):70–74, January 1964.

[47] Bruno Picasso and A. Bicchi. On the stabilization of linear systems under assigned I/O quantization. *IEEE Transactions on Automatic Control*, 52(10):1994–2000, October 2007.

[48] Martin Fowler. *Digital Signal Processing*. Addison-Wesley, 1987.

[49] Yuanqing Xia, Jingjing Yan, Peng Shi, and Mengyin Fu. Stability analysis of discrete-time systems with quantized feedback and measurements. *IEEE Transactions on Industrial Informatics*, 9(1):313–324, February 2013.

[50] S. Tarbouriech and F. Gouaisbaut. Control design for quantized linear systems with saturations. *IEEE Transactions on Automatic Control*, 57(7):1883–1889, July 2012.

[51] I. del Campo and J.M. Tarela. Consequences of the digitization on the performance of a fuzzy logic controller. *IEEE Transactions on Fuzzy Systems*, 7(1):85–92, February 1999.

[52] M.A. Masrur. Studies on the effect of filtering, digitization, and computation algorithm on the ABC-DQ current transformation in PWM inverter drive system. *IEEE Transactions on Vehicular Technology*, 44(2):356–365, May 1995.

[53] Hideaki Ishii and Tamer Baar. Remote control of LTI systems over networks with state quantization. *Systems & Control Letters*, 54(1):15–31, January 2005.

[54] A. Cepeda and A. Astolfi. Control of a planar system with quantized and saturated Input/Output. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 55(3):932–942, April 2008.

[55] Daniel Liberzon. Hybrid feedback stabilization of systems with quantized signals. *Automatica*, 39(9):1543–1554, September 2003.

[56] R. E. Kalman and J. E. Bertram. A unified approach to the theory of sampling systems. *Journal of the Franklin Institute*, 267(5):405–436, May 1959.

[57] K.S. Rattan. Digitalization of existing continuous control systems. *IEEE Transactions on Automatic Control*, 29(3):282–285, March 1984.

[58] Sang Woo Kim, Brian D. O. Anderson, and Anton G. Madievski. Error bound for transfer function order reduction using freqeuncy weighted balanced truncation. *Systems & Control Letters*, 24(3):183–192, February 1995.

[59] Bo Hu and Anthony N. Michel. Stability analysis of digital feedback control systems with time-varying sampling periods. *Automatica*, 36(6):897–905, June 2000.

[60] B. WIE and P. M. BARBA. Quaternion feedback for spacecraft large angle maneuvers. *Journal of Guidance, Control, and Dynamics*, 8(3):360–365, 1985.

[61] Courtland D Perkins and Robert E Hage. *Airplane performance stability and control*. John Wiley & Sons Inc, 1949.

[62] Community. Flightgear. `http://www.flightgear.org/`, 2014.

[63] Glover Keith McFarlane Duncan. *Robust Controller Design Using Normalized Coprime Factor Plant Descriptions*. Springer, 1990.

[64] Ian Postlethwaite Sigurd Skogestad. *multivariable feedback control analysis and design*. Wiley, second edition, 2005.

[65] Chi-Tsong Chen. *Linear system theory and design*. Oxford University Press, Inc., 1995.

[66] JG Ziegler and NB Nichols. Optimum settings for automatic controllers. *transactions of the ASME*, 64(11):759–765, 1942.

[67] Jie Chen, K.H. Lundberg, D.E. Davison, and D.S. Bernstein. The final value theorem revisited - infinite limits and irrational functions. *IEEE Control Systems*, 27(3):97–99, June 2007.

[68] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall, 5 edition, 2009.

[69] J.L. Crassidis and J.L. Junkins. *Optimal Estimation of Dynamic Systems*. Applied mathematics and nonlinear science series. CRC Press, Boca Raton, FL, 2nd edition, 2011.

[70] Robert C Nelson. *Flight stability and automatic control*, volume 2. WCB/McGraw Hill, 1998.

[71] Yew Chai Paw and Gary J Balas. Uncertainty modeling, analysis and robust flight control design for a small uav system. In *AIAA Guidance, Navigation, and Control Conference*, 2008.

[72] anonymous. *Funster V2 ARF Instructions Manual*. Hobby Lobby International, Inc.

[73] IH Abbott, AE von Doenhoff, and L Stivers. Naca report no. 824–summary of airfoil data. *National Advisory Committee for Aeronautics*, 1945.

[74] RD Fink and DE Hoak. Usaf stability and control datcom. *Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio*, 1975.

[75] Mujahid Abdulrahim and Rick Lind. Control and simulation of a multi-role morphing micro air vehicle. *AIAA Paper 2005*, 6481, 2005.

[76] Knife Edge Software. Realflight. `http://www.realflight.com/`, 1997-2014.

[77] Monal Pankaj Merchant. Propeller performance measurement for low reynolds number unmanned aerial vehicle applications, 2005.

[78] Kailash Kotwani, SK Sane, Hemendra Arya, and K Sudhakar. Experimental characterization of propulsion system for mini aerial vehicle. In *31st National Conference on FMFP*, pages 16–18, 2004.

[79] Duane T McRuer, Dunstan Graham, and Irving Ashkenas. *Aircraft dynamics and automatic control*. Princeton University Press, 1972.

[80] Yuandong Ji and H.J. Chizeck. Controllability, stabilizability, and continuous-time markovian jump linear quadratic control. *IEEE Transactions on Automatic Control*, 35(7):777–788, 1990.

[81] CH Woodling. *Apollo experience report: simulation of manned space flight for crew training*. National Aeronautics and Space Administration, 1973.

[82] Frederick Kuhl, Judith Dahmann, and Richard Weatherly. *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR Upper Saddle River, 2000.

[83] Herbert H. Bell and Wayne L. Waag. Evaluating the effectiveness of flight simulators for training combat skills: A review. *The International Journal of Aviation Psychology*, 8(3):223–242, 1998.

[84] Nicolas A Pouliot, Cl, ment M Gosselin, and Meyer A Nahon. Motion simulation capabilities of three-degree-of-freedom flight simulators. *Journal of Aircraft*, 35(1):9–17, 1998.

[85] Robert S. Kennedy, Jennifer E. Fowlkes, and Michael G. Lilienthal. Postural and performance changes following exposures to flight simulators. *Aviation, Space, and Environmental Medicine*, 64(10):912–920, 1993.

[86] RS Kennedy, MG Lilienthal, KS Berbaum, DR Baltzley, and ME McCauley. Simulator sickness in u.s. navy flight simulators. *Aviation, space, and environmental medicine*, 60(1):10–16, 1989-01.

[87] Roy A Taylor. A space debris simulation facility for spacecraft materials evaluation. *SAMPE Quarterly*, 1987.

[88] Daniel Sebastian Monserrat. MODELOS DE ANALISIS ORIENTADO a OBJETOS APLICADOS EN EL DOMINIO AERONAUTICO. Magister en ingenieria de software, Universidad Nacional De la Plata, 2005.

[89] Jack A Adams. *Some considerations in the design and use of dynamic flight simulators*. Operator Laboratory, Air Force Personnel and Training Research Center, Air Research and Development Command, 1957.

[90] IEEE. Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. Standard 1588, Institute of Electrical and Electronics Engineers, 2008.

[91] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer, 2006.

[92] Björn Karlsson. *Beyond the C++ standard library: an introduction to boost*. Pearson Education, 2005.

[93] Richard C Dorf and Robert H Bishop. *Modern control systems*. Pearson, 2011.

[94] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[95] HH Rosenbrock. Some general implicit processes for the numerical solution of differential equations. *The Computer Journal*, 5(4):329–330, 1963.

[96] John C Butcher. Implicit runge-kutta processes. *Mathematics of Computation*, 18(85):50–64, 1964.

[97] Emine Misirli and Yusuf Gurefe. Multiplicative adams bashforth–moulton methods. *Numerical Algorithms*, 57(4):425–439, 2011.

[98] Jeff R Cash and Alan H Karp. A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software (TOMS)*, 16(3):201–222, 1990.

[99] Karsten Ahnert and Mario Mulansky. Odeint online documentation center. `http://headmyshoulder.github.io/odeint-v2/doc/index.html`, 2014. [Online; accessed 24-December-2014].

[100] Karsten Ahnert and Mario Mulansky. Odeint - solving ordinary differential equations in c++. *arXiv:1110.3397 [physics]*, pages 1586–1589, 2011. IP Conf. Proc. - September 14, 2011 - Volume 1389, pp. 1586-1589.

[101] Alan S. Willsky. A survey of design methods for failure detection in dynamic systems. *Automatica*, 12(6):601–611, 1976-11.

[102] R.H. Walden. Analog-to-digital converter survey and analysis. *IEEE Journal on Selected Areas in Communications*, 17(4):539–550, 1999-04.

[103] ME Van Valkenburg. *Analog filter design*. Holt, Rinehart, and Winston, 1982.

[104] W Flenniken, J Wall, and D Bevly. Characterization of various imu error sources and the effect on navigation performance. In *Proceedings of the Institute of Navigation GNSS conference*, 2005.

[105] B. Barshan and H.F. Durrant-Whyte. Inertial navigation systems for mobile robots. *Robotics and Automation, IEEE Transactions on*, 11(3):328–342, 1995.

[106] John L Junkins and James D Turner. *Optimal spacecraft rotational maneuvers*. Elsevier, 1986.

[107] Connie Kay Carrington and JL Junkins. Optimal nonlinear feedback control for spacecraft attitude maneuvers. *Journal of Guidance, Control, and Dynamics*, 9(1):99–107, 1986.

[108] T Dwyer III. The control of angular momentum for asymmetric rigid bodies. *Automatic Control, IEEE Transactions on*, 27(3):686–688, 1982.

[109] SR Vadali and JL Junkins. Optimal open-loop and stable feedback control of rigid spacecraft attitude maneuvers. *Journal of the Astronautical Sciences(ISSN 0021-9142)*, 32:105–122, 1984.

[110] SR Vadali, LG Kraige, and JL Junkins. New results on the optimal spacecraft attitude maneuver problem. *Journal of Guidance, Control, and Dynamics*, 7(3):378–380, 1984.

[111] Panagiotis Tsiotras. Stabilization and optimality results for the attitude control problem. *Journal of Guidance, Control, and Dynamics*, 19(4):772–779, 1996.

[112] Bong Wie and Peter M Barba. Quaternion feedback for spacecraft large angle maneuvers. *Journal of Guidance, Control, and Dynamics*, 8(3):360–365, 1985.

[113] Peter E Crouch. Spacecraft attitude control and stabilization: Applications of geometric control theory to rigid body models. *Automatic Control, IEEE Transactions on*, 29(4):321–331, 1984.

[114] Chih-Jian Wan and Dennis S Bernstein. Nonlinear feedback control with global stabilization. *Dynamics and Control*, 5(4):321–346, 1995.

[115] Peter C Hughes. *Spacecraft attitude dynamics*. Courier Dover Publications, 2012.

[116] Bong Wie. *Space vehicle dynamics and control*. Aiaa, 1998.

[117] Hanspeter Schaub and John L Junkins. *Analytical mechanics of space systems*. Aiaa, 2003.

[118] Nazareth Sarkis Bedrossian. Steering law design for redundant single gimbal control moment gyro systems. *NASA STI/Recon Technical Report N*, 87:28882, 1987.

[119] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of dynamic systems, measurement, and control*, 108(3):163–171, 1986.

[120] Shriram Krishnan and Srinivas R Vadali. An inverse-free technique for attitude control of spacecraft using cmgs. *Acta Astronautica*, 39(6):431–438, 1996.

[121] Bong Wie. Singularity escape/avoidance steering logic for control moment gyro systems. *Journal of Guidance, Control, and Dynamics*, 28(5):948–956, 2005.

[122] Nazareth S Bedrossian, Joseph Paradiso, Edward V Bergmann, and Derek Rowell. Redundant single gimbal control moment gyroscope singularity analysis. *Journal of Guidance, Control, and Dynamics*, 13(6):1096–1101, 1990.

[123] SR Vadali. Feedback control and steering laws for spacecraft using single gimbal control moment gyros. *J. Astronaut. Sci*, 39(2):183–203, 1991.

[124] BR Hoelscher and SR Vadali. Optimal open-loop and feedback-control using single gimbal control moment gyroscopes. *Journal of the Astronautical Sciences*, 42(2):189–206, 1994.

[125] SR Vadali, SR Walker, and H-S Oh. Preferred gimbal angles for single gimbal control moment gyros. *Journal of Guidance, Control, and Dynamics*, 13(6):1090–1095, 1990.

[126] Hanspeter Schaub, Srinivas R Vadali, John L Junkins, et al. Feedback control law for variable speed control moment gyros. *Journal of the Astronautical Sciences*, 46(3):307–328, 1998.

[127] Stephen Lee Canfield. *Development of the Carpal Wrist; a Symmetric, Parallel-Architecture Robotic Wrist.* PhD thesis, Virginia Polytechnic Institute and State University, 1997.

[128] K. H. Hunt. Structural kinematics of in-parallel-actuated robot-arms. *Journal of Mechanical Design*, 105(4):705–712, 1983-12-01.

[129] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation.* CRC press, 1994.

[130] Jing Wang and Clement M. Gosselin. Kinematic analysis and design of kinematically redundant parallel mechanisms. *Journal of Mechanical Design*, 126(1):109–118, 2004-03-11.

[131] Bhaskar Dasgupta and T. S. Mruthyunjaya. Force redundancy in parallel manipulators: theoretical and practical issues. *Mechanism and Machine Theory*, 33(6):727–742, 1998.

[132] Jean-Pierre Merlet. Singular configurations of parallel manipulators and

grassmann geometry. *The International Journal of Robotics Research*, 8(5):45–56, 1989-10-01.

[133] C. Gosselin and J. Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE Transactions on Robotics and Automation*, 6(3):281–290, 1990-06.

[134] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2011.

[135] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Nelson Education Limited, 2008.

[136] R. Adams and L. Bischof. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641 –647, jun 1994.

[137] G.H. Joblove and D. Greenberg. Color spaces for computer graphics. *SIGGRAPH Comput. Graph.*, 12(3):20–25, August 1978.

[138] H. Bay, et al. Surf: Speeded up robust features. *Computer Vision and Image Understanding*, pages 346–359, 2008.

[139] David Lowe. Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision*, 20(2):91–110, 2004.

[140] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[141] V. Kiridena and P. M. Ferreira. Mapping the effects of positioning errors on the volumetric accuracy of five-axis CNC machine tools. *International Journal of Machine Tools and Manufacture*, 33(3):417–437, 1993-06.

[142] Leila Notash and Ron P. Podhorodeski. Forward displacement analysis and uncertainty configurations of parallel manipulators with a redundant branch. *Journal of Robotic Systems*, 13(9):587–601, 1996-09-01.

[143] S. S. Rao and L. Berke. Analysis of uncertain structural systems using interval analysis. *AIAA Journal*, 35(4):727–735, 1997.

[144] Mikhael Tannous, Stphane Caro, and Alexandre Goldsztejn. Sensitivity analysis of parallel manipulators using an interval linearization method. *Mechanism and Machine Theory*, 71:93–114, 2014-01.

[145] Weidong Wu and S. S. Rao. Uncertainty analysis and allocation of joint tolerances in robot manipulators based on interval analysis. *Reliability Engineering & System Safety*, 92(1):54–64, 2007-01.

[146] Xianwen Kong and Clment M. Gosselin. Uncertainty singularity analysis of parallel manipulators based on the instability analysis of structures. *The International Journal of Robotics Research*, 20(11):847–856, 2001-11-01.

[147] Jean-Pierre Merlet. Redundant parallel manipulators. *Laboratory Robotics and Automation*, 8(1):17–24, 1996-01-01.

[148] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[149] S. G. Kim, J. L. Crassidis, Y. Cheng, A. M. Fosbury, and J. L. Junkins. Kalman filtering for relative spacecraft attitude and position estimation. *Journal of Guidance Control and Dynamics*, 30(1):133–143, Jan-Feb 2007.

[150] Xiaocong Zhu, Guoliang Tao, Bin Yao, and Jian Cao. Adaptive robust posture control of parallel manipulator driven by pneumatic muscles with redundancy. *IEEE/ASME Transactions on Mechatronics*, 13(4):441–450, 2008-08.

[151] Kenneth J. Waldron and Kenneth H. Hunt. Series-parallel dualities in actively coordinated mechanisms. *The International Journal of Robotics Research*, 10(5):473–480, 1991-10-01.

[152] Kwun-Lon Ting, Jianmin Zhu, and Derek Watkins. The effects of joint clearance on position and orientation deviation of linkages and manipulators. *Mechanism and Machine Theory*, 35(3):391–401, 2000-03.

[153] Xianwen Kong and Clment M. Gosselin. Kinematics and singularity analysis of a novel type of 3-CRR 3-DOF translational parallel manipulator. *The International Journal of Robotics Research*, 21(9):791–798, 2002-09-01.

[154] S. S. Rao and P. K. Bhatti. Probabilistic approach to manipulator kinematics and dynamics. *Reliability Engineering & System Safety*, 72(1):47–58, 2001-04.

[155] Leila Notash. Uncertainty configurations of parallel manipulators. *Mechanism and Machine Theory*, 33(1):123–138, 1998-01.

[156] Kang Luo and Xiaoping Du. Probabilistic mechanism analysis with bounded

random dimension variables. *Mechanism and Machine Theory*, 60:112–121, 2013-02.

[157] N. Adurthi, P. Singla, and T. Singh. The conjugate unscented transform: An approach to evaluate multi-dimensional expectation integrals. In *American Control Conference (ACC), 2012*, pages 5556–5561, June 2012.

[158] Hendrik Bode. *Network analysis and feedback amplifier design*. The Bell Telephone Laboratories Series. Bell, 1945.

[159] Nikolas Trawny, Anastasios I. Mourikis, Stergios I. Roumeliotis, Andrew E. Johnson, and James F. Montgomery. Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks. *Journal of Field Robotics*, 24(5):357–378, 2007.

[160] Torsten Merz, Simone Duranti, and Gianpaolo Conte. Autonomous Landing of an Unmanned Helicopter based on Vision and Inertial Sensing. In MarceloH. Jr. Ang and Oussama Khatib, editors, *Experimental Robotics IX*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 343–352. Springer Berlin Heidelberg, 2006.

[161] Omid Shakernia, Yi Ma, T. John Koo, and Shankar Sastry. LANDING AN UNMANNED AIR VEHICLE: VISION BASED MOTION ESTIMATION AND NONLINEAR CONTROL. *Asian Journal of Control*, 1(3):128–145, 1999.

[162] Zhenyu Yu, Kenzo Nonami, Jinok Shin, and Demian Celestino. 3d vision

based landing control of a small scale autonomous helicopter. *International Journal of Advanced Robotic Systems*, 4(1):51–56, 2007.

[163] O.A. Yakimenko, I.I. Kaminer, W. J. Lentz, and P. A. Ghyzel. Unmanned aircraft navigation for shipboard landing using infrared vision. *Aerospace and Electronic Systems, IEEE Transactions on*, 38(4):1181–1200, 2002.

[164] Gano B. Chatterji, Padmanabhan K. Menon, and Banavar Sridhar. Vision-based position and attitude determination for aircraft night landing. *Journal of guidance, control, and dynamics*, 21(1):84–92, 1998.

[165] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679 –698, nov. 1986.

[166] P.V.C. Hough. Method and means for recognizing complex patterns, december 1962. US Patent 3,069,654.

[167] ICAO Annex. to the convention on international civil aviation. *Volume I, Aerodrome Design and Operations*, 14.

[168] IETF. User datagram protocol. RFC 768, The Internet Engineering Task Force, 1980.

[169] Jer-Nan Juang, Minh Phan, Lucas G Horta, and Richard W Longman. Identification of observer/kalman filter markov parameters-theory and experiments. *Journal of Guidance, Control, and Dynamics*, 16(2):320–329, 1993.

[170] Martin Diz, Manoranjan Majji, and Puneet Singla. Uncertainty quantification of the eigensystem realization algorithm using the unscented transform. In *The Jer-Nan Juang Astrodynamics symposium*, volume 147 of *Advances in the astronautical sciences*, pages 461–474, Springfield, VA, 2012. American Astronautical Society.

[171] H.G. Barrow and J.M. Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *Artificial intelligence*, 17(1):75–116, 1981.

[172] K.A. Stevens. The visual interpretation of surface contours. *Artificial Intelligence*, 17(1):47–73, 1981.

[173] Takeo Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17(103):409 – 460, 1981.

[174] Russell A. Kirsch. Computer determination of the constituent structure of biological images. *Computers and Biomedical Research*, 4(3):315 – 328, 1971.

[175] J. Kannala and S.S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(8):1335 –1340, aug. 2006.

[176] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.