UNIVERSITY OF SOUTHAMPTON

Development of Technologies for Low–Cost Oceanographic Unmanned Aeronautical Vehicles

by

Matthew Bennett

A thesis submitted in partial fulfillment for the
degree of Doctor of Engineering

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

April 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Engineering

by Matthew Bennett

Oceanographic research vessels and buoys typically provide high-resolution, short-range measurements at a high sample rate. Satellites provide wide-range, low-resolution measurements at a low sample rate. Therefore a gap exists in oceanographic observation capability for medium-range, high-resolution measurements at a high sample rate. An Unmanned Aeronautical Vehicle (UAV) could bridge this gap.

This research has provided a mission-ready autopilot and ground station for future oceanographic application by the National Oceanography Centre, Southampton. A sea landing is the only available option, which carries a low probability of UAV reuse and therefore requires a low-cost system. This and other application-specific requirements led to the development of all autopilot and ground station software. This research provided novel contributions of pseudo-derivative feedback controllers for flight control, the generation of optimised matrix calculations for high-frequency aircraft state estimates on a low-powered processor and the use of a finite impulse response filter for reduced aliasing of transmitted flight data.

A novel in-flight method for autonomously optimising controller response has been developed and successfully demonstrated in realistic simulation and practical flight tests of a commercial model aircraft. This method does not require an experienced operator or known aircraft dynamics and provides a quantitative measurement of optimality.

Two novel path tracking algorithms have been presented. The first controls the derivative of heading rate to command an achievable trajectory. The second controls the aircraft's closing speed on the path by adjusting bank angle. The latter algorithm achieved a robust tracking performance under simulated high wind conditions and practical flight tests and is suitable for oceanographic UAV operation.

Contents

List of Figures

List of Tables

List of Listings

List of Symbols

ϕ	Bank angle (Earth-referenced)
θ	Elevation angle (Earth-referenced)
ψ	Heading (Earth-referenced)
p	Roll rate (body-referenced)
q	Pitch rate (body-referenced)
r	Yaw rate (body-referenced)
A_x	Longitudinal acceleration (body-referenced)
A_y	Latitudinal acceleration (body-referenced)
A_z	Vertical acceleration (body-referenced)
A_c	Centripetal acceleration (body-referenced)
ω_a	Angular rotation used when calculating A_c
R_a	Radial displacement used when calculating A_c
h	Altitude
P_{static}	Static pressure
P_{dyn}	Dynamic pressure
g	Acceleration due to gravity
V_{IAS}	Indicated airspeed
V_{CAS}	Calibrated airspeed
V_{TAS}	True airspeed
V	Airspeed (taken as V_{TAS})
V_g	Ground speed
ψ_g	Ground heading
ψ_{GPS}	GPS measured heading
ψ_c	Commanded ground heading
$\dot{\psi}$	Heading rate
$\dot{\psi}_c$	Commanded heading rate
ϕ_c	Commanded bank angle
T	Air temperature
T_0	International standard air temperature at sea level

P_0	International standard air pressure at sea level
R_d	Specific gas constant of dry air
x	Extended Kalman filter states
$\hat{x}$	Estimated extended Kalman filter states
u	Control inputs (extended Kalman filter)
w	Process noise sources (extended Kalman filter)
z	Measured states (extended Kalman filter)
v	Measurement noise sources (extended Kalman filter)
P	Error covariance matrix (extended Kalman filter)
P_k	Error covariance matrix at iteration k (extended Kalman filter)
K	Kalman gain matrix
K_k	Kalman gain matrix at iteration k
n_{gyro}	Gyroscope noise
n_b	Gyroscope bias noise
v_0	Corrected A_x measurement noise
v_1	Corrected A_y measurement noise
v_2	ψ_{GPS} measurement noise
b_p	Bias of roll rate gyroscope
b_q	Bias of pitch rate gyroscope
b_r	Bias of yaw rate gyroscope
Δt	Update time period (used in multiple algorithms)
K_h	Complementary filter heading gain
K_p	Complementary filter position gain
K_d	PDF controller zero-order feedback gain
K_i	PDF controller integral gain
Y	FIR filter output
X	FIR filter input
B	FIR filter coefficients
S	FIR input sample storage buffer
κ	FIR filter partial output
ϖ	Rotation quaternion
Q	Aircraft orientation quaternion
P_x	Aircraft position east of the origin
P_y	Aircraft position north of the origin
$P_{x,GPS}$	GPS aircraft position east of the origin
$P_{y,GPS}$	GPS aircraft position north of the origin
$\dot{P}_x$	Ground speed component towards the east
$\dot{P}_y$	Ground speed component towards the north

$\ddot{P}_x$	Complementary filter correction factor for P_x
$\ddot{P}_y$	Complementary filter correction factor for P_y
μ	Ideal response mean (controller tuning)
σ	Ideal response standard deviation (controller tuning)
$\tilde{\mu}$	Mean from measured response (controller tuning)
$\tilde{\sigma}$	Standard deviation from measured response (controller tuning)
y_i	Ideal response at sample index i (controller tuning)
s_0	Previous command value (controller tuning)
s_1	Current command value (controller tuning)
d	Measured response (controller tuning)
d_i	Measured response at sample index i (controller tuning)
m	Normalised measured response (controller tuning)
m_i	Normalised measured response at sample index i (controller tuning)
l	Linearised response (controller tuning)
l_i	Linearised response at sample index i (controller tuning)
$\hat{m}$	Normalised, low-pass filtered measured response (controller tuning)
i	Sample index (controller tuning)
i_{start}	Sample index at start of linear response section (controller tuning)
i_{tran}	Sample index at linear to normal CDF transition (controller tuning)
i_{end}	Sample index at end of normal CDF section (controller tuning)
τ_d	Noise threshold (controller tuning)
τ	Normalised noise threshold (controller tuning)
i_{first}	First sample index of optimality calculation (controller tuning)
i_{last}	Last sample index of optimality calculation (controller tuning)
R	Ratio of K_d to K_i (controller tuning)
λ	Fraction before i_{start} threshold crosses (controller tuning)
ρ	Fraction after i_{end} threshold crosses (controller tuning)
l_a	Sample value at first threshold crossing (controller tuning)
l_c	Sample value at last threshold crossing (controller tuning)
W_S	Source waypoint
W_{Sx}	Source waypoint position east of the origin
W_{Sy}	Source waypoint position north of the origin
W_T	Target waypoint
W_{Tx}	Target waypoint position east of the origin
W_{Ty}	Target waypoint position north of the origin
W_{Tr}	Target waypoint orbit radius
d	Perpendicular distance between aircraft and ideal line (ACT)
d_{max}	Maximum value of d used for d-sensitive ACT equation

ε	Angular error in ground heading (ACT)
k	Heading acceleration, taken as derivative of heading rate (ACT)
k_{turn}	Highest magnitude heading acceleration in ideal ACT path
k_{max}	Aircraft's highest magnitude heading acceleration (ACT)
Δx, Δy	Orthogonal distance components of aircraft path (ACT)
T	Time travelled along ACT path
L	Distance travelled at unit speed when turning 90° using ACT
$\dot{\psi}_{ideal}$	Ideal heading rate (ACT)
$\dot{\psi}_{c-}$	Previously commanded heading rate (ACT)
t_L	Heading rate lag time (ACT)
$\hat{\psi}_g$	Predicted ground heading (ACT)
ω	Constant heading rate used for ACT predicted ground heading
δx, δy	Orthogonal distance components travelled during t_L (ACT)
$\hat{P}_x$	Predicted position east of the origin (ACT)
$\hat{P}_y$	Predicted position north of the origin (ACT)
α	Cross-track distance (CST)
$\dot{\alpha}$	Cross-track speed (CST)
$\dot{\alpha}_c$	Commanded cross-track speed (CST)
β	Along-track distance (CST)
$\dot{\beta}$	Along-track speed (CST)
$\dot{\beta}_c$	Commanded along-track speed (CST)
l_{0-3}	Line constants used for α and β calculation (CST)
κ	Ratio of cross-track distance to commanded cross-track speed (CST)
ψ_n	Ground heading perpendicular and towards ideal line (CST)
χ	Closing speed (CST)
χ_c	Commanded closing speed (CST)
t_w	Time remaining till aircraft reaches W_T (CST)
t_a	Time from W_T when payload trigger is activated (CST)

Acknowledgements

I would like to thank my industrial supervisor Dr. Matt Mowlem for his ceaseless enthusiasm and energy behind this project. I would also like to thank Dr. Tom Kazmierski for agreeing to take over as my academic supervisor after the departure of my previous supervisor early in my EngD.

I am forever grateful to our experienced test pilots, Ted Webb from the Winchester Model Aeroplane Club and Tom Bryer from Aerial Target Systems Ltd., for their superior piloting skills without which no UAV flight tests could have been achieved and who both offered their skills for free. Their intervention saved our aircraft from an unintended autonomous landing on more than one occasion.

My thanks also to the National Oceanography Centre, Southampton as both the industrial sponsor for my EngD and provider of a friendly and knowledgable research environment.

Thank you too, to those regular few who joined me for tea and kept the essential balance to my working day.

This thesis is dedicated to my fiancée Sally, who was my willpower throughout its creation.

Chapter 1

Introduction

An oceanographic research ship is capable of taking short range measurements of its surroundings at high resolution and sample frequency. Satellite data is available that covers a wide range, though typically at a lower resolution and sample frequency. Therefore a gap exists in current oceanographic observation capability for medium range, high resolution and frequency measurements. An airborne platform could bridge this gap. The increased speed and altitude offers a range advantage over ships and buoys. The aircraft can frequently visit the subject of interest and remain in close proximity, maintaining a high resolution and sample frequency.

The high cost and operational limitations of manned aircraft prohibit their use by the majority of scientific institutions. The use of an Unmanned Aeronautical Vehicle (UAV) offers a viable alternative. The lack of a human pilot allows a significantly smaller, lighter and less expensive aircraft to be developed. The former two aspects permit relatively small research vessels to transport and launch such an aircraft out at sea, greatly extending the observable regions. The unmanned nature relaxes the aircraft's flight safety margin, widening the range of operational conditions. Should the aircraft be relatively inexpensive, the potential for disposable UAV's and multiple platforms per research institution becomes available. Ship time may be saved by using the enhanced search capabilities of the UAV to direct the ship to specific areas of interest, or sailing distance reduced if the UAV used as a communications relay to devices in remote locations. Any reduction in ship time would lead to a corresponding reduction in mission cost.

While many operational UAV's exist, a high proportion are limited to military applications and are largely unavailable to civilian research institutions. Commercial

solutions remain prohibitively expensive or not specifically designed for oceanographic application. One of the most popular research UAV's, the Aerosonde [5], currently lacks a ship-based launch capability. The ability to carry custom payloads is essential for many research projects yet is often unsupported, with only pre-installed vision-based packages typically provided.

A variety of papers have been published by the academic community documenting their own research into the design and flight testing of inexpensive UAV's. Few of these projects are intended for use outside their own research departments, and none offer a suitable platform for the oceanographic missions envisaged. However the growing knowledge base and success of many of these projects suggest a custom UAV may be realistically developed by the National Oceanography Centre, Southampton (NOC,S) tailored to their specific mission profile and environment. The NOC,S and its partnership with the University of Southampton provides a research base in oceanography and engineering, offering a combination of both technical skills for UAV development and feedback (during design) from the potential end-users of this platform. The NOC,S is the industrial partner and a sponsor for this Engineering Doctorate research.

1.1 Research Challenges

The aim of the NOC,S project in its entirety is to deliver a low-cost UAV suitable for oceanographic research. The wide scope of this project requires multiple research initiatives focused on different technical aspects to run concurrently towards this aim. This Engineering Doctorate represents one such initiative, with a focus on the development and testing of hardware and software for both an autopilot and a ground station.

The main challenge is to provide a reliable autopilot at low cost. The low-cost aspect is particularly necessary given the operational conditions expected for research cruises. Permission to land on or in close proximity to the ship is unlikely to be given due to safety concerns to the ship and crew, making a landing in the sea inevitable. Though reasonable effort will be made in minimising the damage to the UAV on landing and to retrieve it from the sea, the presence of salt water and potential for sinking or simply being lost amongst the waves significantly reduce the chance of recovery and reuse. As a consequence the UAV must be considered disposable and financial budgets will take this into account.

A number of design challenges must be met as a result of the low-cost requirements. The sensors used for aircraft orientation require the use of a robust state estimation algorithm to compensate for the increased noise, bias and non-linearity of such components. Power usage as well as cost limit the computational resources of the autopilots's embedded microcontroller, which in turn reduces the complexity of the algorithms that can be used by the autopilot to maintain real-time autonomous control.

Operational challenges require the development of novel algorithms. The UAV is expected to accurately follow a pre-planned path under high wind conditions. The need to maintain visual contact with the aircraft during land-based trials necessitates a small path radius and emphasises the need for accurate and robust tracking from an early stage. To reduce uncertainty in the accuracy of airframe simulations, flight data is analysed. The ability to tune the autopilot's control behaviour to the real aircraft during flight allows improvement from estimated values obtained during simulation.

Further challenges relate to the UAV's expected users. Such users typically have a scientific rather than aerospace background. The ground station user interface must take such factors into account, and provide specific support for oceanographic mission profiles. The autopilot must also be capable of basic interaction with custom payloads developed by the scientists.

1.2 Research Objectives

Key research objectives to meet the main challenge of a low-cost autopilot for oceanographic application are as follows:

1. Demonstrate a robust autonomous flight control system.
2. Provide a reliable and accurate path tracking ability.
3. Investigate real-time autonomous tuning of controllers. The elimination of manual tuning offers a more rigorous technique suitable for inexperienced operators.
4. Develop reliable ground station software tailored to the oceanographic UAV and expected operator experience.

5. Demonstrate autopilot interaction with payload. This is an essential requirement for scientific missions.

1.3 Research Contributions

In meeting these objectives this research provides the following novel contributions:

1. Pseudo-derivative feedback controllers used for UAV flight control.

   Implementation is described in Subsections 2.6.1 and 3.3.4. Successful simulation and practical flight tests are presented in Subsection 3.3.4 and Section 3.5.

2. Customisation of extended Kalman and complementary filters for aircraft state estimation using the available sensors and processor (Subsection 3.3.5).

   An application was written to optimise the matrix calculations required to enable the autopilot to calculate them within the limited resources of a low-power, low-cost microcontroller. State estimates were split between a complementary filter and extended Kalman filter to reduce the computational load when compared to a Kalman-only solution.

3. Use of a finite impulse response filter to reduce aliasing of autopilot states transmitted to the ground station.

   The low-bandwidth link between the UAV and ground station results in the ground station receiving aircraft states at a lower rate than they are internally updated by the autopilot. A finite impulse response filter was implemented by the autopilot software (Subsection 3.3.3) to low-pass filter these states prior to their transmission, reducing aliasing in the received data.

4. Real-time autonomous tuning of controllers.

   To provide optimal flight performance the autopilot's controllers must be tuned to the unique characteristics of the aircraft. This is typically performed during aircraft simulation by a manual trial-and-error technique. A novel algorithm is presented in Chapter 4 capable of autonomously tuning the controllers in real-time during practical flight tests (Section 4.8). This technique removes the dependance on the human operator's controller tuning experience and is not dependant on an accurate simulation model. Unlike

manual tuning, a quantitative assessment of controller optimality is generated (Section 4.4). The algorithm's flexibility allows it to tune all required controllers with minimal input from the operator.

5. The development, characterisation and optimisation of two novel path tracking algorithms.

   The first algorithm commands realistic changes in heading rate by controlling its derivative (Section 5.3). The second maintains a closing speed on the commanded path by direct adjustment of bank angle (Section 5.4). Both algorithms dynamically generate realistic flight paths and are shown to satisfactorily command a simulated aircraft under low-wind conditions. The closing speed algorithm was shown to be stable under high-wind conditions and capable of guiding the real aircraft during practical flight tests. This algorithm is considered to meet the path tracking objective.

1.4 Thesis Structure

Following this introduction chapter, Chapters 2 and 3 form the first half of the thesis and document the development and testing of a complete UAV system capable of fully autonomous flight. The second half, Chapters 4 and 5, extends this system with research into higher-level control algorithms.

Chapter 2 provides a review of the literature. The identification of oceanographic applications for UAV's and potential scientific instrumentation clarifies the research context. Existing hardware is reviewed for both airframes and autopilots. Autopilot algorithms in key categories of state estimation and flight control are reviewed, and the state of the art over the broad range of UAV technologies is discussed.

Chapter 3 focuses on achieving the research objectives of robust autonomous flight control and ground station software. Aerodynamic modelling and simulation software identified in the literature review is used to create a custom simulation environment tailored to the aircraft chosen for autopilot testing. This environment is used extensively during the subsequent development and testing of the low-level autopilot control algorithms and ground station software. Following analysis of the autopilot's simulated performance, the results are compared with real flight test data to determine the success of the UAV system.

Chapter 4 presents a novel algorithm for the autonomous tuning of the autopilot's controllers in real-time, meeting the associated research objective. The chapter introduces the theory behind the algorithm and provides simulation results in support of the adopted approach. The method is extended following practical considerations, and concludes with an analysis of flight test data. The algorithm's use in other controller types is suggested as future work.

The remaining path tracking and payload interaction research objectives are met in Chapter 5. Novel tracking algorithms are introduced and analysed, and a comparison of simulated aircraft paths performed. The demonstration of the real UAV's successful path tracking ability allows the chapter to conclude with results from the autonomous triggering of an onboard digital camera payload.

Chapter 6 concludes the work achieved and relates both research and end product back to the original oceanographic application.

Chapter 2

Literature Review

2.1 Oceanographic UAV's

A study commissioned by the National Oceanography Centre as part of a Masters of Engineering course [6] identified a specific need for UAV's in oceanographic research. A similar report prepared for the Integrated Ocean Observing System [7] also recommended the use of UAV's for oceanography. Both reports suggest an airborne platform fills the gap between short-range high-resolution *in situ* measurements and wide-range low-resolution satellite observations. UAV's can also resurvey areas of interest more frequently than available satellite passes, particularly when there is cloud cover. Unmanned operation is perceived as a cost-effective and often more flexible alternative to manned flight.

Parallels may be drawn between the operational advantage of UAV's over manned aircraft claimed in polar research literature and the envisaged oceanographic application, since both may be considered harsh environments where a human aircrew would be at increased risk. Arctic UAV research conducted by Williams [8] conclude they were a cost-effective, long-term solution suitable for operation under the extreme conditions. Earlier manned flights were considered expensive and short-term. Similarly, Antarctic UAV research conducted by van den Kroonenberg et al [9] claim the high expense and risk of using ice-bound ships or manned aircraft during the antarctic winter has left gaps in some scientific observations over the sea-ice zone. UAV's are suggested as a plausible alternative.

The following subsection reviews potential oceanographic applications and sensor payloads as these inform design decisions at all stages of project development. The

next subsection identifies key specifications for an oceanographic UAV following this review. The final subsection examines UAV's already available for similar purposes or under similar environments, and compares them to the specification. This section concludes with the decision to develop a custom UAV airframe.

2.1.1 Applications and Sensors

Both [6] and [7] suggest a number of meso-scale oceanographic applications would be enhanced by the increased observation frequency and resolution offered by UAV's. Some applications relate to marine safety, including the monitoring of hazardous algal blooms, oil spills and the mapping of ice for ship navigation. The latter application would also be of scientific interest. Further scientific applications enhance observation of the localised productivity of phytoplankton, organic heat transport and air-sea interaction.

The airborne nature of the platform provides further opportunities beyond passive observation. Miles et al [10] describe an expendable miniature buoy designed to be dropped from the air by a manned or unmanned aircraft. The buoy is capable of wave height and temperature recording and has been practically tested. Uses for these buoys include tropical cyclone and surface current monitoring [7], wave model research and support of military operations.

A UAV lends itself as a communications relay, particularly relevant in the large distances typically presented at sea. Horner et al [11] suggest UAV's may be used to relay communications between AUV's and the research ship. Finn et al [12] claim UAV's have the capacity for significant horizon extension of radio communications, with a 500 m altitude extending the radius to nearly 100 km. Practical marine military operations are also cited, with the successful handover of UAV control from a ground launch/recovery site to a ship. The jamming of a marine radar was achieved using an electronic warfare package on board the UAV.

A further *in situ* advantage of the UAV over remote satellite observation is gas and spray sampling at the air-sea boundary [13], suggested by the National Oceanography Centre. Practical examples of this technique are rare in literature though a similar concept is suggested by Williams [8]. A UAV is used to collect ice and particle samples from cirrus clouds, with sample imaging performed on board using a specialised video system. Practical demonstration was performed over the Arctic, and the results correlated well with those observed by a ground-based millimetre wavelength cloud radar.

A variety of on-board sensors are required to achieve the oceanographic applications listed. Returning to passive observation, a common payload is a digital camera. A panchromatic camera is suggested by Lomax et al [7] as a method of identifying harmful algal blooms. It is expected a standard video or stills camera would be sufficient for mapping oil spills and sea ice. As previously indicated, such a device is also a component of airborne particle sampling. Imaging in the infra-red spectrum provide sea surface temperature readings [7]. A compact hyper-spectral imaging system for UAV's is presented by Johnson et al [14] with application to shallow water bathymetry [7], mine [14] and phytoplankton [7] detection.

Aside from imaging, metrological sensors such as air temperature and humidity are also of scientific interest. A recent UAV mission to the Antarctic by the British Antarctic Survey and the Technical University of Braunschweig [15] used such sensors to examine the boundary layer, along with a five-holed probe for measuring air flow angles and dynamic pressure around the aircraft [9]. Static (atmospheric) pressure was also recorded both for scientific information and aircraft navigation, as were position fixes from the Global Positioning System (GPS) receiver. Holland et al [5] identify a hot-wire probe as a suitable instrument for measurement of liquid water content of cloud and aerosols by a UAV, and may be applicable to studies of air-sea interaction.

While all the sensors described above have been carried by UAV's, some remain too large for present UAV application. Examples cited by Lomax et al [7] include microwave sensors for remote salinity measurement and high-frequency radar for mapping surface currents by radio-wave backscatter (though an alternative use of UAV-deployed GPS-enabled buoys is suggested). Holland et al [5] claim the technology for smaller scientific instruments is rapidly developing in response to UAV, balloon and spacecraft requirements. Lomax et al [7] also foresee many sensors not currently available for UAV's becoming sufficiently miniaturised in the near future.

This review of oceanographic applications and sensors highlights a clear potential for scientific gain through the use of UAV's by the National Oceanography Centre. It also helps form the design specification for both aircraft and autopilot.

2.1.2 Specification

For maximum value the UAV should be capable of long range and endurance. Since many potential oceanographic applications involve remote observation, a

longer range represents a greater improvement over existing ship-based observation. A long endurance additionally increases the number of measurements the UAV can record during a single flight, leading to a better understanding of dynamic environments and potentially increasing the frequency of visits to sites of interest. Should the UAV be deployed to search for a suitable target for the research vessel, or to act as a radio relay, both range and endurance are key criteria in minimising unproductive ship time. Not only must the airframe and engine meet this criteria, but the power consumption of the on-board avionics and autopilot must be sufficiently low to match this endurance (assuming an alternator is not present).

Attention has been drawn to further physical requirements. A greater payload capacity, in both volume and mass, is desirable to increase the number and variety of sensors (or deployable buoys) available for airborne research. The electrical interface to the payload section should also take into account the wide range of potential sensor configurations.

A ship-based launch capability must be available. Two previous NOC,S Masters theses [6, 16] suggested a bungee catapult launch as the most likely method, with the aircraft being required to support this. Various ship-based aircraft retrieval methods are also explored in the Masters theses, though both highlight the safety concerns of flying the aircraft in close proximity to the ship. Subsequent discussion with NOC,S staff and research vessel operators as well as a technology review have concluded that no ship-based landing must be attempted to avoid potential injury to crew or damage to the ship. It is inevitable therefore that a landing directly into the sea will be required. While reasonable effort will be made in minimising the impact force of the UAV and to retrieve it from the sea, the presence of salt water and potential for sinking or simply being lost amongst the waves greatly increase the risk of significant damage or loss of the aircraft. Since any oceanographic UAV is considered likely to encounter harsh flying conditions during operation, the added risk of this landing method further decreases the life expectancy of the UAV. The conclusion is the UAV should be considered disposable. Such a requirement clearly emphasises the need for a low cost system, even when offset against potential savings in ship running costs. An acceptable cost for a disposable UAV is estimated at 2000 GBP [6] to include both airframe and autopilot. The running costs of a research ship is in excess of 15,000 GBP per day [6]. The low cost aspect will form a major design consideration for both the airframe and autopilot components.

In summary key requirements for an oceanographic UAV are as follows:

1. Long range and endurance to add sufficient value to UAV observations
2. Sufficiently low cost system to be considered disposable
3. Ability to withstand potentially harsh flight conditions at sea
4. Ability to be launched from the ship
5. Available volume, mass and electrical interface of payload sufficient to support a variety of sensors

Existing UAV's will be compared to these requirements in the next subsection.

2.1.3 Existing Systems

A broad review of the literature suggests very few UAV's have been designed specifically for oceanographic purposes. The Aerosonde [17] UAV was originally designed for meteorological applications, though offshore monitoring was a routine part of such missions [5]. Though performing regular sea flights the Aerosonde has no ship-based launch capability, which is a key requirement for this project. A report on the current commercial Aerosonde version, the Mk.3 [5] describes the use of a car-mounted launch system and a belly landing. A catapult system is in active development [5], though no landing suggestion (neither sea nor ship) is cited. Later developments of the Aerosonde include the Mk.3.2 (August 2005) [17] with structural strengthening for (amongst other purposes) catapult launches, and the experimental Mk.4 is undergoing tests at time of writing. While the current launch and recovery methods have been demonstrated autonomously using differential GPS [5, 17] standard procedure requires manual control by a trained operator [5, 12].

The Aerosonde report [5] states the aircraft's reliability is lower than some larger and more sophisticated aircraft due to its comparably low-cost design. It contains virtually no redundant systems, with the aim of keeping construction and operating costs to a minimum. Though the resulting component failures can lead to loss of the aircraft, alternative solutions have been sought to minimise this occurrence. The report claims great care has been taken to maximise the reliability of critical components (presumably in both software and hardware), an extensive pre-flight

testing regime is employed and operator training is maintained at a very high level. Losses are still expected by Aerosonde, with the mean time till loss of a Mk.3 aircraft being 500 flight hours.

Aerosonde's operators sell flight hours and data to third parties rather than the aircraft itself. Though the latter does still occur, the specialised crew required to operate the Aerosonde is seen as a key factor for rental preference. The selling of operations on an hourly basis is cited as highly competitive compared to alternative resources. While specific costs are difficult to obtain, in 2003 NASA was quoted 785 USD per flight-hour [18] for use of an Aerosonde. McGeer et al [19] quote the commercial cost of an earlier edition of the Aerosonde aircraft at 25,000 USD in 1999. Though the design philosophy of the Aerosonde and its scientific application is similar to the specification stated in subsection 2.1.2, it is apparent from the expected flight hours, preference for rental and quoted costs that each Aerosonde is not designed to be wholly disposable at the end of each flight. Additionally, support for ship-based catapult launches is only at an early development phase.

A military UAV specifically designed for ship-based operation is the ScanEagle [20], and its commercial version the SeaScan [21] produced by Insitu. The product sheets show both to be of similar performance. The system provides a field-tested ship-mounted catapult launch system and hook-based landing capability. Due to the commercial nature of these UAV's detailed information is not available in the literature, though news articles (May 2006) highlight successful sea trials of a ScanEagle with the Royal Navy. Insitu provide a set of payload packages, though all are related solely to imaging (visual, and infra-red military-only option). No mention is made of user-configurable payloads and this may be a factor in the apparent absence of these UAV's for academic oceanographic research. While cost would most likely be another factor, official figures are not provided. Ramsey [22] however quotes in a 2004 online article the aircraft cost as 72,000 USD but reaching 420,000 USD for the complete package with launcher and ground support. If such quotes are accurate, this system would not be considered low enough cost. The lack of user-configurable payloads also makes this UAV unsuitable for the range of oceanographic research required.

The military Pioneer UAV is designed for rocket-propelled launch and net-based recovery [23]. The Navy note three key limitations [24] during operational service. The lack of an automated launch, landing or mission execution capability lead to a high accident rate. The UAV communicated solely with the ground station, which in turn was not able to be integrated directly with additional systems and units

that may require it. Finally, the lack of waterproof avionics and radar capability made it useless in bad weather. The Pioneer was not intended for oceanographic research (only visual and infra-red imaging payloads were supported) and the rocket-propelled launch would be unsuitable for NOC,S research ship operation. Waterproof avionics would be necessary for the range of weather conditions the aircraft is expected to operate in. Given the collaborative nature of many research projects the lack of data integration with external applications would also be expected. A military system such as this would be prohibitively expensive and unlikely to be available commercially.

Naval research has provided a tube-launched UAV [25] to be deployed by manned aircraft and helicopters. A fixed-wing UAV with swept back wings (due to size limitations) and electric motor (since explosive material was not permitted) is proposed. Since the UAV would only encounter one flying speed, the aircraft design is optimised to this alone. In addition, payload packages formed the nose cone, which could be quickly and independently swapped between aircraft with no additional avionics configuration. The manufacture of the UAV is said to be achievable using only low-cost commercial off-the-shelf parts. This is mostly due to its single mission disposable nature (it is not designed to be recoverable) and very compact and light weight specification. The very small payload capacity and requirement for a manned aircraft makes this system unsuitable for this project.

A company based at NASA's Ames research centre has designed and test flown a twin hull shaped UAV capable of launch and landing on water as a seaplane [26]. While not specifically designed for oceanographic research (it was a demonstrator for an autonomous cargo transporter concept) its marine nature would be advantageous for such applications. An undercarriage was also fitted for land-based operations. The demonstrator aircraft was scaled for only one hour flight duration, though the final design was intended for an impressive 50 hours. Successful autonomous land and water-based flight tests were performed, though a number of practical difficulties are cited. This NASA project did not progress beyond the first demonstrator stage (though all the stage goals had been achieved). Even if this system were available for purchase, a seaplane is unlikely to operate during the moderate sea states expected for a NOC,S UAV. Since the seaplane could not be scaled greater than the cited demonstrator size for NOC,S use, its endurance is also unlikely to be sufficient.

While vertical take-off and landing (VTOL) techniques appear promising for ship-based oceanographic research [27], the only practical method given low-cost requirements would be in the form of a helicopter. Helicopters however have a significantly lower flight endurance and range compared with a fixed-wing aircraft [28]. In addition payload capacity is typically lower than fixed-wing equivalents. As a result VTOL capability is highly unlikely to meet the NOC,S UAV specifications and will not be pursued. This early design decision was important not least because VTOL requires significantly different hardware and autopilot software [27, 29].

The lack of scientific support from existing commercial UAV's suggested by this review is supported by a recent NASA study [18]. Concern was raised that most UAV manufacturers, even those that previously developed civilian applications, now focus on military business. Increased use of UAV's in US Homeland Security is expected to further divert industry's attention from scientific application.

This review concludes that no suitable UAV system is currently available that matches the NOC,S oceanographic UAV design specification. The particularly low-cost requirement is below the purchase price of all potential systems reviewed. Few systems are believed to be capable of the long range and endurance necessary, nor provide the custom payload support for the range of scientific sensors required.

2.2 Test Aircraft

2.2.1 Specification

Outside of the identification of key UAV requirements, the design and construction of the custom NOC,S airframe does not fall within the scope of this EngD research. However, the practical verification of autopilot and avionic hardware required by this research is to run in parallel with the custom airframe development. The selection of an additional test aircraft is therefore necessary. The test aircraft must have the following specification:

1. Short lead time to minimise delay to autopilot test schedule
2. Inexpensive airframe and easily obtainable spare parts
3. Proven and stable flight characteristics to reduce demand on early-stage autopilot integration

4. Sufficient payload volume and mass to support the extra components required for autonomous flight

Similarly to the oceanographic UAV's section, existing aircraft will be reviewed against this specification. The literature review in the next subsection shows many academic institutions engaged in autopilot research share similar test aircraft requirements. The subsection concludes with the selection of a suitable aircraft.

2.2.2 Existing Systems

Several UAV projects [2, 30–32] have been performed at Brigham Young University in the USA using a light-weight flying wing design, known as a Zagi. This airframe increases crash resilience, as does the soft leading edge by absorbing impact energy [32]. The depth and surface area of the flying wing is claimed to provide adequate payload space for the avionics and antennas. The use of a micro-UAV weighing a total of 850 grams is considered to pose a much lower safety risk to civilian population with fewer special requirements and less operator training [32]. While the ease of operation would be advantageous in the early development stage, the payload weight (between 200 and 850 grams depending on model [2]) and volume provided would be insufficient for some sensor packages (such as an off-the-shelf digital camera) expected for latter stage test flights. Endurance of the electronic motor may be short given maximum battery weight, though up to 35 minutes has been achieved by [32]. The use of elevons and flying wing will not be present on the custom NOC,S airframe, so would be undesirable features on the test aircraft.

An alternative platform selected by several Universities is an almost-ready-to-fly (ARF) Trainer class of fixed-wing model aircraft. Like the Zagi, these aircraft are also commercial off-the-shelf (COTS) designs popular with remote control aircraft enthusiasts. The State University of Buffalo [33], Georgia Institute of Technology [34] and Massachusetts Institute of Technology (MIT) [35] use Trainer 60's as their platform for UAV development, the latter employing a fleet of eight for cooperative control experiments. The 60 designation is the minimum recommended engine size in hundredths of cubic inches [36] (around 9.8 cc). The Trainer 60 has easy handling characteristics, a relatively large payload capacity and readily-available off-the-shelf parts and replacement models [33, 35]. Only minor modifications to the aircraft were needed to suit mission requirements (in contrast to the retrofitting of an electric motor to the Zagi in [32]). More than 1.3 kilograms of payload

space was available on the Trainer 60 [33] and flight durations in excess of 50 minutes were achievable with moderate throttle settings [35]. Buffalo and MIT use 9.1 cc four-stroke engines, and Georgia use an unspecified 9.8 cc engine [34]. Brigham Young University themselves used a Trainer 60 as their initial UAV platform before moving to the Zagi, citing increased resilience to crashes as a motivation, though the loss of payload capacity is noted [37].

Pennsylvania State University [38] fly an alternative Trainer ARF model, the SIG Kadet. A Kadet model is also flown by the Institute for Scientific Research Inc. [39] for their autonomous flight testing. The Kadet is larger than the Trainer 60, and Pennsylvania use a four-stroke 15 cc engine [38]. The aircraft is capable of payloads in excess of 2.2 kilograms, though flight durations are not specified. Due to the increased payload the landing gear was strengthened. The Kadet used for similar research by the India Institute of Technology [40] is deemed particularly suitable for the role due to its "roomy" fuselage. Even so, structural modifications were made to their off-the-shelf model due to payload requirements, including the lengthening of wingspan by 25 cm, the widening of the fuselage by 2 cm and the relocation of the actuators. A relatively low-powered 7.5 cc engine was used to perform several successful test flights. A Kadet with an electric motor is used for sense and avoid testing in [41], though was later replaced by a larger aircraft due to concerns it could not handle the desired payload weight.

A SIG Rascal 110 ARF model has been flown successfully by the University of Colorado, though carbon fibre reinforcements were added to the original airframe to permit a payload capacity twice that of the airframe specifications [42, 43]. The UAV weighed 23 kg and employed a considerably larger engine than similar ARF models, at 32 cc with a petrol fuel two-stroke design. Cornell University fly the same airframe as part of their economical UAV programme [44], though it too underwent considerable modification from the COTS product. A custom lifting tail was produced to shift the centre of gravity and large in-wing flaps added to reduce the stall speed. A smaller 24 cc petrol engine was used, and was able to provide up to 1.5 hour flight durations.

Larger aircraft models are also reviewed. Pennsylvania University run two quarter-scale Piper J3 cub models as UAV platforms [45]. While this model has a short flight duration (15-20 minutes) it includes two pods under the wings for external payloads. One holds an external camera (capable of rotation inside the pod) and the other is deployable in flight. The modular design of the pods is claimed to permit fast production and assembly. The airframe and engine combination is

capable of holding "significant" scientific payload [45]. MIT use a similar platform [46] based on a quarter-scale Monocoupe 90A. The 2.2 kg of internal payload space is coupled with an impressive 2 hours flight duration. Santa Clara University [47] reject the use of Piper and Lancair scale models as a UAV platform due to construction difficulties, with the Piper's undercarriage leading to a high susceptibility to wind while on the ground and the Lancair's high stall speed and poor payload capacity (the latter both due to limitations imposed by an accurate scaled model). An ARF trainer similar to a Trainer 60 was seen to improve all these issues and reduce cost.

Some institutions have chosen to fly a custom aircraft from the outset. Stanford University operate three such airframes, named the DragonFly series [48–51]. The first DragonFly is a heavily modified model aircraft with a 3.6 m wingspan, 2.4 m fuselage and a 4.5 hp, two stroke single cylinder engine [51]. The DragonFly II has a fully custom design with 3 m wingspan and two-cylinder, 8 hp engine. The DragonFly III is of similar dimensions (though a redesigned airframe) but capable of carrying a substantial 11 kg of payload [48]. Both platforms have provided Stanford with a "wonderful" [50] test platform, though further quantitative detail of the airframes (including flight duration) is not provided. Other custom airframes include a small electrically-powered flying-wing design by the National University of Singapore [52], a composite fixed-wing aircraft with a blended wing-fuselage by the North Carolina State University [53, 54] and a pusher-prop airframe similar to the Aerosonde (though with an 2 kW electric motor and undercarriage) by the University of Catania [55], used for volcanic gas sampling. While these papers confirm the success of custom aircraft tailored to specific mission requirements, the design cost (in both time and resources) and difficulty in repair make such an approach outside of the test aircraft specification.

This aircraft review has identified many aircraft used for academic UAV research. The ARF class of model aircraft is a frequently used type for such research and has been shown to have satisfactory performance. The Trainer 60 is selected as the test aircraft most capable of fulfilling the specification. Its COTS and ARF nature permits both rapid purchase, construction and ensures a good supply of replacement parts. Its trainer designation provides the necessary flight stability. The reviewed literature and examination of online information suggests the payload capacity will be sufficient.

This review also identifies the potential need to strengthen the airframe against the increased payload weight though the standard Trainer 60 2-stroke 9.8 cc engine

should provide sufficient thrust.

2.3 Autopilots

This section reviews commercial products with the aim of identifying both the state of the art and whether COTS solutions exist that meet the project's specifications (Subsection 2.1.2). Complete and commercially available autopilots are reviewed in the first subsection. The range of electronic components used to build both commercial and academic autopilots are reviewed in the second subsection. Such a review is necessary to determine the feasibility and optimal construction of an in-house autopilot design. All systems presented in the research literature are reviewed with particular attention to low-cost implications.

While a quantitative performance comparison between autopilot implementations would be particularly useful, such information is rare in the literature. In general commercial documents emphasis autopilot functionality and academic papers discuss their autopilot meeting research aims, though neither provide quantitative benchmarks.

2.3.1 Commercial Solutions

Several commercial autopilot solutions are available off-the-shelf. One of the lower-cost products is the MicroPilot [32, 56] system. Like most autopilot products, the company provide a board containing a microprocessor with pre-programmed software, embedded sensors, GPS and ground station software. The University of South Australia have performed UAV testing using a MicroPilot MP2028 autopilot [57, 58]. They experienced numerous unspecified difficulties achieving autonomous flight, and found controller tuning particularly problematic. Singapore University also performed basic flight tests with the same MicroPilot system and their light-weight flying-wing UAV [52]. While the aircraft flew successfully between two points, it was susceptible to wind disturbance (at times causing it to "turn uncontrollably"). Whether this was due to a poorly-tuned autopilot or the experimental nature of the aircraft is uncertain. Difficulties in selecting the correct gains during practical flights are also cited. Santa Clara University report successful flight tests [47] using a MicroPilot MP2028 autopilot. The UAV achieved 3rd place in an international student UAV competition and has successfully demonstrated autonomous follow-the-leader flights. No direct reference is made to the operating

performance and usability of the autopilot itself. Technical difficulties and integration limits of the MicroPilot prevented fully autonomous flight of a Zagi for the University of Arizona [59]. A small project by the Swedish Defence Research Agency [60] claims their MicroPilot software did not work as promised, and cite this as one of the reasons for the failure of some of their project's aims, though do not elaborate on the area of dissatisfaction. The MP2028 has been recently quoted as 5000 USD.

The Piccolo range of autopilots by Cloud Cap Technology [61] are popular and successful in both academic and commercial UAV's. There are strong links between Cloud Cap Technology and the Aerosonde UAV [5, 61], with the Piccolo autopilots being used on all Aerosonde editions [62]. The achievements of the Aerosonde UAV reflect well upon the abilities of the Piccolo. In contrast to MicroPilot integration difficulties do not appear in the literature. The University of Colorado [63, 64] use a Piccolo Plus as the autopilot in their co-operative UAV flights aimed at improving wireless signal coverage. The ability of the Piccolo to stream input and output aircraft states through its serial port was used to build a distributed avionics package, with the GPS location (provided by the Piccolo) was shared between several subsystems. Pennsylvania University use the Piccolo for multi-UAV experiments [65] and in their Kadet ARF UAV [38]. Bayraktar et al[65] include a thorough explanation of the Piccolo path following equations, and Miller et al[38] report the successful integration of an on-board intelligent waypoint-setting computer via the Piccolo's serial interface. The University of California use the Piccolo ground station to interface with their Piccolo autopilot to permit UAV road tracking [43]. A custom ground station is also developed by the university to allow ground vehicle [66] tracking using the Piccolo autopilot. This literature demonstrates the integration ability of the Piccolo with third-party systems. MIT's Trainer 60 UAV's [35] also use the Piccolo system, described in the paper as "well-designed and user configurable". A Thesis acknowledgement relating to the same project states the Piccolo was "a pleasure to work with, and worked without fail on every occasion" [33]. An online source quotes the Piccolo autopilot at 7500 USD with a further 7500 USD for the ground station.

UAV Flight Systems Inc. [67] also produce a commercial autopilot and ground station solution. The system appears to have similar features to both autopilots considered so far. Surprisingly few references are found to its use in academic literature, particularly given it is available at a lower cost than both MicroPilot and Piccolo [32]. Caltabiano et al [55] cites the company's AP-40 autopilot being

used for a volcanic gas sampling UAV by the University of Catania (Italy) though no further information is provided.

Unav LLC. produce a very small, lightweight though less sophisticated autopilot and ground station software [68]. It is believed other autopilot packages of similar price offer greater functionality, which may explain the apparent absence of this autopilot in the literature.

Due to the commercial nature of the autopilots discussed, a key concern with adopting this route is the lack (or expense) of source code. Such code is required to make any modification to the internal operation of the autopilot. Even though a third-party computer has taken advantage of the Piccolo's serial interface to intercept and modify certain aircraft commands [38, 63], the customisation options remain limited for any closed system. It was this inflexibility that lead Brigham Young University to reject the use of MicroPilot in favour of developing their own software [37], citing the inability to directly modify the flight control software as failing their long-term research needs. Similar concerns about the inflexibility of a commercial system were raised by MIT [35] prior to their adoption of the Piccolo, though the considerable reduction in development time was seen as an acceptable trade-off. While Cloud Cap do offer the Piccolo source code for purchase [61], the additional cost, licensing restrictions and ambiguities due to unfamiliar code make this option unattractive. Merging custom code with official updates may also lead to difficulties.

The cost issue alone could be resolved by use of open-source autopilot software available on SourceForge [69]. Suggested hardware components and autopilot circuit diagrams are also provided. While some hardware and software similarities are noted by Brigham Young University in comparison to their own implementation [32], the SourceForge project is still incomplete. The site author notes the initial target is a model helicopter (rather than fixed-wing aircraft) and as such reference is made to helicopter implementation throughout the code and documentation. Reviewing this source suggests the current system is basic when compared to systems such as the MicroPilot or Piccolo. It is difficult to assess the reliability of such a project without knowledge of the development process, particularly one aimed at hobby enthusiasts rather than commercial or academic applications. De Nardi et al [70] and Doncieux et al [71] both reference the project's helicopter simulator software, though no reference is found of the open-source autopilot in academic or commercial research. It is interesting both sources chose to develop their own helicopter autopilots even though they would have been aware of the

project's own offering. It is also likely portions of the code would require rewriting should the hardware components chosen differ from those recommended by the SourceForge project, reducing the advantage of shortened development time.

The target UAV cost of 2000 GBP (Subsection 2.1.2) must include the airframe as well as autopilot cost. Following this review of commercial autopilots, the majority of systems fail to meet this key low-cost requirement. This is likely due to the presence of proprietary software running on the embedded microprocessors which form the complete autopilot. This factor both increases the selling cost and prevents user customisation of the autopilot algorithms. Thus while the success of commercial autopilots are reported in the literature, such an option is not available for this research.

The proposed solution lays with low-cost hardware (identified in the next subsection) combined with license-free and accessible software. Though open-source software is available on SourceForge, concern over its reliability and suitability lead to its dismissal. The development of in-house autopilot software shall be undertaken to ensure adherence to project specifications in reliability, oceanographic mission support, compatibility with chosen hardware and payload interface, and low unit cost.

2.3.2 Hardware

A review of the commercial and academic autopilot hardware shows the same core sensor package is present on all implementations; 3-axis micro-electro-mechanical systems (MEMS) accelerometers and gyroscopes, static and dynamic pressure sensors and GPS module. The key factor that differentiates between models is the choice of embedded processor. The Piccolo series all use a 32-bit Motorola MPC555 [61], operating at 40 MHz with 26 KB of RAM, floating point unit and a number of on-chip peripherals [72]. A 40 MHz processor is used for the UNAV3400 [68], with a large 512 KB of RAM and manufactured by AMD as the 16-bit AM188ES [73]. MicroPilot do not appear to publicise their choice of processor (even in the product manual [57]) and direct reference is rare in the academic literature. Borys et al [74] cites the MP2028 as using a Motorola 68332 processor. Van der Molen [75] only refers to a 16 MHz, 2 KB RAM Motorola processor, though a parametric search of the Motorola website reveals the 68332 processor as the only match. The relatively old processor is slow compared to competing autopilot hardware with

fewer features - though software remains a key factor in overall autopilot performance. The most recent MicroPilot autopilot is the MP2128, is very similar in functionality to the MP2028 except for an impressive 50 times faster processor [56]. The processor make is again unpublished.

UAV Flight System Inc.'s autopilot has a different architecture [76]. Two unspecified "low-power RISC microprocessors" work together on the board - one as the flight processor (handing low-level aircraft stability) and the other as the mission processor (handling navigation, mission events and ground station communications). While no further details are available, the result is apparently successful and may have advantages over a single processor system. The Institute for Scientific Research (USA) autopilot also adopts a dual processor architecture [39]. A 30 MHz PIC microcontroller "sensor data acquisition unit" samples and digitally filters external sensor inputs, as well as interface with the actuators. This is linked to a higher-power 32-bit 100 MHz Etrax 100 processor "flight processing unit" for the remaining flight control operations. The inclusion of an independent sensor data acquisition unit is said to remove pressure from the flight processor, due to high overheads on sensor sampling and filtration. The interrupts generated by external sensors would otherwise degrade the timing determinism of the flight processor. The two processors are mounted on independent boards and interfaced via a single common connector. The Institute believes this modular design leads to easier, independent upgrades of either processor should improved products become available. They cite the main disadvantage of their flight processor as its lack of hardware floating point unit, though the extra computational overhead was not considered a significant limiting factor [39].

Crossbow Technology Inc. provide a commercial variant of the modular autopilot architecture. The company markets a stand-alone sensors and servo control board, the μNav [77]. This contains the "standard" sensors (with GPS) previously identified, a three-axis magnetometer, actuator interface and three temperature sensors to aid calibration. The on-board 8-bit ATmega128(L) microcontroller runs at 16 MHz with 4 KB RAM [78], though is only employed to interface the board's external serial (RS-232) link with on-board components. Crossbow originally designed this product to interface with their Stargate single board computer (SBC) [79] in mind, though sufficient detail is provided in the μNav manual [77] to allow any system to be interfaced. The Stargate has in fact recently been discontinued, though was a Linux-based and ran a powerful X-Scale 400 MHz processor. Use of such a processor represents a considerable jump in processing power compared to other commercial autopilots. The combined cost of the μNav and Stargate was

less than previous autopilots reviewed, most likely due to the absence of autopilot software.

Crossbow does however host an open-source (under a commercially-restrictive GNU license) autopilot project [80] with source code originally designed to run on the Stargate. Since its discontinuation, the latest code release supports the very similar Gumstix SBC family [81]. Since both SBC's use the X-Scale processor and run Linux, few changes are required. The software appears to be at an early development stage, aside from the sensor fusion component. The combination of μNav sensors and Gumstix or Stargate SBC's was considered for the next generation of UAV platforms developed by the Queensland University of Technology (Australia) [82]. While the lack of hardware floating point unit is seen as a negative aspect of the Gumstix/Stargate, the combined system is considered a clear improvement on their current PC/104 implementation. No reference is made to the open-source Crossbow autopilot, and the system is yet to be fully assembled. No further reference to the μNAV or Stargate is found in the literature. The use of Gumstix SBC's for UAV's are cited in [41, 63, 70]. [70] is a small helicopter design, where the Gumstix's on-board Bluetooth was used as a (temporary) solution to wireless communication. The success of the Gumstix SBC in this lightweight application is promising, though its power requirement is less strict due to the short duration (<15 minutes) flights required. The Gumstix SBC in [63] is used as a processing node where high computational power was required. The other nodes in the distributed avionics system presented are simple 8-bit microcontrollers, handling a specific subset of the aircraft's avionic requirements. A cluster of four Gumstix SBC's are used in [41], combined with a MicroPilot MP2128 autopilot onboard a large petrol-powered model helicopter. This substantial processing power is intended to study sense-and-avoid techniques, though the paper concludes at the early airframe evaluation and testing stage.

O-Navi LLC. differ from previous commercial solutions by providing an autopilot-capable board [83] with no software. The Phoenix board itself contains the "standard" sensor set (with integrated GPS but no magnetometer) and runs a 32-bit Motorola MCORE 2114 processor at 32 MHz with 32 KB of RAM [84]. Supporting material for the board is low, with no manual or datasheet originally available. The Phoenix could bypass hardware development delay prior to in-house software development. Processor performance is expected to be between that of MicroPilot and Piccolo. There is little academic reference to its use, though Stanford University have used it on an autonomous ground vehicle as a test of concept for forest-flying UAV's [85]. A successful practical demonstration was performed,

with no reported hardware difficulties. O-Navi quote the Phoenix system in the region of 1200 USD.

Custom autopilot hardware developed by academic institutions may be similarly grouped into those that use low or high computational power processors. The Zagi UAV's developed by Brigham Young University use the RCM3100 microcontroller [32] based on the low computational power Rabbit 3000 processor [2, 86]. The 8-bit processor runs at 29 MHz with 512 KB RAM. The processor is described as being fast enough to run UAV sensor processing and control algorithms, but inadequate for complex operations such as video processing [32]. The module is very low cost (around 35 GBP) and "extensive" [32] C libraries are provided. Following the success of the Brigham Young University Zagi UAV projects, the same RCM3100-based autopilot is sold commercially as the Kestrel [87]. The processor used for the NASA seaplane UAV [26] was a 32-bit Motorola 68336 running at 20 MHz with 7.5 KB RAM [88]. It is an improved version of the Motorola 68332 used by MicroPilot. Georgia Institute of Technology considered a 16-bit 80386SX Intel processor running at 33 MHz with a large 4 MB of RAM or the Motorola 68332 (MicroPilot) for their Piper-based UAV's [34]. They chose the former, though do not elaborate on their decision.

Regarding the higher computational power processors, a number of universities implement a PC/104+ based system. Though the PC/104+ standard relates to a hardware specification rather than a specific processor, the computing power required to meet these guidelines is typically high. Stanford University's DragonFly series [48] use a rugged VersaLogic PC/104+ SBC running the QNX operating system, though the specific model is not referenced. The North Carolina State University Stingray UAV also uses a PC/104+ SBC [53, 54] running Linux on a CMH586DX133-64 processor by Real Time Devices Inc. The board operates at 133 MHz with 64 MB of RAM [53]. A particularly high-power PC/104+ system is used on board various University of California UAV's [42] and University of Columbia UAV's [89] with processor speeds ranging from 700-800 MHz. In both cases the PC/104+ is used for complex navigation operations and runs the QNX operating system, though actual flight control is performed by a Piccolo autopilot. A further PC/104+ user is Cornell University's "economic UAV" project. An 800 MHz Crusoe processor with 180 MB RAM runs Windows XP Embedded. Much of the flight software is written to utilise the Windows .NET framework supported by the embedded operating system. Though in-house autopilot software is used, attitude estimation is performed by a third-party device. Given the cost of this extra device is in excess of 1000 USD [90], and the considerable computing

power already available on the processor, it is surprising this route was adopted given the low-cost aim. As may be expected from a system as powerful as this, the 15 W power usage reported may well make long duration flights infeasible.

A recent review on the state of autopilots for small fixed-wing UAV's is presented in [91]. The wide scope of the paper has lead to inevitable brevity in some aspects, though no new components are presented that have not previously been reviewed.

In conclusion, the typical hardware components of an autopilot consist of 3-axis MEMS accelerometers and gyroscopes, static and dynamic pressure sensors and a GPS module. The wide range of embedded processors reviewed may be broadly grouped into high or low computational power. The literature suggests low computational power processors are still capable of providing satisfactory autonomous flight control. Given the typically higher power consumption of processors with greater clock speeds, a low computational power processor is recommended to maintain operational endurance from a battery power source.

Low power processors have been reviewed with 8, 16 or 32-bit architectures. The majority of arithmetic operations performed by the autopilot software are expected to be 32-bit (particularly 32-bit floating point calculations); a lower width architecture would be unlikely to achieve sufficient processing speed due to the increased number of instructions required per calculation.

The Phoenix board provided by O-Navi is chosen as best matching the research criteria. The board integrates all autopilot components identified for a base system. Though potentially more expensive than building an in-house solution, the board cost remains below the autopilot budget and is expected to free development time for algorithmic work directly attributed to the research objectives.

2.4 Sensor Fusion

Study of UAV and autopilot literature identifies the almost universal use of sensor fusion algorithms for determining the aircraft's orientation.

With perfect gyroscopes, the angular difference between initial and current aircraft orientation could be determined simply by integrating the angular rate data. While very high precision gyroscopes are available, their cost (and in most cases size) make such sensors unattainable for the low-cost autopilot. The actual gyroscopes used are considerably cheaper, at the expense of accuracy. An alternative

orientation calculation uses accelerometers to sense the gravity vector and estimate aircraft attitude in relation to this. Unfortunately as soon as external forces are applied to the aircraft, particularly from centripetal acceleration during turns, the accelerometers cease to measure the gravity vector alone and estimation errors occur. The solution lays with sensor fusion to determine the best orientation estimate from both types of sensor data. The short-term accuracy of the gyroscopes is fused with the long-term stability of the accelerometers to provide a best estimate.

Additional estimation problems occur in the use of GPS to determine position, ground speed and ground heading. While some error is present in the GPS data, of particular concern is the relatively slow update rate and a non-guaranteed output. The GPS module included with the O-Navi board has a fixed update rate of 1 Hz, which is considered too slow for accurate path tracking - particularly during test flights when the distance covered by the aircraft between updates represents a significant fraction of the available flying area. As with all GPS units, the device will cease output should it fail to track sufficient satellites. Such an occasion can occur due to heavy precipitation absorbing the signal, obstruction due to buildings or during moderate changes in bank or elevation angle, leading to the aerial pointing away from some of the tracked satellites and/or their signal being blocked by other parts of the airframe. The slow update rate and potential for data loss (albeit temporary) both necessitate the use of an alternative position and velocity estimation technique to fuse with the GPS data. Accelerometer readings can be used, though the bias error present in low-cost sensors make them only suitable for very short period estimates. An alternative complement to GPS is dead reckoning using air or ground speed measurements and current heading. The use of bias-corrected gyroscopes is a suitable estimate for high frequency components of heading.

Stand-alone commercial products employing sensor fusion for orientation estimation are reviewed in the first subsection, followed by the algorithms themselves with the aim of implementation by the in-house autopilot software.

2.4.1 Commercial INS's

Self-contained inertial navigation systems (INS's) may be purchased as an independent component. INS's estimate position and/or orientation using inertial measurements. Such measurements are typically obtained from three-axis accelerometers and gyroscopes. Where these sensors are contained in a single hardware unit,

this hardware is referred to as an inertial measurement unit (IMU). An INS that only estimates orientation is often referred to in the literature as an attitude and heading reference system (AHRS), though such devices typically include a three-axis magnetometer to provide an additional compass heading. All INS's reviewed in this subsection are AHRS's, since additional position estimation brings their complexity and cost to similar levels seen in complete commercial autopilots.

MicroStrain produce the 3DM-G [90], as used by Cornell University's UAV [44]. The AHRS is said by Cornell to offer a very reliable method of determining attitude, was "extremely durable" and lightweight. The device is currently available for 1195 USD and is one of the lowest-cost systems reviewed. The price is in the region of 3-6 times cheaper than a complete commercial autopilot system [32], or half the cost of a μNAV/Stargate solution with open-source software. Given additional autopilot hardware will always be required to act on this data, and such hardware is typically capable of performing AHRS functionality itself with minimal sensor cost, the purchase of a separate device is unlikely where low cost is a driving factor. The use of the MicroStrain AHRS for Brigham Young University's Zagi UAV was also rejected in the belief it would produce inaccurate measurements during coordinated turns [32], though the company's fact sheet [90] does declare its accuracy for dynamic as well as static conditions. This AHRS has been used successfully in helicopter autopilots by some entries to the International Aerial Robotics Competition [92–94]. Arnold et al [92] claim the vibration caused by the helicopter rotors made the MicroStrain data unusable without the addition of a custom isolation mount. The same device is carried in a payload pod of the University of Pennsylvania's Piper UAV's [65], though it is used for imaging applications and not aircraft control.

Norwegian company Xsens produce an AHRS at twice the cost, though it is specifically marketed for autonomous vehicles [95]. No reference to its use has been found in academic literature, though the product is relatively new. Crossbow Technology Inc. also produce a range of AHRS systems, though are heavier and more expensive [32, 96] than previous references and as such unsuitable for this project. Some of the DragonFly UAV's have used a Crossbow AHRS [97], as has an experimental VTOL UAV design from the University of Sydney [98].

While an exhaustive review of currently available solutions is not included, it is believed these three products represent the range of AHRS hardware available following examination of trade listings and previous studies [32, 99]. Though a COTS INS would save development time by not requiring sensor fusion algorithms

to be developed, it does not add sufficient value to justify the increase in autopilot hardware cost. INS functionality will instead be incorporated into the in-house autopilot software using the algorithms described in the remainder of this section.

2.4.2 Kalman Filter

A form of Kalman filtering is the standard algorithm for combining inertial measurements to produce an AHRS [100–102] reviewed commercially in products by MicroStrain and Crossbow Technology [90, 96]. The same methodology is adopted by nearly all UAV projects who chose to develop their own attitude estimators [2, 27, 39, 48] and also by commercial autopilots [3, 80].

Welch et al [1] provide a detailed description of the algorithm, with an article by Simon [103] also a useful foundation. Key advantages of the Kalman filter in general are its ability to complement the strengths and weaknesses of noisy sensor combinations [102], estimate states which are not directly measured (such as gyroscope bias) and replace failed or unavailable sensor outputs with estimates [104]. Application to aircraft attitude estimation is introduced by [101, 102, 105].

The Kalman filter consists of a time update followed by a measurement update. In the time update, a prediction of the future value of the estimated states is provided using a custom process model. The measurement stage compares this prediction to the true (but noisy) state measurements, and adjusts the Kalman gain and error covariance matrices in an attempt to minimise the mean squared error of the estimation. This procedure begins again upon the next filter update cycle, allowing a recursive estimation suitable for real-time operation. The iterative adjustments to the Kalman gain effectively alter the balance of "trust" between the predicted and measured states, allowing the filter to select and track the optimal weighting.

An extended Kalman filter (EKF) is typically used in preference to the standard Kalman filter when aircraft dynamics are estimated [1, 105] and is the dominant implementation in UAV's. In contrast to the linear Kalman filter the EKF supports non-linear time and measurement update functions, which better model true aircraft dynamics.

The mathematics of the EKF are outlined below with reference to Welch et al [1]. For iteration k the new estimated states x_k are described by the non-linear difference equation f (Equation 2.1) with reference to the previous states x_{k-1},

control inputs u_{k-1} and process noise sources w_{k-1}.

$$x_k = f\left(x_{k-1}, u_{k-1}, w_{k-1}\right) \tag{2.1}$$

The measurements z_k are related to the current estimated states by non-linear equation h (Equation 2.2) with v_k representing the measurement noise sources.

$$z_k = h\left(x_k, v_k\right) \tag{2.2}$$

All process and measurement noise sources are expected to be independent of each other, white and follow normal probability distributions (Equation 2.3).

$$\begin{aligned} p\left(w\right) &\sim N\left(0, Q\right) \\ p\left(v\right) &\sim N\left(0, R\right) \end{aligned} \tag{2.3}$$

Process and measurement noise covariance matrices are Q and R respectively. Though they can change with each time step, this notation (and all practical implementations reviewed) assumes they do not.

Since the true noise offsets at a given iteration are unknown, an *a priori* state estimate $\hat{x}_k^-$ and measurement $\bar{z}_k$ is provided by Equation 2.4 by ignoring the noise sources.

$$\begin{aligned} \hat{x}_k^- &= f\left(\hat{x}_{k-1}, u_{k-1}, 0\right) \\ \bar{z}_k &= h\left(\hat{x}_k^-, 0\right) \end{aligned} \tag{2.4}$$

In order to project the error covariance matrix P and update the Kalman gain matrix K, the model is linearised about these estimates using a series of Jacobian matrices of partial derivatives, shown by Equations 2.5–2.8. Though the Jacobians are recalculated at each iteration, the k subscript is removed from the notation

for simplicity.

$$F_{[i,\,j]} = \frac{\delta f_{[i]}}{\delta x_{[j]}} \left(\hat{x}_{k-1}, u_{k-1}, 0\right) \tag{2.5}$$

$$W_{[i,\,j]} = \frac{\delta f_{[i]}}{\delta w_{[j]}} \left(\hat{x}_{k-1}, u_{k-1}, 0\right) \tag{2.6}$$

$$H_{[i,\,j]} = \frac{\delta h_{[i]}}{\delta x_{[j]}} \left(\hat{x}_k^-, 0\right) \tag{2.7}$$

$$V_{[i,\,j]} = \frac{\delta h_{[i]}}{\delta v_{[j]}} \left(\hat{x}_k^-, 0\right) \tag{2.8}$$

The *a priori* error covariance matrix P_k^- is projected from the previous iteration $k-1$ to the current k iteration by Equation 2.9.

$$P_k^- = F_k P_{k-1} F_k^T + W_k Q_{k-1} W_k^T \tag{2.9}$$

The new Kalman gain matrix K_k is calculated by Equation 2.10.

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R_k V_k^T\right)^{-1} \tag{2.10}$$

The *a priori* state estimates $\hat{x}_k^-$ are corrected by the measurement update to form the *a posteriori* state estimates $\hat{x}_k$ using the measurement residual and Kalman gain (Equation 2.11).

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - \bar{z}_k\right) \tag{2.11}$$

The *a priori* error covariance matrix P_k^- is similarly corrected to form P_k as Equation 2.12. I is the identity matrix.

$$P_k = \left(I - K_k H_k\right) P_k^- \tag{2.12}$$

Only P and $\hat{x}$ are carried over to each new iteration. As a result they must be initialised prior to the first iteration. Typically P would be initialised to an identity matrix, though the diagonal elements can be any value other than 0, with higher values suggesting a lower trust in the initial $\hat{x}$ estimate. A summary of the EKF implementation is shown in Figure 2.1, based upon a similar diagram in [1].

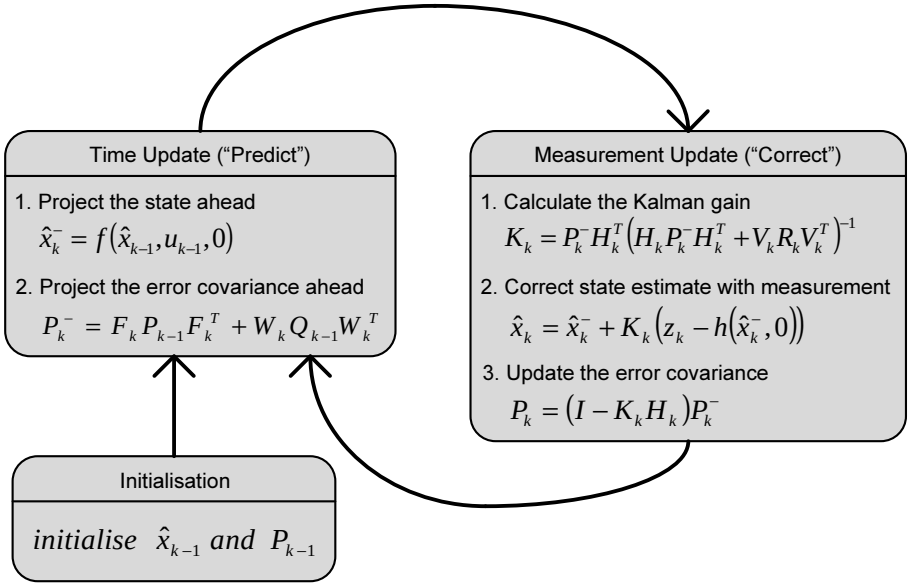


FIGURE 2.1: Summary of EKF operation (based upon a similar diagram in [1])

To complement the introduction by Welch et al, further depth is provided by dedicated books such as those by Grewal et al [106] and Crassidis et al [107]. Papers by McBurney [108] and Gopalratnam et al [109] show sensor failure can be detected by monitoring measurement residuals for abnormally high values, along with the ability to flag and manage filter divergence. In both cases the redundancy provided by the multiple sensor readings allow valid state estimates to continue, providing the update processes is quickly aware of the failure. Brief practical advice for standard Kalman filter implementations is provided by Pedersen [110], highlighting key areas of filter operation likely to destabilise estimations.

The potential for numeric instabilities have led to several variants of the original Kalman filter. Most aim to reduce the affect of numeric round-off errors inevitable in practical implementation. The *Joseph stabilised version* [111] is one such variant requiring an expansion of the standard error covariance update (Equation 2.12) to Equation 2.13.

$$P_k = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \tag{2.13}$$

This expansion ensures P_k meets the required positive-definite condition even in the presence of round-off errors. Alternatively the Kalman Riccati equations (Equations 2.9, 2.10 and 2.12) can be reformed using Cholesky factors as described by [106]. One method of accomplishing this (without the use of square-roots) is the modified Cholesky (UD) decomposition, where a unit triangular matrix U and diagonal matrix D are extracted from a symmetric positive-definite matrix M

such that $M = UDU^T$. The error covariance matrix should always be a symmetric positive-definite matrix, and using U and D matrices within the Kalman Riccati equations increases the filter's numeric stability as well as offering faster matrix inversion [106]. The Bierman-Thornton UD filtering technique is used in [112] to help a UAV track a moving object for sense and avoid abilities, with satisfactory performance concluded.

Modification to the underlying extended Kalman filter process itself is also suggested in an attempt to improve the filter's characteristics. One such example is the unscented Kalman filter (UKF). This technique attempts to improve upon estimation of highly non-linear systems by offering a higher-order Taylor series expansion of the model's probability distribution [107]. An unscented transform sampling technique selects multiple sample points around the current mean state estimate at each filter iteration. These sample points are propagated through the non-linear model and their distribution provides an improved estimate of the true mean and variance. An additional advantage of this technique is the removal of the need to provide Jacobian matrices of the model and measurement equations (as required by the EKF), particularly useful for complex model equations. Lievens [113] and Niculescu [104] both claim improved performance for UAV attitude estimation using an UKF and comparing to an equivalent EKF. However a significant disadvantage lies with the increased computational requirements of the UKF algorithm [104, 107], with a ten-fold increase in calculation time cited by Lievens. St-Pierre et al [114] compare EKF and UKF estimates of a position-only INS aided by GPS measurements. They conclude the low dynamics of the vehicle meant the linearisation errors of the EKF were not significantly high for the UKF to demonstrate any significant performance improvement. They also highlight the increased computational time of the UKF (65.8 ms versus 2.8 ms average per iteration). Following this review of UKF literature it is believed the EKF linearisation error should remain low enough to closely match UKF performance providing the autopilot's EKF iterates at a sufficiently high frequency when compared with the aircraft's dynamics. Under such conditions the EKF would be preferable due to its significantly lower computational requirements.

Details of the specific implementation for the custom autopilot is provided in the System Design chapter (Section 3.3.5).

2.4.3 Complementary Filter

A complementary filter simply combines a high-pass filtered measurement with a low-pass filtered measurement [115], and may in fact be considered a steady-state Kalman filter [116]. The simplicity of the algorithm means less computational resources are required when compared to a Kalman implementation. The basic concept for attitude estimation remains the same, with high-frequency gyro measurements and low-frequency accelerometer measurements being trusted [115, 117]. While the technique is used for attitude estimates across a range of applications, very few references are found for UAV's. A few helicopter projects implement one [117, 118] and both papers show satisfaction with the filter's performance. A follow up to [117] however states the complementary filter was later replaced by a Kalman filter [119]. This decision was made due to the inability of a complementary filter to estimate non-measured properties, of which gyro bias was found to be an important factor. The performance of the Kalman filter was compared to the complementary, and found to be more accurate once these bias terms were estimated. [118] attempts to minimise the effect of gyro bias in their complementary filter estimates by ensuring the accelerometer measurements were used in the widest possible frequency range, reducing the influence of low frequency gyro errors.

The decision to develop an AHRS in-house is supported by the quantity of published material reviewed in these last two subsections, low cost of required sensors (already provided by the Phoenix board, Subsection 2.3.2) and the number of successful UAV's flown using reviewed algorithms.

2.4.4 Attitude Representation

A representation of the aircraft's attitude is required for all sensor fusion techniques. While Euler angles (bank ϕ, elevation θ and heading ψ) are the simplest, they are not the best choice for internal representation during the estimation processes. This is due to the singularities introduced as the aircraft approaches certain angles, a susceptibility to "gimbal lock" (when two axes become aligned so as to loose the information of one of them) and the requirement of computationally expensive trigonometric functions. Such problems are cited and resolved through the use of four-dimensional quaternions by numerous papers describing attitude-estimating EKF's [120–123]. Kuipers [124] provides an in-depth reference on quaternions and rotation science. The quaternion q is a sum of a scalar q_0 and

a vector $\mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$ where $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ are orthonormal. The properties and operators of quaternions are further discussed by Kuipers.

Euler angles are converted to a quaternion with components $q_{0\ldots3}$ by Equation 2.14.

$$\begin{aligned} q_0 &= \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ q_1 &= \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ q_2 &= \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ q_3 &= \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \end{aligned} \tag{2.14}$$

A quaternion a can be rotated by quaternion b to form a new rotated quaternion c using the quaternion multiplication of Equation 2.15. This assumes a represents an orientation rather than a point, and b is relative to a's frame of reference.

$$\begin{aligned} c_0 &= a_0b_0 - a_1b_1 - a_2b_2 - a_3b_3 \\ c_1 &= a_0b_1 + a_1b_0 + a_2b_3 - a_3b_2 \\ c_2 &= a_0b_2 + a_2b_0 + a_3b_1 - a_1b_3 \\ c_3 &= a_0b_3 + a_3b_0 + a_1b_2 - a_2b_1 \end{aligned} \tag{2.15}$$

During the time update stage of an EKF for an AHRS, the three-axis body-referenced angular rates (p, q and r) measured by the gyroscopes are added to the aircraft's estimated orientation to provide a new prediction (Subsection 3.3.5). Bachmann [123] shows how this step can be performed by a quaternion-based EKF without using trigonometry. If the aircraft rotates by angle γ about body axis $\mathbf{B}$, the rotation quaternion ϖ is expressed by Equation 2.16.

$$\varpi = \left(\cos\frac{\gamma}{2},\ \mathbf{B}\sin\frac{\gamma}{2}\right) \tag{2.16}$$

If the time period Δt between EKF iterations is small ($\ll$1 s), the angle $\frac{\gamma}{2}$ should be small enough to use small angle approximation (Equation 2.17) to produce Equation 2.18.

$$\begin{aligned} \cos x &\simeq 1 \\ \sin x &\simeq x \end{aligned} \tag{2.17}$$

$$\varpi = \left(1,\ \mathbf{B}\frac{\gamma}{2}\right) \tag{2.18}$$

Given Δt is small it is also assumed the angular rates will remain constant during this time. Components of γ about each body-axis become $(p,\ q,\ r)\Delta t$ and ϖ becomes Equation 2.19.

$$\varpi = \left(1, \frac{p\Delta t}{2}, \frac{q\Delta t}{2}, \frac{r\Delta t}{2}\right) \tag{2.19}$$

Representing the last three components of ϖ as $\hat{p}$, $\hat{q}$ and $\hat{r}$ the aircraft orientation quaternion Q is updated to Q^+ by $Q\varpi$, substituting Equation 2.15 to form Equation 2.20.

$$\begin{aligned} Q_0^+ &= Q_0 - \hat{p}Q_1 - \hat{q}Q_2 - \hat{r}Q_3 \\ Q_1^+ &= Q_1 + \hat{p}Q_0 - \hat{q}Q_3 + \hat{r}Q_2 \\ Q_2^+ &= Q_2 + \hat{p}Q_3 + \hat{q}Q_0 - \hat{r}Q_1 \\ Q_3^+ &= Q_3 - \hat{p}Q_2 + \hat{q}Q_1 + \hat{r}Q_0 \end{aligned} \tag{2.20}$$

Whenever an orientation quaternion is modified in software (such as by the EKF in Equation 2.20), internal rounding errors may result in the quaternion failing to remain normalised [69, 125]. The quaternion q is "renormalised" by Equation 2.21, and should be performed at the end of each EKF iteration to maintain stability.

$$q = \frac{q}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}} \tag{2.21}$$

The quaternion-based EKF AHRS literature also identifies the Direction Cosine Matrix (DCM) as a method of transforming between Earth-referenced vectors (such as gravity and compass heading) and aircraft body-reference. Such transforms are needed in the EKF measurement update. When the DCM is formed from aircraft orientation quaternion Q by Equation 2.22 its multiplication with an Earth-referenced vector performs the transform [124].

$$\begin{bmatrix} 2Q_0^2 - 1 + 2Q_1^2 & 2Q_1Q_2 + 2Q_0Q_3 & 2Q_1Q_3 - 2Q_0Q_2 \\ 2Q_1Q_2 - 2Q_0Q_3 & 2Q_0^2 - 1 + 2Q_2^2 & 2Q_2Q_3 + 2Q_0Q_1 \\ 2Q_1Q_3 + 2Q_0Q_2 & 2Q_2Q_3 - 2Q_0Q_1 & 2Q_0^2 - 1 + 2Q_3^2 \end{bmatrix} \tag{2.22}$$

Though an extra estimated state is required when using quaternions instead of Euler angles in an EKF, the removal of computationally-intensive trigonometric functions and increased stability would likely make a quaternion EKF preferable to an Euler EKF for an AHRS.

Though not performed during the EKF process, a quaternion q is converted back to Euler angles by Equation 2.23.

$$\begin{aligned}\phi &= \texttt{atan2}\left(2q_0q_1 + 2q_2q_3,\ 2q_0^2 + 2q_3^2 - 1\right)\\ \theta &= \sin^{-1}\left(2q_0q_2 - 2q_1q_3\right)\\ \psi &= \texttt{atan2}\left(2q_0q_3 + 2q_1q_2,\ 2q_0^2 + 2q_1^2 - 1\right)\end{aligned} \tag{2.23}$$

The `atan2` macro [126] calculates the four-quadrant inverse tangent and is common to many programming languages. As θ approaches $\pm\frac{\pi}{2}$ the north and south pole singularities of the Euler representation appear, shown by the second term of the `atan2` macros approaching zero (corresponding to a division by zero). To resolve this in practical implementations, Baker [127] suggests Euler angles where $q_0q_2 - q_1q_3 \rightarrow \pm 0.5$ are approximated by Equation 2.24. The trigger threshold depends on the accuracy of the software's numeric representation and functions.

$$\begin{aligned}\phi &= 0\\ \theta &= \frac{\pi}{2}\,\text{sign}\left(q_0q_2 - q_1q_3\right)\\ \psi &= 2\,\texttt{atan2}\left(q_3,\ q_0\right)\end{aligned} \tag{2.24}$$

The EKF continues to track the true orientation through the quaternion, remaining unaffected by the approximation.

Euler angle representation retains advantages over quaternions due to its ease of human visualisation [128, 129] and direct application to the controller architectures (Subsection 2.6.2). This ability of the individual to clearly understand the internal orientation representation is essential for software debugging, simulation and flight operations. As a result it is expected the Euler angle form will remain in use outside of the EKF algorithm.

2.5 Direct Attitude Estimation

2.5.1 Horizon Sensing

An alternative approach to sensor-fusion attitude measurement is suggested by Christiansen [32] as the use of orthogonal infra-red sensors. The sensors produce an error signal proportional to the difference in sky and ground temperature. The further the aircraft deviates from level flight, the greater this error becomes (with a

maximum should the aircraft be at right angles to the ground). A well-calibrated sensor set was found to produce estimates accurate to within 7 degrees, worse than the sensor fusion techniques reviewed. Calibration was considered difficult and lengthy, and the process needed repeating (typically hourly) as environmental conditions changed. These factors lead to Christiansen's conclusion that the technique was inappropriate for their UAV project.

Some basic and hobbyist semi-autonomous designs adopt this method [130, 131] though further use is very rare in academic literature. One exception mitigates calibration issues by the use of a movable platform mounted on top of the UAV [132]. The platform contains the thermal sensors and is rotated to the negative value of the commanded flight angle. The autopilot controls the aircraft to minimise the thermal sensor error, keeping the platform aligned to the horizon. This method has been verified by several successful autonomous flights using an ARF model. A quantitative description of attitude estimation errors using this approach are not provided. The method is noted as being inappropriate during non-visible meteorological conditions (such as fog) or flight near high terrain (such as in canyons or low-altitude urban environments).

While the very low cost of the thermal sensors combined with very low computational requirements is attractive, environmental and accuracy limitations make this approach unsuitable for the project.

2.5.2 Multiple GPS Antennas

A direct measurement of aircraft attitude is achieved using multiple GPS antennas. Favey et al [133] use four GPS receivers placed close to the aircraft's extremities. One receiver is arbitrarily chosen as a reference and differential GPS analysis using carrier phase offsets calculates the locations of the other receivers. These locations determine the aircraft's attitude. The results are of similar accuracy to the independent on-board INS, though conditions such as satellite signal loss and interference are noted as potential hazards. The aircraft under test was a commercial manned aircraft and the attitude estimates were not therefore used for autonomous control. A similar scheme is used to successfully control an autonomous helicopter in [134], though no quantitative analysis of attitude errors are provided. Test flights involving carrier-phase GPS attitude measurements were also performed using the DragonFly UAV [135] though this data was augmented with inertial measurements using a Kalman filter to provide the final estimates.

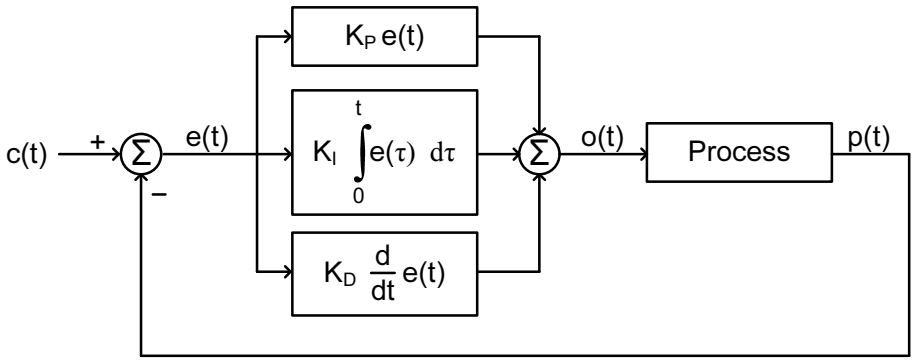


FIGURE 2.2: PID controller structure

The high cost and bulky nature of the differential GPS receiver and antenna equipment is suggested by Simsky et al [136] as being particularly restrictive for UAV's. These concerns, combined with a reliance on a good quality and continuous GPS signal, make this approach inappropriate for the project.

2.6 Controllers

2.6.1 Implementation

One of the primary functions of an autopilot is the ability to control the aircraft response to match commanded parameters. This ability is largely determined by the performance of the control algorithm implemented by the autopilot software. All commercial autopilots reviewed state their control algorithm uses the well-established proportional, integral, derivative (PID) controller [3, 57]. The structure of a continuous time PID controller is shown in Figure 2.2. For this case the process is the aircraft. At a given time t the process state under control p is compared to the commanded value c to produce an error signal e. Proportional, integral and derivative functions are applied to e and weighted in accordance to the controller gain constants K_P, K_I and K_D. The function outputs are summated to produce the controller output o that is fed back into the process. The simplicity of the algorithm would lead to a straightforward software implementation.

The end result is the PID controller attempts to minimise the error e between a commanded and measured process state. The manner by which it adjusts its output in response to this error is characterised by the three gain constants. These gains must be tuned to provide an optimal controller response when matched with

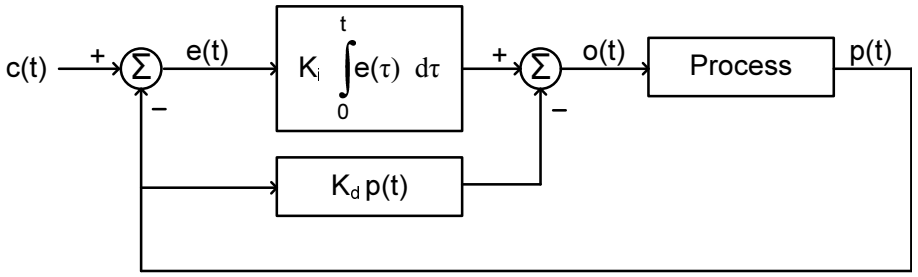


FIGURE 2.3: PDF controller structure

the particular behaviour of the process. An explanation of the PID controller in an autopilot context is provided by the MicroPilot MP2028 manual [57].

Of particular importance is obtaining gain values capable of satisfactorily controlling the aircraft. While methods are found in academic literature for the tuning of generic PID controllers [137], such techniques tend to be applicable only to well-defined, first-order transfer functions of the system under control. Accurate transfer equations of the practical aircraft dynamics are rarely known in practise [57, 137]. A popular PID tuning formula used in industry is the Ziegler-Nichols method. Though it does not require knowledge of the system transfer function, it is shown to have a tendency to produce an oscillatory [137] and generally non-optimal response [138]. Eng et al [139] used this technique to tune a simulated UAV, but state additional experimental tuning was required.

Trial and error is given by the MicroPilot manual as the most practical method for tuning controller gains. The Piccolo vehicle integration guide also claims there is no "magic formula" for determining the gains [140], with trial and error being the primary tuning method – initially using their aircraft simulator. A similar technique is used for custom autopilot controller tuning. Like the commercial autopilot packages, the ground station interfaced to Brigham Young University's UAV has the ability to view and adjust PID controller information in real time [2]. This feature is said to have accelerated their trial and error tuning process and allows new airframes to be tuned in less than 15 minutes. Cornell University's paper [44] also emphasise this method with their custom software.

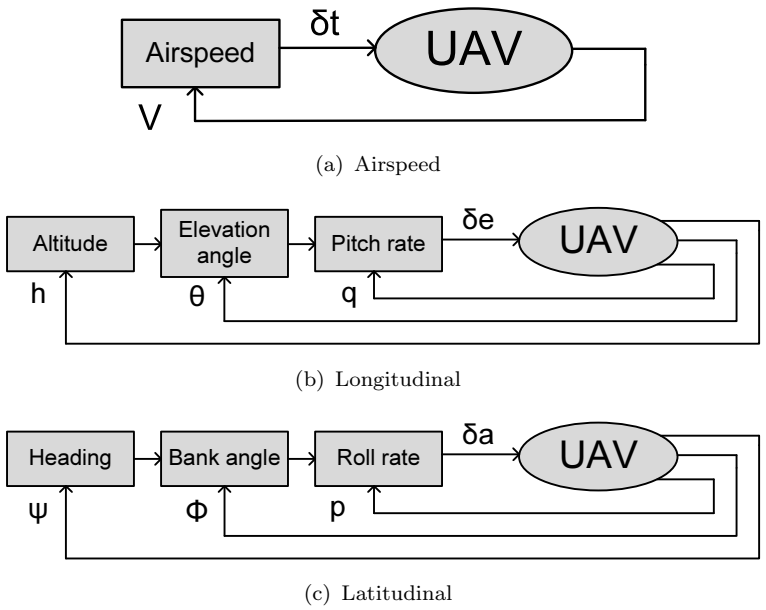
Of similar algorithmic simplicity to the PID controller is the pseudo-derivative feedback (PDF) controller. The PDF controller is based on the work of Phelan [141] who claimed the proportional and derivative terms are more effective in the feedback stage of the controller than the forward stage. This is the key difference to the PID controller, where all terms are in the forward stage. The continuous time PDF controller structure is shown in Figure 2.3. As for the PID controller, output

o controls a desired process state p in an attempt to maintain it at commanded value c. Only two gain values are required, integral K_i and zero-order feedback K_d, though must be tuned to the process behaviour as for the PID.

Compared to the PID controller, PDF usage amongst academic literature is relatively rare. Those papers [142–144] who do test a PDF scheme conclude satisfactory control performance in a range of industrial applications, and outperforms a conventional PID controller when comparisons are made. Of particular interest is a paper documenting its use in ship autopilots [145]. The paper claims the PDF algorithm demonstrated a "significant improvement" over the corresponding PID for autonomous ship steering. The PDF autopilot had a greater ability to resist, and recover faster from, load disturbances. Smoother changes in rudder movement, less overshoot and better settling times were also recorded. A similar paper by the same authors present similar conclusions with PDF controllers in cascade [146]. These areas of improvement correspond to those observed in the other comparative papers. Since there appear to be no literature relating to the use of PDF controllers in UAV autopilots, should similar improvements be feasible the result will be of considerable interest. Similarly to PID controllers, no suitable tuning algorithms are found for PDF controllers. Paraskevopoulos et al. [147] identify a series of PDF tuning algorithms, though the system identification process requires the accurate measurement of several aspects of the controller output under controlled conditions which is unlikely to be achieved under practical flight conditions. The identification and tuning algorithms assume the system model is unstable first-order plus dead-time, which may not be applicable for the true aircraft dynamics. As a result this tuning technique is considered unlikely to be sufficiently robust for this application. One disadvantage of a PDF implementation in a UAV context is the apparent absence of guidelines even for manual trial and error tuning, while well established rules exist for PID gains [140].

Prevention of integral wind-up and handling of output saturation are two additional issues to be considered in a practical implementation. Both cases and their relatively simple remedial steps are covered well by the literature, though Pernebo et al [148] cover these aspects particularly well. PID and PDF controllers may be treated equally in this case, since both share a forward integral term. Further detail is provided in Subsection 3.3.4.

While a number of additional control algorithms are presented in the literature, the simplicity and performance reported by PID and PDF controllers make them clear starting points for autopilot design.



(a) Airspeed

(b) Longitudinal

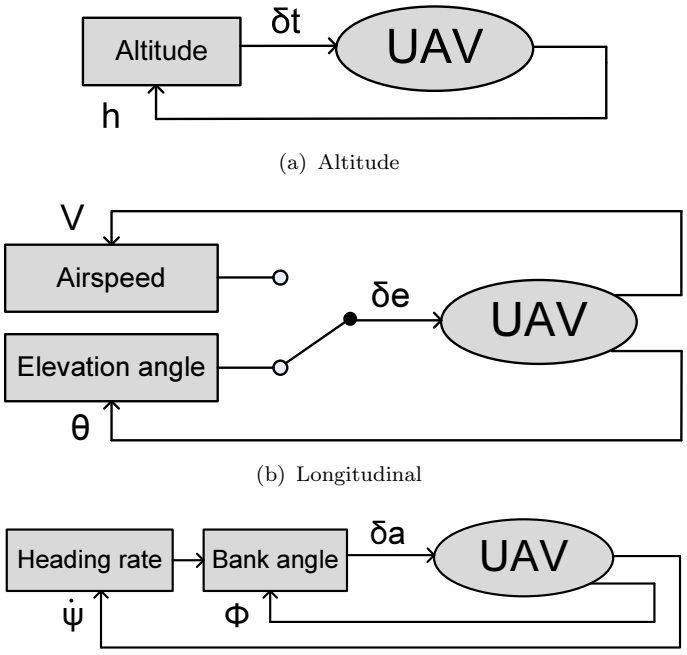
(c) Latitudinal

FIGURE 2.4: Brigham Young University UAV controller architecture [2]

2.6.2 Architecture

The architecture by which these controllers are arranged is found to be similar across the autopilots reviewed. In each case it is assumed the longitudinal and lateral dynamics of the aircraft are decoupled, and as such controller paths for each are designed independently. For Brigham Young University's UAV [2] a single control loop commands airspeed V by adjusting the throttle δt. Longitudinal and lateral control paths consist of three cascaded controllers. For longitudinal, an outermost controller commands elevation angle θ from desired altitude h. The output is fed to an elevation angle controller, commanding pitch rate q. The commanded pitch rate is fed to an innermost controller adjusting elevator deflection δe. Similarly for lateral control, an outermost heading ψ controller commands bank angle ϕ. This signal is fed to a bank angle to roll rate p controller. This in turn feeds to an innermost controller adjusting aileron deflection δa. This architecture is shown in Figure 2.4.

North Carolina State University's Stingray autopilot [53] adopt a simpler autopilot control structure. Five PID controllers hold pitch rate, yaw rate, altitude, airspeed and bank angle. A turn coordinator is included, taking airspeed and bank angle as inputs and providing commanded pitch and yaw rates. Brigham Young did not implement direct yaw rate control since their Zagi airframe does not have a rudder.



(a) Altitude

(b) Longitudinal

(c) Latitudinal

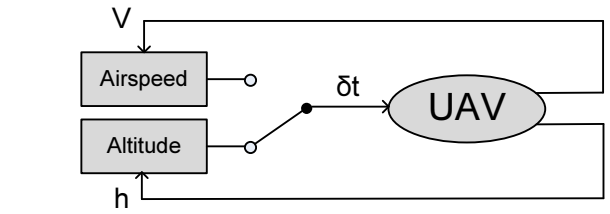
FIGURE 2.5: Simplified Piccolo controller architecture [3]

The Piccolo autopilot (before version 2) also decouples longitudinal and lateral control [3, 65], and does not recommend exceeding 30 degrees bank to prevent significant interference between the two. However, their low-level implementation differs from [2]. Airspeed is controlled by elevation angle and altitude is controlled by throttle setting. The pitch and roll rate controllers are removed, with angle hold controllers commanding elevator and aileron deflection directly. Since these deflections directly affect rates rather than angles, non-linearities are introduced in the control process. The PID controllers adequately control such dynamics but with a simpler control path. The airspeed controller commands elevator directly making specific elevation angle and airspeed control redundant. A turn rate controller commands bank angle, rather than the direct heading controller of [2]. A simplified controller architecture is shown in Figure 2.5 demonstrating the alternate airspeed and altitude control. A high-level path tracking algorithm calculates commanded turn rate. An optional turn-compensator mixes in extra elevator and throttle action during a turn. This provides an approximation to a coordinated turn, though some knowledge of the aircraft parameters (such as weight and wing area) is required. The ability to switch to altitude control by elevator is retained for mission phases where precise altitude control is required, such as during automated landing. A yaw dampening controller is also available in manual flight mode where yaw rate is fed back to rudder.

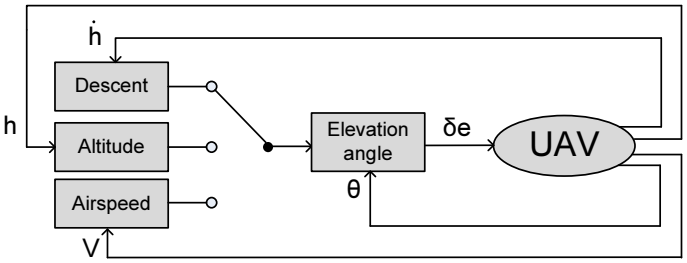
MicroPilot implement a similar structure [57] to the Piccolo (pre-v2) autopilot, though the concept of control path switching (i.e. switching airspeed control between throttle and elevator) is more integrated with the flight requirements. During straight and level cruise between waypoints, the architecture is typically in "airspeed controlled by throttle" and "altitude controlled by elevator" mode. The option still exists to control altitude by throttle however. When the commanded altitude is increased following the switch to a new waypoint, the control paths are automatically switched to a "climb" architecture. Here, the throttle is fixed and airspeed is controlled by elevation angle. When descending to a new altitude, the throttle is again fixed but elevation angle is controlled by a commanded vertical descent speed. A separate path is also activated for pre-take-off (elevation, bank and throttle settings are fixed). While the advantage of airspeed controlled by throttle and altitude controlled by elevation has already been stated as a more precise control of altitude, MicroPilot confirms the primary advantage of the alternative method (airspeed by elevation and altitude by throttle) as offering a safer response should the engine fail. This method may require a greater number of trial and error iterations to obtain acceptable controller gains [57]. The yaw controller can command rudder deflection in response to commanded heading or to reduce sideslip (determined from the latitudinal accelerometer). The control structure is shown in Figure 2.6 identifying main path switching routes. As with previous architectures all controllers can take fixed-value commands, though these paths have been omitted for clarity. Unlike the Piccolo, all longitudinal control passes through a low-level elevation angle controller.

Both MicroPilot and Piccolo (pre-v2) autopilots offer optional gain scheduling - a feature not found in other reviewed literature. All the Piccolo controllers (except altitude hold via throttle) have the ability to scale their gains according to dynamic pressure. The gains entered at the ground station during tuning are considered referenced at a fixed dynamic pressure [3]. The MicroPilot manual [57] claims most gains require reducing as airspeed increases. MicroPilot's scheduling method provides users with the optional ability to enter different controller gains for individual airspeed ranges. They state this is only to be used should satisfactory fixed gains not be found for the entire speed range. Previous versions of their autopilot produced step changes in gains as each airspeed threshold was crossed. Newer versions default to linear interpolation of the gains between each schedule, providing a smoother transition and enhanced flight performance.

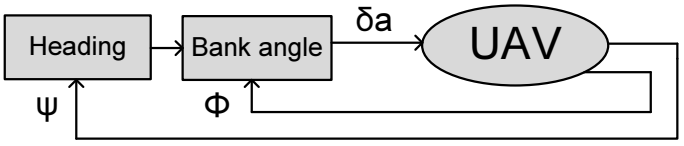
The controller architecture of the newest (version 2) Piccolo autopilot [149, 150] has changed considerably. The control laws are built around simplified fixed-wing



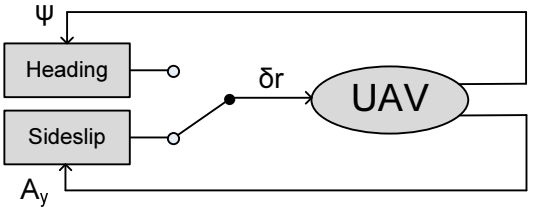
(a) Throttle



(b) Longitudinal



(c) Latitudinal



(d) Yaw

FIGURE 2.6: Simplified MicroPilot controller architecture

aerodynamic and kinematic equations. They are claimed to have high resolution gain scheduling built around the aircraft's stability derivatives, allowing a single set of base gains to remain optimal over a wide airspeed and altitude range. Control tuning shifts from an emphasis on correct gain values to adjusting the control model's aircraft characterisation parameters. No further details are available in the public literature, with Cloud Cap Technology stating the new control laws are proprietary information. The updated user manual [149] suggests an inner and outer control loop concept remains however, with an outer loop (such as bank angle) commanding an inner loop (such as roll rate).

This review of controller architecture literature has shown multiple controllers with dynamic path links provide a flexible autopilot capable of adjusting to specific mission phases. However, the increase in controller count inevitably leads to a

longer gain tuning processes and greater potential for control instability. The success of simpler architectures in achieving the basic aim of controlled flight suggests the first phase of autopilot development has no need to incorporate the flexibility of the Piccolo and MicroPilot architectures. Though specific mission phases such as catapult launch and autonomous sea landing will be required by the custom NOC,S UAV, they will not be performed by the test aircraft.

2.7 Path Tracking

The ability of the UAV to track a desired path is a high-level control requirement necessary for the majority of missions. While all UAV's reviewed have the ability to navigate to a waypoint by commanding ground heading to point at the target, this technique rarely results in the most efficient track between waypoint pairs - particularly in the presence of wind. A lateral tracking technique is a common solution, and is the primary tracking algorithm for the Piccolo series [33, 65], MicroPilot [57] and custom autopilots in use by the Indian Institute of Technology [40] and some Brigham Young University Zagi flights [32]. The algorithm used by the Piccolo was originally developed specifically for the Aerosonde UAV [4, 33].

The lateral tracking algorithm itself is relatively simple, and UAV-specific implementation is detailed by several sources [4, 33, 65]. With reference to Figure 2.7, cross track error a is calculated as the shortest distance between the UAV and "ideal" line connecting source waypoint W_S to target waypoint W_T. This distance is a line between UAV and its intercept C with the ideal line, such that it runs perpendicular to the ideal line. The Indian Institute of Technology autopilot then determines the bearing of W_T from the UAV and adds an offset directly proportional to a, and commands this heading [40]. Most other implementations project a further line from the intercept point C for a fixed distance b along the ideal line, and command a heading ψ_c towards the end of this.

The length of b is generally set to the aircraft's minimum turn radius at cruise speed, which may be easily estimated [33]. Shorter values tend to result in instability at the tracker commands impractical heading rates, whereas longer values lead to slower convergence on the path segment. The line length can also be adjusted as a becomes smaller than the minimum turn radius to ensure optimal tracking. King [33] claims several propriety non-linear scaling algorithms are used by Cloud Cap Technology. In an earlier paper Niculescu [4] describes the Aerosonde's (known to

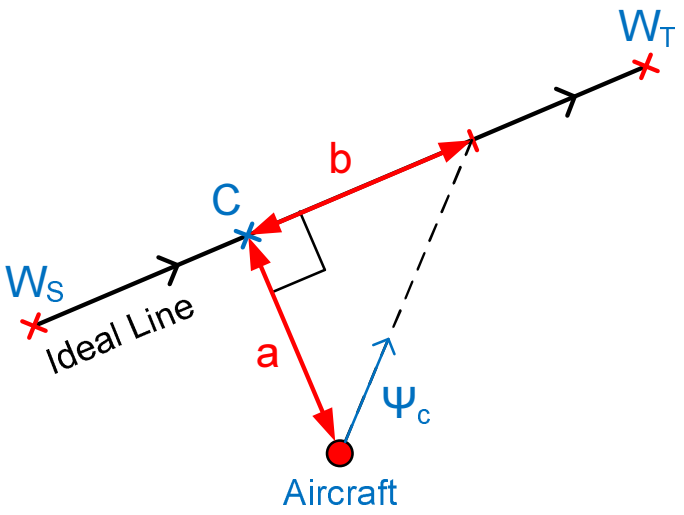


FIGURE 2.7: Lateral tracking

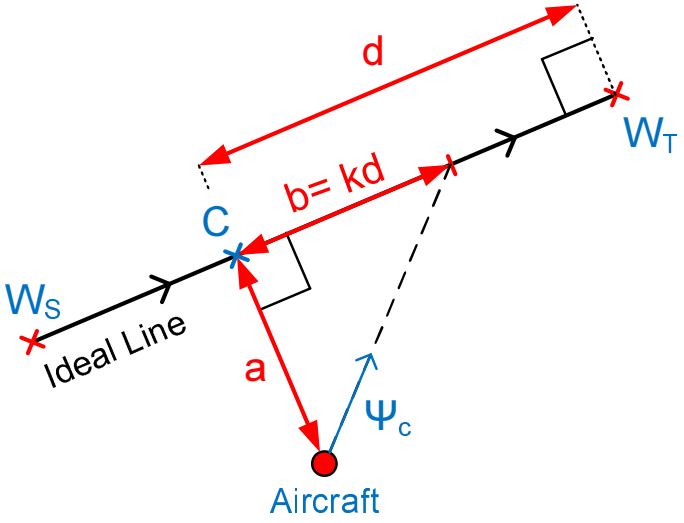


FIGURE 2.8: Aerosonde lateral tracking [4]

use the Piccolo autopilot [62]) lateral tracking algorithm. With reference to Figure 2.8 the distance b is set as a fraction k of the total distance d from intercept C to target waypoint W_T, such that $b = kd$. The commanded ground heading ψ_c is set towards the end of b as normal. Since d takes into account the distance to W_T, k sets the balance between convergence on the ideal line and direct line-of-sight approach to W_T (Figure 2.9). With $k = 0$, ψ_c aims at C. With $k = 1$, ψ_c aims at W_T. When $k > 1$ the UAV track runs increasingly parallel to the ideal line until almost perpendicular with W_T. Difficulties were experienced with the algorithm during high winds (leading to persistent unrecoverable turns), and an alternative algorithm needed to be used under such conditions (the switch condition is not presented). This algorithm requires knowledge of the wind bearing and does not use the k convergence factor. Two additional parameters are required that are

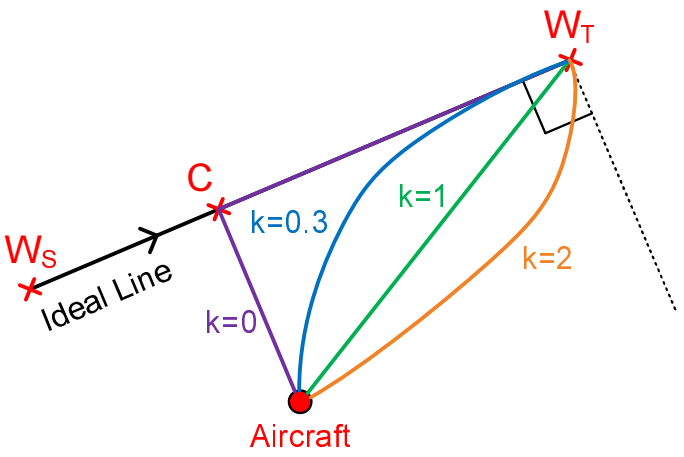


FIGURE 2.9: Approximate aircraft track for varying k using Aerosonde lateral tracking [4]

manually tuned to provide acceptable pointing to wind and line convergence for the aircraft's dynamics. The simplicity of the Aerosonde tracking method permits a high frequency update on a low-power processor. However, its inefficient handling of high winds and the requirement to calculate the wind vector are disadvantages. The proposed O-Navi autopilot hardware (Subsection 2.3.2) does not have an on-board compass, making wind vector calculations particularly difficult. Some Aerosonde aircraft also did not have an on-board compass. Holland et al [5] claim they used a proprietary algorithm that estimated the wind vector in 10–30 seconds of GPS readings and an S-turn manoeuvre. While a similar algorithm could be implemented, it cannot fully compensate for the lack of compass due to the path tracking disturbance caused by the S-turns necessitating a particularly low frequency update.

Bayraktar et al [45] describe a method of orbiting a single waypoint using lateral tracking. The intercept point is calculated between the orbit circle and a line from the UAV to the waypoint. A tangential line is created at this point, and lateral tracking is employed to align the UAV to this. A new tangential line is calculated for each tracking update, resulting in the desired orbit.

Patcher et al [151] use an unscented Kalman filter to estimate a constant wind direction and heading during a variety of UAV manoeuvres. Their UAV has a GPS but no onboard compass. The observability of the solution is shown to vary according to manoeuvre type, with straight and level flight resulting in only a partially observable state given the lack of body-frame compass heading. The presence of at least one steady coordinated turn however provides a completely observable state. Due to the online estimation algorithm this solution does not require the

specific S-turn manoeuvre used by some Aerosonde aircraft, providing at least one significant turn is performed during standard path following. The use of the unscented Kalman filter is however most likely to require greater computational resource than the Aerosonde technique.

Bhatia et al [152] and McNeely et al [153] discuss the use of Dubins paths [154] for UAV path generation. Dubins paths consist of circular and straight line sections calculated to navigate through predetermined waypoints covering a minimum distance given a maximum heading rate constraint. Similarly, required heading rate values (adjusted by the path's arc radii) can be generated to provide a commanded total path track distance. This allows multiple UAV's at different locations to synchronise their arrival time at a particular waypoint. Bhatia et al develop a high-level feedback control scheme to command a Kestrel autopilot's bank and airspeed and successfully demonstrate the navigation of a Dubins path, even though the Kestrel itself has no built-in Dubins functionality. McNeely et al extend the Dubins algorithm to allow for wind, and would be of particular interest should Dubins paths be adopted. Their extensions show Dubins paths still generate minimal distances under a general time-vary wind vector, though the authors emphasise the importance of taking wind into account for any Dubins implementation.

Brigham Young University also implement Dubins path generation [2, 155], though they do not refer to Dubins by name. Instead, they term their algorithm extremal tracking based on the use of step heading rate commands. It is observed by [2, 155] that a practical aircraft has a finite heading rate and as such would be unable to make the instantaneous changes in heading required at the corners of the ideal path. A "smoothed" path is therefore generated whereby the corners are replaced by circular segments. These segments have a radius corresponding to the minimum turn radius of the aircraft at cruise speed. As such, the new path would be ideally followed using only heading rate commands of zero, or clockwise and anticlockwise turns at maximum rate (extremal control). An algorithm is presented to calculate the turn points that would ensure the total distance travelled equals the perimeter of the initial ideal path, corresponding to the minimal distance ability highlighted during the Dubins path introduction. The algorithm may be varied to ensure all waypoints are directly flown over (as opposed to the nominal behaviour of being orbited). While this "smooth" path has attractive properties and is more feasible than the original path of straight lines only, it is still not possible to follow precisely due to the inability of the aircraft to make the instantaneous heading rate transitions assumed. To compensate for this, and for external disturbances and arbitrary start locations, a secondary algorithm is included in the Brigham

papers that guides the aircraft towards the extremal path. This tracking algorithm is shown to be asymptotically stable, though is only applicable to straight line segments. A reasonable degree of computational complexity is required for this tracking method, including the determination of circle and line intercepts and 4th order Runge-Kutta solving. The inability of real aircraft dynamics to perform the step response in heading rate required by Dubins paths are also highlighted by Shanmugavel et al [156].

An additional tracking method also developed at Brigham Young University, vector field tracking, is capable of both straight line and arc following [31]. The commanded trajectories are based upon field line vectors, and may be computed relatively simply. Of particular interest is the emphasis on stability under high wind conditions, confirmed both analytically and through practical flight demonstration. The ability of this algorithm to follow their extremal path generation technique (Dubins path) is also useful.

A number of complex algorithms, such as dynamic path generation for terrain following, moving object avoidance and vehicle tracking, are present in the literature. However the considerable computing resources and development time required for their implementation, coupled with a lack of immediate application make them unsuitable for this project.

2.8 Simulation

2.8.1 Aircraft Dynamics

Cloud Cap Technology claim it is critical that an accurate model of the aircraft is flown in simulation before actual test flights. In addition, hardware-in-the-loop testing is said to be a powerful technique in minimising development time and cost. A key benefit of such simulation is claimed to be the ability to tune the controller gains to an acceptable degree prior to the first flight [157]. Perhaps unsurprisingly, good support for such simulation is provided by the company through both packaged software and documentation [61]. The majority of academic literature reviewed citing their use of the Piccolo autopilot also mention the steps taken for aircraft characterisation and hardware-in-the-loop simulation using this software, in particular [33, 35]. A stand-alone simulator application [158] is supplied by Cloud Cap Technology that interfaces directly with the Piccolo hardware via

standard USB and CAN interfaces, linking it to the simulated flight environment. A later addition provides software-in-the-loop simulation, whereby the Piccolo autopilot firmware is executed directly on the simulation PC – negating the need for the autopilot hardware to be present.

For a useful simulation the flight characteristics of the test aircraft must be accurately obtained. The user's aircraft model may be created by entering linear and non-linear aircraft parameters into Cloud Cap's standard aerodynamics model. For aircraft designs that fall outside of the "standard model", Cloud Cap recommend the Athena Vortex Lattice (AVL) software [159] by Prof. Mark Drela and Harold Youngren. Cloud Cap provide a specific guide [160] to using AVL to produce models that are compatible with their Piccolo simulator. The guide itself cites a conference paper [161] where AVL has been used to successfully simulate an aircraft and tune the Piccolo autopilot's gains. In [161] minor modifications are made to the AVL coefficients following feedback from human pilots who had previous experience of flying the aircraft under test. This was achieved through simulation alone by allowing the pilots to "fly" the aircraft model using the Piccolo simulation software and assess its accuracy. Earlier examples of the use of AVL for aerodynamic modelling of UAV's are found [60, 162–164], though in all but the last case the study did not go beyond simulation. In [162] the AVL-generated coefficients are used in a custom aircraft simulation developed primarily to test robust control strategies. In [60] AVL is briefly mentioned as an alternative model generation technique used to compare with their X-Plane model (discussed shortly). The results of the comparison are not discussed, except to highlight the difficulties of assessing the accuracies of simulated coefficients from real flight data. [163] states AVL excels at stability and control derivatives (though the latter conflicts with the findings of [161, 164]) though the force predictions were not expected to be as accurate for small vehicles. Two custom UAV designs are modelled at MIT using AVL for aerodynamic modelling [164]. The controller design for the larger aircraft was entirely based on AVL results, whereas the smaller aircraft had access to a wind tunnel. Useful comparisons are drawn between AVL and wind tunnel data for the latter design. Since the fuselage had not been modelled in AVL, the absence of wake turbulence lead to a greater elevator deflection coefficient than that observed in the tunnel. Control surface deflection coefficients were in general overestimated mostly due to the aerofoil thickness not being considered by AVL [164]. Similarly high AVL control surface deflection coefficients were also found in [161] though this was attributed to gaps around the physical aircraft's control surfaces' hinges (improved in later designs). Both studies suggest care should

be taken when assessing an AVL model's response to control surface deflection, particularly given the importance of these coefficients during simulated controller gain tuning [161]. In general however, the AVL model's behaviour appears to have been within acceptable limits in all cases where comparisons to the true aircraft had been made.

MicroPilot does not provide simulation software or the ability to interface the hardware to a computer for simulated aircraft control [57]. An attempt is however presented by Mana et al [165] to enable hardware-in-the-loop testing of an older MP2000 autopilot, purchased with no sensors. An aerodynamic model for the test aircraft was created using the Aerosim blockset [166], running on Simulink in MATLAB. Some model parameters were measured directly from the airframe, and a minimisation function was used to adjust those remaining to minimise the difference between model data and real flight test data [165]. Though the resulting dynamic model was considered good enough for controller tuning, some notable discrepancies were observed. To provide actuator inputs to the model during hardware-in-the-loop testing, the pulse-width modulated (PWM) signals from the MP2000 were fed to a digital acquisition board, and Simulink converted the board's output to the equivalent control surface deflections. Output from the model, in the form of simulated sensor readings, were fed back to the autopilot in the appropriate form via two acquisition boards, a serial interface and some custom PCB's. Testing of the system was still in progress during the publication of this paper, and no follow-up is found in the literature. While simulation support is expected to be better integrated with the custom autopilot, the aircraft classification techniques and simulation environment remain of interest.

The Aerosim blockset is also used as a simulation environment by the Queensland University of Technology (QUT) [167]. The X-Plane flight simulator [168] had previously been used to model the aircraft dynamics, though a number of advantages are cited following the change to Aerosim. Firstly, the Aerosim blockset was capable of running in real time and at the same rate as the extra autopilot simulation blocks created in Simulink. A number of synchronisation difficulties had been experienced by X-Plane. Secondly, the range of simulated aircraft and avionics failures in X-Plane was limited, with no method of customisation. Thirdly, accurate modelling of the sensors was not performed by X-Plane. QUT used RS-232 to interface their custom autopilot for hardware-in-the-loop, with all sensor and actuator data being passed through this serial link. RS-232 bandwidth constraints lead to the use of Ethernet (PC) and CAN (hardware board) interfaces. A separate software module on-board the autopilot interprets the serial data and provides

a transparent interface with the main autopilot code. QUT state hardware-in-the-loop simulation "proved invaluable", particularly in fixing bugs in the ground station interface. Entire simulated missions have been flown using this technique. The Aerosim blockset contains the Aerosonde and Navion as a pre-configured example aircraft. These default models have been used for proof of concept studies such as [169, 170] and as a basis for custom airframe parameters in [165]. The blockset is also used in simulation of the Stingray UAV [54] though the aircraft dynamics themselves used the MATLAB flight dynamics and controls toolbox, with Aerosim providing the transform from body motion to latitude and longitude.

Though a number limitations with the X-Plane simulator for aerodynamics modelling were noted in [167], this technique has been adopted by some projects [171–173], particularly for helicopter simulation used by undergraduate students. In each of these papers successful hardware-in-the-loop simulation is reported using the X-Plane network packet interface.

JSBSim [174] is also referenced in the literature as an aircraft dynamics simulator for UAV's. The Institute of Scientific Research have performed successful hardware-in-the-loop simulations using this simulator combined with several custom bridging applications for interfacing with Simulink and the hardware board [175]. The custom hardware interface application presents simulated sensor data over a RS-232 serial link to the autopilot. As previously reviewed, this board has a modular design with the sensor acquisition board running independently to the flight control board [39]. Since communication between these modules is also performed using RS-232, the data provided by the interface module appears identical to that received by the real sensor board. This provided a simple method of realistic and transparent handling of simulated data by the hardware autopilot. The JSBSim aerodynamics core itself was accessed by the custom bridging software via network communications with the FlightGear simulator [176]. While the real-time limit of the FlightGear simulator was useful for the hardware interface, it was expected such simulation could be performed much faster for the additional Simulink autopilot model. As a result, a direct interface between Simulink and JSBSim is proposed for future work, permitting the rapid simulation of UAV missions [175]. The paper concludes that the development of the JSBSim-compatible flight model of their aircraft took the most effort, primarily due to difficulties calculating the necessary model coefficients.

While the majority of papers reviewed use third-party software for aerodynamics

modelling and simulation, a complete hardware-in-the-loop system has been developed for the DragonFly UAV [49]. A comprehensive aerodynamics and sensor simulator was produced in-house and run on two Windows PC's. Communication between the PC's and the DragonFly hardware is performed using ethernet network packets in a similar manner to other techniques reviewed. An overview of key aerodynamic and control algorithms implemented by the simulation software is provided by the paper. A further custom aerodynamic modelling approach using neural networks is presented in [177]. The paper focuses on the online identification of aircraft parameters, and claims a good prediction for a three degrees of freedom UAV. Such parameters would be fed in real time to onboard adaptive controllers. A complete six degrees of freedom model is identified as future work. Caution was raised regarding the computational complexity of the current algorithm due to the inversion of potentially large matrices. Such resource is unlikely to be available for real-time calculation by the autopilot processor envisaged for this project. Some value in this technique may come from offline parameter identification from prerecorded flight data, though concern remains given the incomplete model parameter set currently offered.

The literature review highlights the importance and widespread use of simulation techniques in autopilot development. Simulation is shown to be particularly important in controller gain tuning in a risk-free environment prior to real flight tests. AVL is suggested as a viable tool for determining an aerodynamic model of the test aircraft without access to a wind tunnel. Autopilot hardware can be linked to a simulated environment via an RS-232 interface, with the ability to "fly" the aircraft model using autopilot software identical to that used in the real system considered a great aid in rapid and reliable software development.

2.8.2 Visualisation

A visual display of the aircraft during simulated flight manoeuvres is documented by a number of papers. This feature allows the accuracy of the simulated aircraft flight dynamics to be qualitatively assessed by manually flying the aircraft in a virtual reality environment. The Piccolo simulator is designed to interface with FlightGear [158] though since aerodynamic modelling is provided by the Piccolo software, FlightGear's optional JSBSim module is not used. The Aerosim blockset also contains a FlightGear interface module [166]. As a result, many projects using

Piccolo or Aerosim aerodynamic simulators use FlightGear for aircraft visualisation [33, 38]. The Institute of Scientific Research project also use FlightGear for visualisation as well as aerodynamic modelling [175].

The Piccolo ground station software is also capable of being interfaced via network packets with Microsoft Flight Simulator for real-time and stored flight data visualisation, as demonstrated in [45]. Visualisation of simulated missions is also performed by Microsoft Flight Simulator for the custom UAV system developed in [44]. A Microsoft Flight Simulator interface is also provided by the Aerosim blockset.

McManus et al [167] describe their replacement of X-Plane with Aerosim for aerodynamic simulation, though they retain X-Plane for visualisation. This is likely due to previous development time spent on their custom X-Plane interface software, though the lack of switch to Aerosim's built-in FlightGear or Microsoft Flight Simulator visualisation is surprising. Other projects previously reviewed using X-Plane [171–173] all retain this software for visualisation.

The literature shows real-time visualisation of the simulated aircraft is possible using readily available software. The use of visualisation with a human test pilot to assess the accuracy of the aircraft in flight is considered of particular interest given the importance placed on a realistic simulation model.

2.9 Conclusion

This chapter has identified a range of applications for a UAV in the field of oceanographic research. Such a system would be used predominately for high resolution passive observation, though some physical interaction in the form of sea spray sampling and deployment of miniature buoys is cited by the literature. When combined with a review of potential sensors, a specification for an oceanographic research UAV has been formed. Key requirements were long range and endurance airframe capable of accommodating a sufficiently large and heavy payload to maximise the range of supported scientific instruments. The inherent risk of the oceanographic operating environment and in particular sea-based landing means a low cost, potentially disposable, UAV is required.

Following a review of commercial UAV's, no suitable system was identified. The low cost and support for custom payload integration were commonly missed specifications. An additional NOC,S project would instead focus on the development

of a custom airframe specifically tailored to NOC,S mission requirements. Since the research objectives for this project focus on autopilot development, practical flight tests would be performed using a COTS model aircraft due to its rapid acquisition and readily available spare parts. The autopilot must be low cost and low power to meet the UAV specifications. A suitable commercial system was not found in the literature, primarily due to the high cost associated with the use of embedded proprietary autopilot software. A review of commercial and academic autopilot hardware components suggested the purchase of a hardware-only solution and in-house software development was the most realistic option for meeting the specifications.

Algorithms for determining and representing aircraft attitude were reviewed for implementation by the in-house software. An extended Kalman filter is used by the majority of autopilots and its operation well documented by the literature. Quaternion attitude representation was presented as an alternative to Euler angles with advantages in both computation speed and stability. The literature suggests the PID algorithm as a popular controller implementation in both commercial and academic autopilots. The PDF controller was suggested as a novel alternative for UAV control, with similarly low computational requirements and robust performance in equivalent applications. Both algorithms require gain tuning, with a manual trial and error technique being most favoured in UAV literature.

No suitable path tracking algorithm was found in the literature that combined a simple algorithm with an efficient and robust response to high winds. The modified lateral tracking algorithm used by the Aerosonde UAV had the greatest potential though the requirement to calculate the wind vector was a disadvantage, particularly given the lack of compass in the chosen autopilot hardware. Autopilot development will therefore include research into novel path tracking algorithms.

The need for a realistic simulation environment was also emphasised in commercial and academic literature. The realism of the aircraft model can be verified by the manual test pilot through the use of visualisation software. With an accurate aircraft model, manual tuning of the controller gains can be performed in a risk-free environment with minimal discrepancy when used by the real aircraft. Interfacing the real autopilot software with the simulation is also cited as a important step in more robust code and shorter development time.

Chapter 3

System Design

3.1 Simulation

3.1.1 Choice of Software

MATLAB® has been selected as the principle simulation environment, due to its potential for rapid algorithm prototyping and data analysis. Use of MATLAB's Simulink®[126] encourages a modular design enabling a clear visual overview of the system in a schematic format. Individual Simulink blocks may also be created directly from C code (as S-Functions), permitting code used by the actual autopilot hardware to be interfaced and tested directly within the Simulink model.

Following the literature review, the free (for academic use) Aerosim blockset by Unmanned Dynamics was selected as the aerodynamic modelling component for Simulink that is particularly suited for UAV's. While the mathematics of a non-linear six degree of freedom aircraft dynamics are well established, a custom implementation of such a model for this research would be time consuming and potentially error prone. As such, the use of this third-party blockset represents a considerable saving in simulation development time.

3.1.2 Aircraft Characterisation

Initial testing and development of the simulation environment was performed using one of the default aircraft models provided by the Aerosim package. While this allowed for rapid construction and testing of the environment, a custom aircraft

model was eventually required. Such a model should closely represent the real test aircraft to enable a more accurate assessment of the autopilot's flight performance. In particular, a more realistic aircraft model leads to the more appropriate selection of controller gains during simulation. Due to time restrictions and risk of damage to the real airframe and avionics, it is preferable to begin any practical flight tests with as accurate choice of controller gains as possible.

To create a custom aircraft model in Aerosim, over thirty aerodynamic coefficients are required. While standard textbook equations exist for determining some of these coefficients from readily available aircraft parameters, other desired coefficients require a greater depth of aircraft analysis. Such coefficients may be obtained from computational fluid dynamics (CFD) software or wind tunnel data, though both are time consuming in both initialisation and analysis and typically require regular access to specialist facilities. The use of Athena Vortex Lattice (AVL) software as an alternative was identified during the literature review. Given AVL only requires relatively quick and simple measurements of the test aircraft, and the software's rapid calculation of all necessary aerodynamic coefficients, this approach has been adopted for this project.

AVL requires a specially constructed text file to represent the aircraft. This file contains the geometry of the test aircraft's main wing, horizontal and vertical stabilisers and the shape of their cross section. The size and location of any control surfaces built into these structures is also provided. Fuselage information is not required. These measurements, along with the aircraft's centre of gravity and approximate airspeed, are all that AVL requires to generate the desired coefficients. The aircraft's geometry loaded in AVL is shown in Figure 3.1.

In addition to the aerodynamic coefficients, the test aircraft's moments of inertia are required. These were calculated by breaking the airframe and contents down into a collection of primitive objects; cylinder (aligned along x, y or z axis), cuboid or a point mass. The moments of inertia for each primitive could then be calculated by standard textbook equations. This data, along with each primitives' mass and relative position were entered in a second text file as an input to AVL. This allowed AVL to calculate the overall aircraft moments of inertia in terms of I_{xx}, I_{yy}, I_{zz} and I_{xz} and the centre of gravity as required for the Aerosim model. Comparison between the centre of gravity calculated by AVL and that experimentally measured on the test aircraft was accurate to within 3mm, validating at least the primitives' mass and locations. The AVL results and input file listings are included in Appendix A.

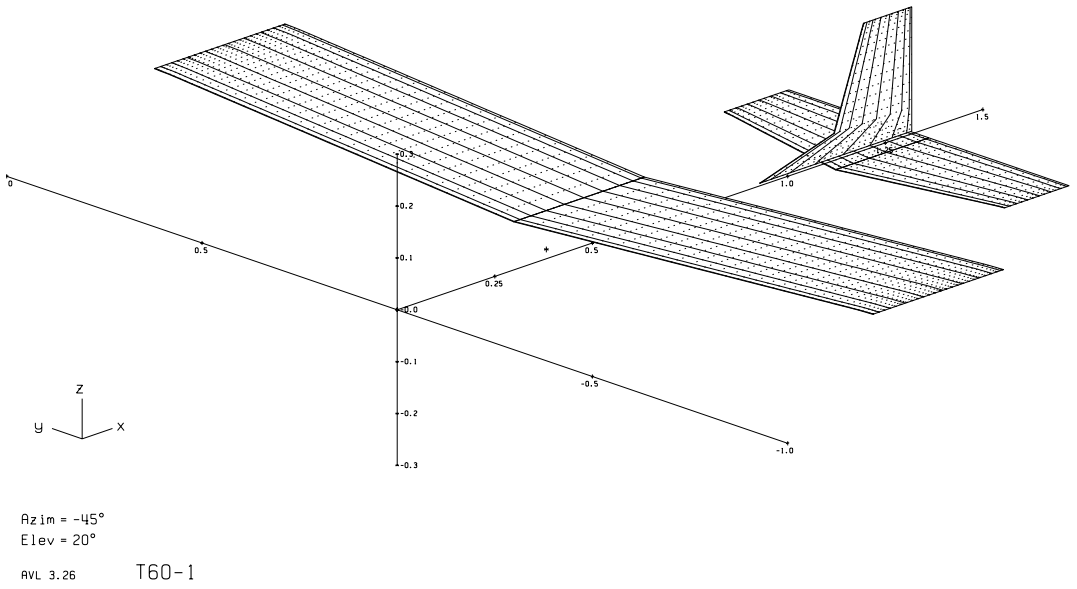


FIGURE 3.1: Test aircraft geometry loaded in AVL

AVL cannot however provide the propeller and engine data also required for the Aerosim aircraft model. To reduce model creation time it was believed this data did not need to be as accurately matched to the custom aircraft as the aerodynamic and inertia calculations, since they would be less critical in controller tuning. As a result, a simple online utility JSBSim Aeromatic [178] was used to generate approximate propeller coefficients. The utility only required the maximum engine power and RPM (provided by the engine manufacturer) and the fixed-pitch propeller diameter. A more advanced analysis tool, JavaProp [179], was used at a later time for another aircraft. In retrospect, such a tool could have been used for the test aircraft model to potentially create more accurate data, but at the time the simplicity of the Aeromatic utility led to its initial selection.

A similarly rapid but less accurate technique was adopted for the engine data. The data chosen was simply that provided by the Aerosim blockset for the Aerosonde engine, deemed close enough for initial testing. The RPM levels were scaled to approximate that expected of the test aircraft. Fuel flow data was left unchanged since this was not of interest at this stage of the project and could be safely ignored during simulation. Upon simulation of the test aircraft, it was found the throttle setting for straight and level flight at a given airspeed was reasonably similar to that experienced during practical flight tests. The engine data was therefore considered adequate, particularly as a poorly tuned throttle controller alone would be unlikely to place the aircraft in immediate risk.

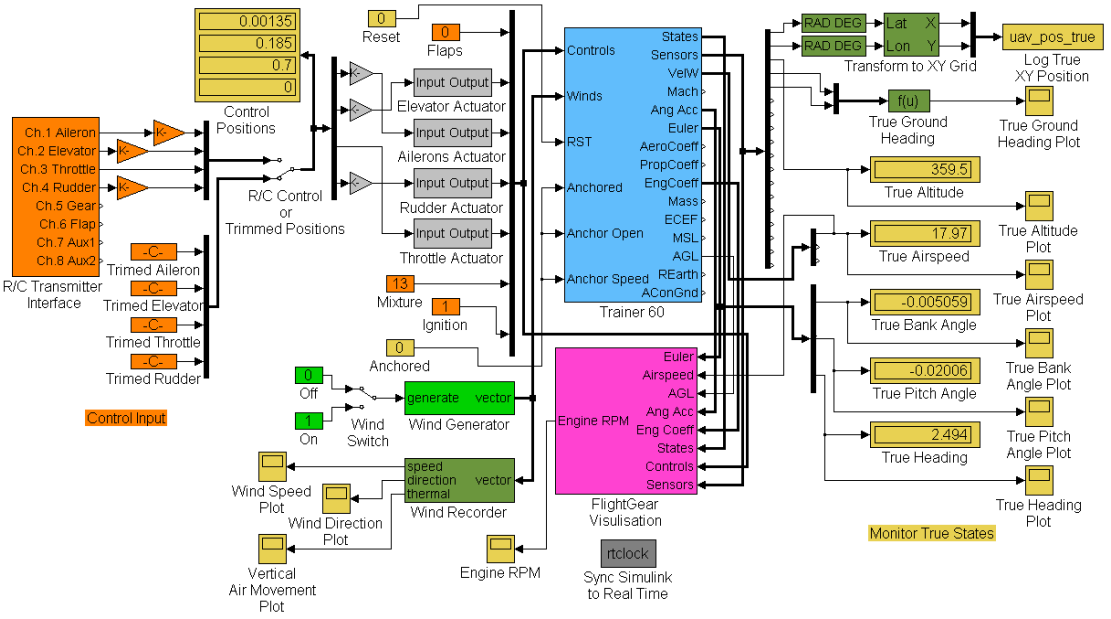


FIGURE 3.2: Simulink model of test aircraft in flying environment

3.1.3 Basic Simulation Model

A Simulink representation of the test aircraft flying in the simulated environment is shown in Figure 3.2. The test aircraft can be "flown" manually (via a standard hobby transmitter attached to the PC) or by setting the control surfaces and throttle to default values. The aircraft state can be monitored in real-time by the numeric values displayed in the model, viewing the data plots created by Simulink, or by viewing the aircraft in a three-dimensional virtual world visualised through a link to the FlightGear[176] simulator (Figure 3.3). In Figure 3.2, control input blocks are orange, state monitoring blocks are yellow, FlightGear link is magenta and the test aircraft model itself is blue. The dark grey block is included to slow down the simulation to real time (particularly useful when visualising using FlightGear) and light grey blocks represent the aircraft's actuators. These generic actuator models are included with Aerosim, and have been customised to provide the limited actuating rate and output delay expected from the real system.

The "Transform to XY Grid" block has been created to convert the aircraft's latitude and longitude to Cartesian Y and X displacement (respectively) in metres from a predefined origin location. The equations used are sufficiently simple to permit rapid conversions by low-powered processors, whilst retaining an acceptable level of accuracy for long-distance flight planning and are based upon those presented in the American Practical Navigator[180]. Converting latitude and longitude into this format simplifies tracking algorithms and map overlay functions

FIGURE 3.3: Simulink model execution visualised in FlightGear

(due to the uniform scaling of the grid and use of the SI metre unit) and is considerably easier to relate to by the human operator. As a result, an identical conversion is used internally by both autopilot and ground station software. Waypoint positions are only stored as XY grid locations. This has the additional advantage of not tying them to specific geographic points; rather the stored waypoint pattern is mapped onto the desired geographic area simply by moving the conversion's datum point.

The ground heading block takes the inverse tangent of northerly and easterly velocities provided by the aircraft model to produce ground heading. This is the same technique used to calculate ground heading from GPS velocity information. The wind generator block (light green in Figure 3.2) internally uses a random number generator block for each wind axis (with the desired mean speed and variance for the environmental conditions). A moving average filter is passed over each sequence to provide smooth transitions between velocities.

3.1.4 Advanced Simulation Models

Additional blocks were inserted around the basic model to simulate realistic sensors and autopilot functionality. Figure 3.4 illustrates the generation of sensor data of the same format presented to the autopilot by the real hardware. True sensor information generated by the Aerosim aircraft model block enters from the left. A series of dark-green blocks scale, bias and saturate the simulated true values to the

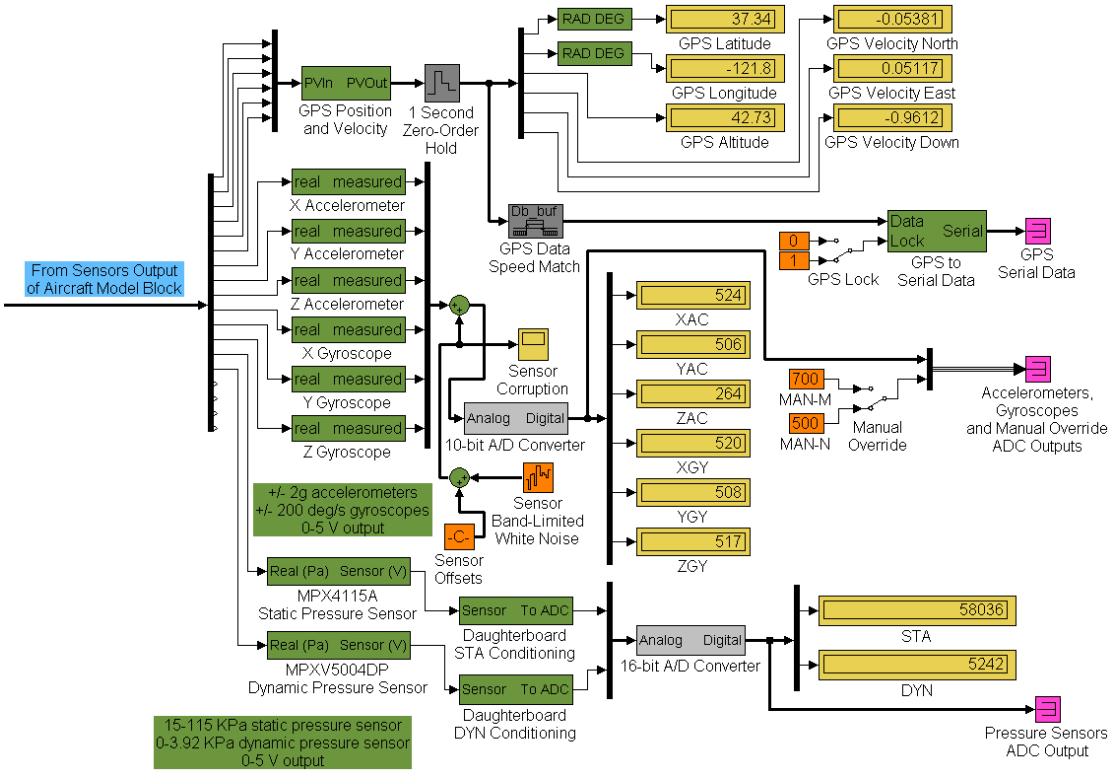


FIGURE 3.4: Simulink model of realistic sensor hardware

output voltages specified by the real sensors' datasheets. For the pressure sensors, additional signal conditioning is performed to their voltage outputs to mimic that provided by the custom daughterboard hardware (Section 3.2.2). White noise and bias effects are added to the accelerometer and gyroscope sensors. Corrupting these sensor readings by a known amount is of particular use when accessing the performance of the autopilot's Extended Kalman Filter (Section 3.3.5). In the final stage the output voltages are converted to 10-bit (for the accelerometer and gyroscopes) or 16-bit (pressure) values using an analogue to digital convertor (ADC) block provided by Simulink.

Aerosim includes a block to simulate GPS output, including realistic transport delay and Gauss-Markov noise correlation. Since this provides a continuous output, a sample and hold block is included to provide the one second updates expected from the real system. A custom block converts this raw position and velocity information to a sequence of bytes sent through a serial link in the same form that the real autopilot receives this information. This serial data, along with the sensors' ADC readings, is ready for direct integration with the autopilot (at the magenta output ports in Figure 3.4).

Two representations of the autopilot were created during development. The first

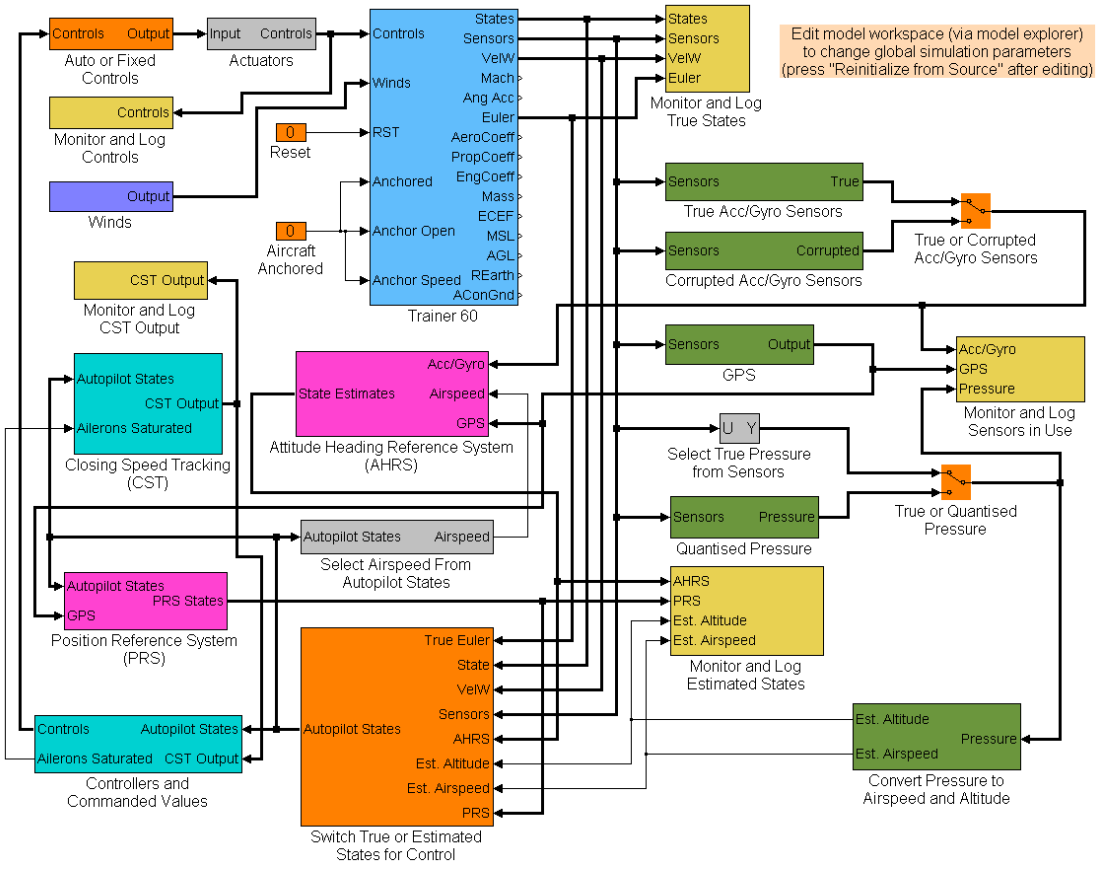


FIGURE 3.5: Simulink model demonstrating complete autopilot functionality in a realistic environment

form, primarily used during the first half of the research process, used a collection of interlinked Simulink blocks. This structure took advantage of the modular nature of Simulink, allowing the rapid construction of a basic system with clearly defined relationships between autopilot components. Individual blocks could be easily modified and improved with little interference to the rest of the system. Visual debugging, logging and graphing of all aircraft and autopilot states was simple to achieve using this framework. The ability to quickly compare true states directly alongside estimated states was essential during the development of the state estimation blocks. All update rates (such as those for sensors, controllers, filters and tracking algorithms) were simulated to match those provided by the real autopilot, though were adjusted at times to explore the simulated system's sensitivity to these rates. In addition, the inputs to the control and navigation blocks could be individually switched between true or estimated values. This allowed the performance of such blocks to be evaluated without the inaccuracies of state estimation. The top-level schematic of the first form autopilot and simulation environment is shown in Figure 3.5.

The second autopilot form was the entirely C-code based source code, compiled by external software tools and directly executed on the real autopilot's processor. While the design process was quite different, the ability of Simulink to create blocks from C-code using S-Functions and MATLAB's MEX functionality permitted a high degree of overlap between the two forms. In the first schematic block form, many blocks were actually written directly in C. This has the considerable advantage of being able to integrate and test the performance of C functions while running within the Simulink environment. Once satisfied with the behaviour and integration of this code block, the underlying C function could be transferred directly to the processor-ready autopilot source code with relative confidence.

As a final verification tool prior to practical flight testing, the main core of the full C-code autopilot could be run as a single Simulink block. A simple software wrapper was provided around the autopilot core functionality (performed by the processor) that formed the interface layer with the peripheral devices (sensors, GPS, wireless modem, external memory and servo switching). This wrapper had two operating modes, hardware or simulation, but maintained the same functional interface with the core autopilot code. In hardware mode, the wrapper would call upon device drivers and setup routines specific to the real autopilot hardware. In software mode, the wrapper took the form of a Simulink S-Function whose data was sent and received through the block's interface or from a file (in the case of the external memory driver). This allowed the core autopilot code to be embedded directly in a Simulink environment, yet remain unchanged when compiled and downloaded to the real hardware.

Since the ground station software communicates with the autopilot through a standard TCP/IP network link (Section 3.4.1), an additional interface block was added to Simulink to provide a two-way link between this software and the simulated autopilot. This allows the core autopilot code to be run under simulated conditions yet still communicate with the unmodified ground station software. This software-in-the-loop functionality has proven very useful for rapid and robust software development, both for the autopilot and the ground station software. In particular the ability to interface the ground station with a fully simulated aircraft allowed realistic evaluation of the station's graphical user interface (GUI) without the constrains imposed during a real flight test. Similarly, the autopilot code could be exposed to simulated flight conditions that would not have been possible to replicate with the actual hardware under laboratory conditions. The second form autopilot code is shown embedded in a fully simulated environment in Figure 3.6.

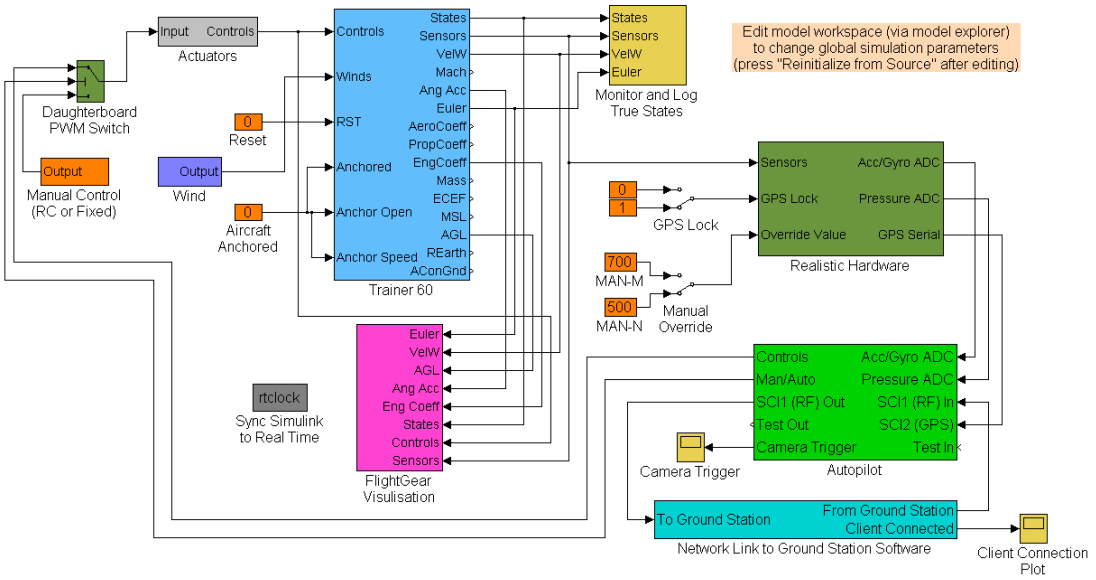


FIGURE 3.6: Simulink model with core autopilot code executing in a realistic environment

The autopilot code is executed on the desktop computer rather than on the autopilot hardware itself, hence software rather than hardware in the loop. This approach is found to fulfil the development advantages of hardware-in-the-loop testing identified in the literature review (Section 2.8) while offering a more convenient interface with the simulation software. The autopilot Simulink block always runs at the same rate as the simulation (and may also be paused) ensuring input and output data remain synchronised. Autopilot software changes are quickly incorporated in the block without the need to erase and re-flash external hardware memory. In addition, full debugging of the running autopilot code is possible using the same programming tools used to debug other native applications.

3.2 Hardware

3.2.1 Aircraft

The COTS aircraft chosen following the literature review (Section 2.2) is a Trainer 60 ARF model (Figure 3.7).

A 10cc Irvine 61 engine was installed in line with the recommended minimum engine capacity of the aircraft and following consultation with experienced model aircraft pilots. A digital transmitter and receiver combination was chosen for

(a) Test pilot readying aircraft for flight (b) Fitting and testing hardware on the bench

FIGURE 3.7: The Trainer 60 ARF model aircraft

use during manual control of the aircraft to minimise the impact of electronic interference from the autopilot hardware. Previous tests using an analogue receiver had highlighted a small but significant "flutter" on the actuators confirmed to originate from the autopilot. The probable source of this noise was identified through trial and error as disturbance to the avionics board ground level during communication between the processor and certain peripheral components.

A Pitot tube was attached to the underside of the aircraft's wing and connected via plastic tubing to the autopilot's dynamic pressure sensor. Ground tests showed relatively rigid plastic tubing was preferable to reduce unwanted pressure variation caused by movement of the tube, particularly during engine vibration. Static pressure is typically read from a small metal tube with an opening orthogonal to the airflow during flight. However, ground tests highlighted pressure disturbances from the wake of the propeller adversely affecting this reading. Improved readings were measured once the static tube was simply left open inside the airframe. The airframe has a number of gaps where outside air can rapidly equalise the internal and external pressures, maintaining an acceptable agreement with true static pressure whilst shielding the input from propeller wash.

The GPS, manual control receiver and ground station communication aerials were fixed to the outside of the airframe. A thin wooden panel was added to reinforce the aircraft's front bulkhead to which the nose wheel was attached. This provided additional support when landing with the increased payload weight.

3.2.2 Autopilot

Following the literature review (Section 2.3) a COTS autopilot-ready avionics board was purchased from O-Navi. The board provides a full complement of inertial and pressure sensors with integrated GPS and a 32-bit, 32 MHz Motorola MCORE processor with 32 kilobytes SRAM and 256 kilobytes program memory.

While the processor supplied with the O-Navi board was acceptable, subsequent trials revealed the need to incorporate a daughter board to add new or enhance existing peripheral functionality. This board was developed in-house and interfaced with the O-Navi hardware to provide a non-volatile memory, digital switching between manual and autopilot generated signals and higher resolution sampling of the pressure sensors.

The non-volatile memory permitted the retention of autopilot parameters and waypoint data once the power supply was disconnected. This allowed the autopilot to function immediately from startup without needing to receive such information from the ground station. The possibility to recover from a power supply interruption during flight is also introduced, though is not currently implemented. The ability for the autopilot to switch actuator control between manual and automatic in response to ground station commands was also felt an essential requirement. Finally, the 10-bit analog to digital convertors (ADC's) connected directly to the pressure sensors' output by the O-Navi board were found to be inadequate for altitude and airspeed resolution requirements. The daughter board replaced these with 16-bit ADC's and some basic signal conditioning. The reasoning for the conditioning and resolution upgrade is as follows.

The output voltage of both pressure sensors used is directly proportional to their input pressure [181, 182]. The autopilot uses Equation 3.1 [183] to estimate altitude h from static pressure, using the International Standard Atmosphere (ISA) at sea level to define temperature T_0 as 288.15 K, pressure P_0 as 101.325 kPa, lapse rate γ as -0.0065 Km^{-1} and gravity g as 9.80665 ms^{-2}. The specific gas constant R_d of dry air is taken as 287.053 $Jkg^{-1}K^{-1}$. P_{static} is the current static pressure (in kPa) read by the sensor. The subscript AMSL denotes altitude above mean sea level.

$$h_{AMSL} = \frac{T_0}{\gamma}\left(\left(\frac{P_{static}}{P_0}\right)^{\frac{-\gamma R_d}{g}} - 1\right) \tag{3.1}$$

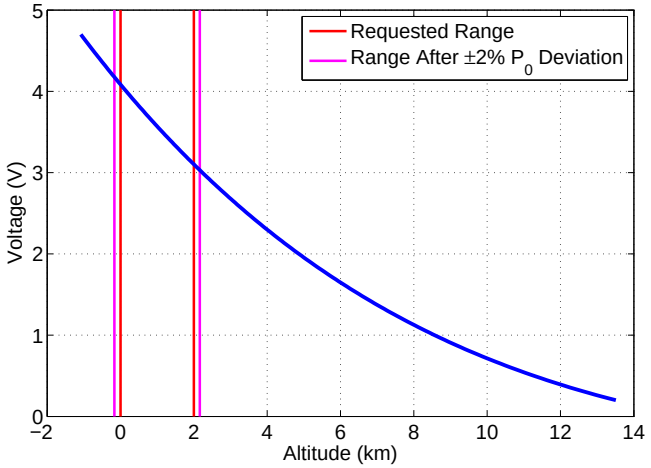


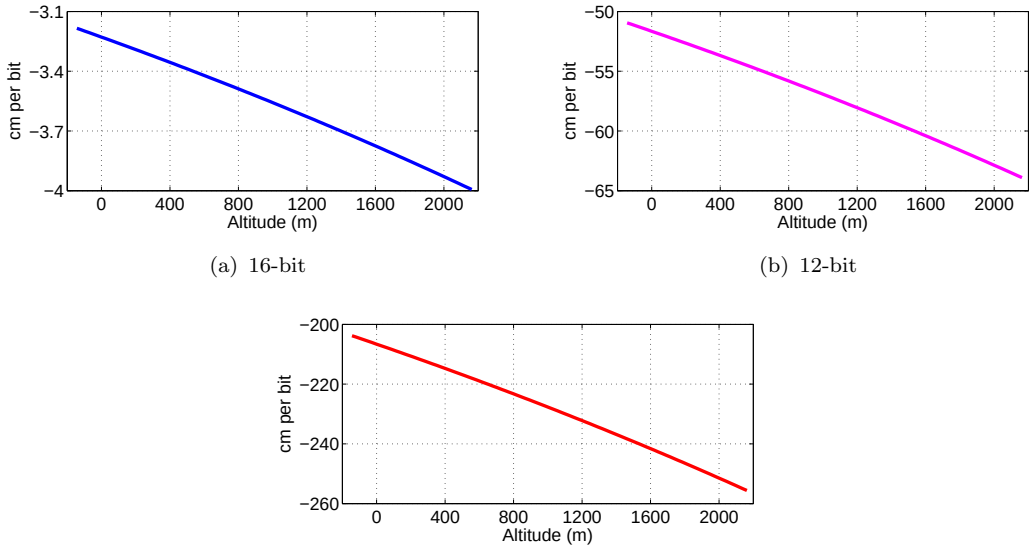
FIGURE 3.8: Static pressure sensor output voltage against altitude with desired range highlighted

The static pressure calculated by the autopilot will have a fixed resolution, set by the hardware's analog to digital convertor (ADC). The pressure sensor itself is capable of measuring 15 to 115 kPa, equivalent to 13.5 to -1 km altitude respectively. This range exceeds that expected for the missions envisaged, where 0 to 2 km altitude was considered sufficient. Assuming a 2% deviation in sea level pressure from the ISA standard, Figure 3.8 illustrates the sensor output voltage limits for the altitude range of interest. A simple gain and offset circuit was implemented on the daughter board to expand this region of interest to the full input range (0-5 V) of the ADC, thereby maximising the altitude resolution obtained. Since the relationship between pressure and altitude is nonlinear (Equation 3.1) the actual resolution obtained will vary with height as shown in Figure 3.9. A typical cruise altitude of 100 m would provide a minimum resolution of 208 cm using a 10-bit ADC, 52 cm using a 12-bit ADC and 3.3 cm for a 16-bit ADC. Given the output from any ADC was expected to include electrical noise equivalent to several quantisation steps, it is apparent only a 16-bit ADC could offer the resolution deemed necessary for accurate (<1 m) altitude control.

A similar situation and upgrade was identified for the dynamic pressure sensor. Indicated airspeed (IAS) assumes a fixed ISA air density of 1.255 kgm^{-3} and is calculated from dynamic pressure P_{dyn} (in Pa) as Equation 3.2.

$$V_{IAS} = \sqrt{\frac{2P_{dyn}}{1.225}} \tag{3.2}$$

The autopilot assumes calibrated airspeed (CAS) matches IAS. Given the typically



(a) 16-bit

(b) 12-bit

(c) 10-bit

FIGURE 3.9: Altitude resolution for 10, 12 and 16-bit ADC's

small airspeed discrepancy introduced by pitot and static tube position, installation and instrument errors this assumption is considered valid for the purposes of this research. Since CAS directly relates to the dynamic pressure acting on the aircraft's surfaces, this measurement is referenced to the airframe manufacturer's parameters such as best rate of climb and best glide angle. In this case such data for the Trainer 60 was not available, though CAS remains relevant due to its direct association with stall speed. While true air speed (TAS) is used for control and position estimation (Section 3.3.4) CAS will be used to assess the desired dynamic pressure range. The sensor itself is capable of measuring 0 to 3920 Pa, equivalent to 0 to 80 ms^{-1}. This range exceeds the expected maximum CAS of 44 ms^{-1} (85 knots) considered for the custom oceanographic UAV (which in turn exceeds that of the Trainer 60). Figure 3.10 shows the sensor output voltage limits of 1.0 to 2.2 V for the 0 to 44 ms^{-1} airspeed of interest. As with the static pressure sensor, a gain and offset circuit was added to the dynamic pressure sensor output to expand this region to the full 0 to 5 V input range of the ADC. The nonlinear relationship between airspeed and dynamic pressure is particularly apparent at low airspeeds (<5 ms^{-1}), were the CAS resolution is seen to markedly degrade in Figure 3.11. The maximum desired resolution was considered to be 0.1 ms^{-1} (0.2 knots) per bit, assuming some electrical noise as before. Figure 3.11(b) suggests a 12-bit ADC would be sufficient, since the resolution only falls below this threshold at speeds most likely below any aircraft's stall speed. However, since a 16-bit ADC was already selected for the static pressure sensor, it was felt the familiarity gained

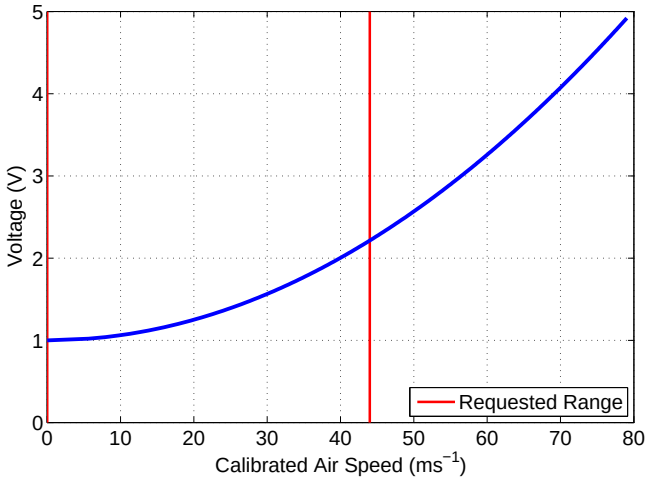
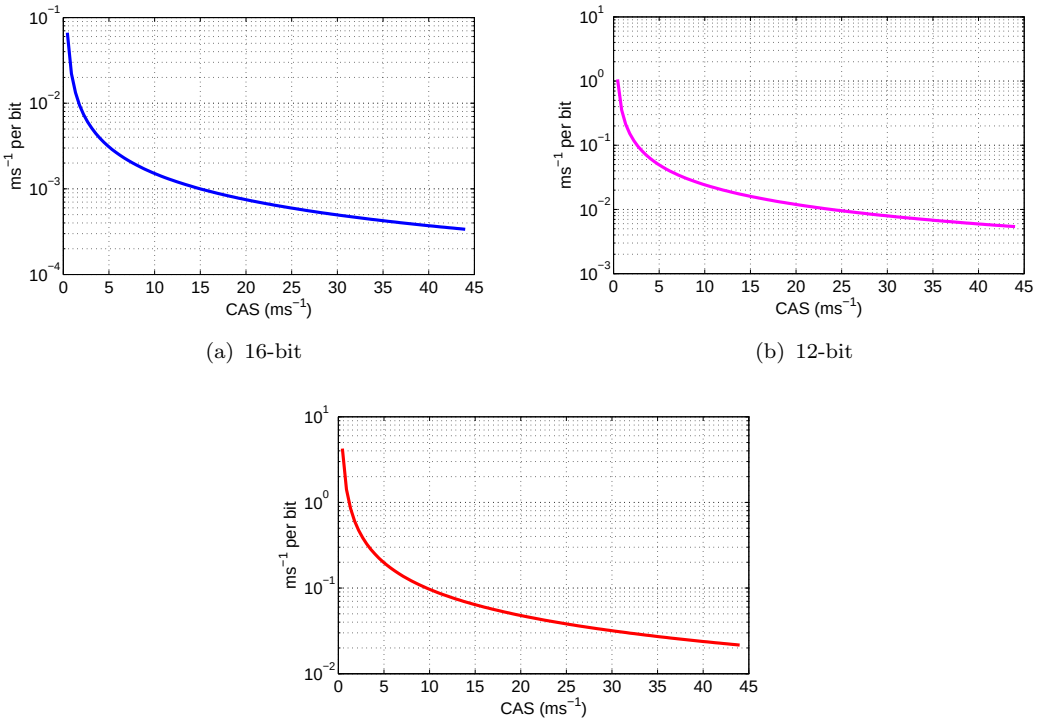


FIGURE 3.10: Dynamic pressure sensor output voltage against CAS with desired range highlighted



(a) 16-bit

(b) 12-bit

(c) 10-bit

FIGURE 3.11: CAS resolution (on logarithmic scale) for 10, 12 and 16-bit ADC's

during its use would offset the relatively small cost savings of selecting a new 12-bit ADC component. As a result, the conditioned dynamic pressure sensor output was also sent to a 16-bit ADC on the daughter board.

The complete timing data for the hardware autopilot board is provided in Appendix B. Flags were set in the code to provide a direct output to one of the board's external IO pins. These signals were monitored and timed using a digital oscilloscope. This technique was found to be particularly useful in code optimisation, particularly when the quantity of machine code generated by the compiler for a particular high-level C function was difficult to judge. In addition the timing analysis verified the autopilot remained capable of meeting its interrupt-driven hard deadlines. A key deadline was the 70 Hz main flight control update. The analogue low-pass filters on the O-Navi board and recommended processor setup expect the embedded sensors to be read at 140 Hz, triggered by an internal interrupt signal. The aircraft's actuators deflecting the control surfaces and throttle have a maximum bandwidth of 50 Hz, so controller updates at higher frequencies offer little practical advantage while increasing the computational load on the processor. To save the extra computational overhead of setting another internal interrupt, the main flight control update was therefore set to trigger on alternate 140 Hz sensor readings, resulting in the 70 Hz update.

3.2.3 Communications and Safety

The first form of the autopilot model (Figure 3.5) took advantage of the built-in user input and data plotting abilities of Simulink and MATLAB to set basic autopilot commands and analyse data. Clearly such functionality is not available to the hardware autopilot, and as such a wireless interface is required to custom ground station software.

The radio modem selected (MaxStream 24XStream) is a 2.4 GHz system capable of continuous 19.2 kilobits per second half-duplex transmission. A frequency-hopping spread-spectrum technique is used to reduce interference disturbance. The maximum line of sight range using the proposed non-directional dipole antenna is quoted as 3 miles (5 km). Since the aircraft will not be flown beyond visual sight during flight testing, this range extends beyond that required to offer an acceptable degree of reception quality assurance. As with all radio modems, the selected device was felt to offer the best compromise between data rate, transmission range and license-free ISM (industrial, scientific, medical) band availability. Should the

eventual aim of beyond visual sight operation be required, a secondary communications system (most likely satellite linked) would be required. The selection and integration of such a system has not been required during this research.

A separate wireless interface links the test pilot's manual control transmitter to an onboard COTS receiver. This transmitter and receiver pair use a separate radio frequency specifically allocated for model aircraft control. The receiver decodes this signal to actuator commands that are passed to the switching circuit of the daughter board. The autopilot commands the daughter board to selectively route either its own or the manual receiver signals to individual actuators, allowing fully autonomous, partially manual or fully manual control of the aircraft.

In addition, the daughter board monitors a separate emergency override signal from the manual control transmitter using a simple analogue circuit and can force a fully manual switch if requested. Since the switching circuit also defaults to full manual routing in a powered-off state, the ability for the test pilot to take full command should the autopilot loose power, develop a fault or become unresponsive is retained. The same action may be taken should the ground station fail and commands no longer be sent to the autopilot. The receiver and actuators are powered independently from the autopilot and RF modem, partially for noise considerations but also for the failsafe described.

As a final safety feature, the manual control receiver is capable of commanding default actuator positions should it loose the carrier signal from the test pilot's transmitter. These actuator positions are preset to hold the aircraft in a gentle turn while setting the engine to idle, with the aim of minimising both distance travelled without test pilot control and structural damage to the aircraft upon landing. The emergency override signal is also set active to remove autopilot interference. Since the engine is idled rather than cut off altogether, the possibility remains of regaining control should the test pilot's signal be reacquired.

3.3 Autopilot Software

3.3.1 Overview

The autopilot's software structure (Figure 3.12) closely represents that of the first form Simulink model (Figure 3.5) due to their shared Simulink-based development process. The software wrapper described in Section 3.1.4 switches the input source

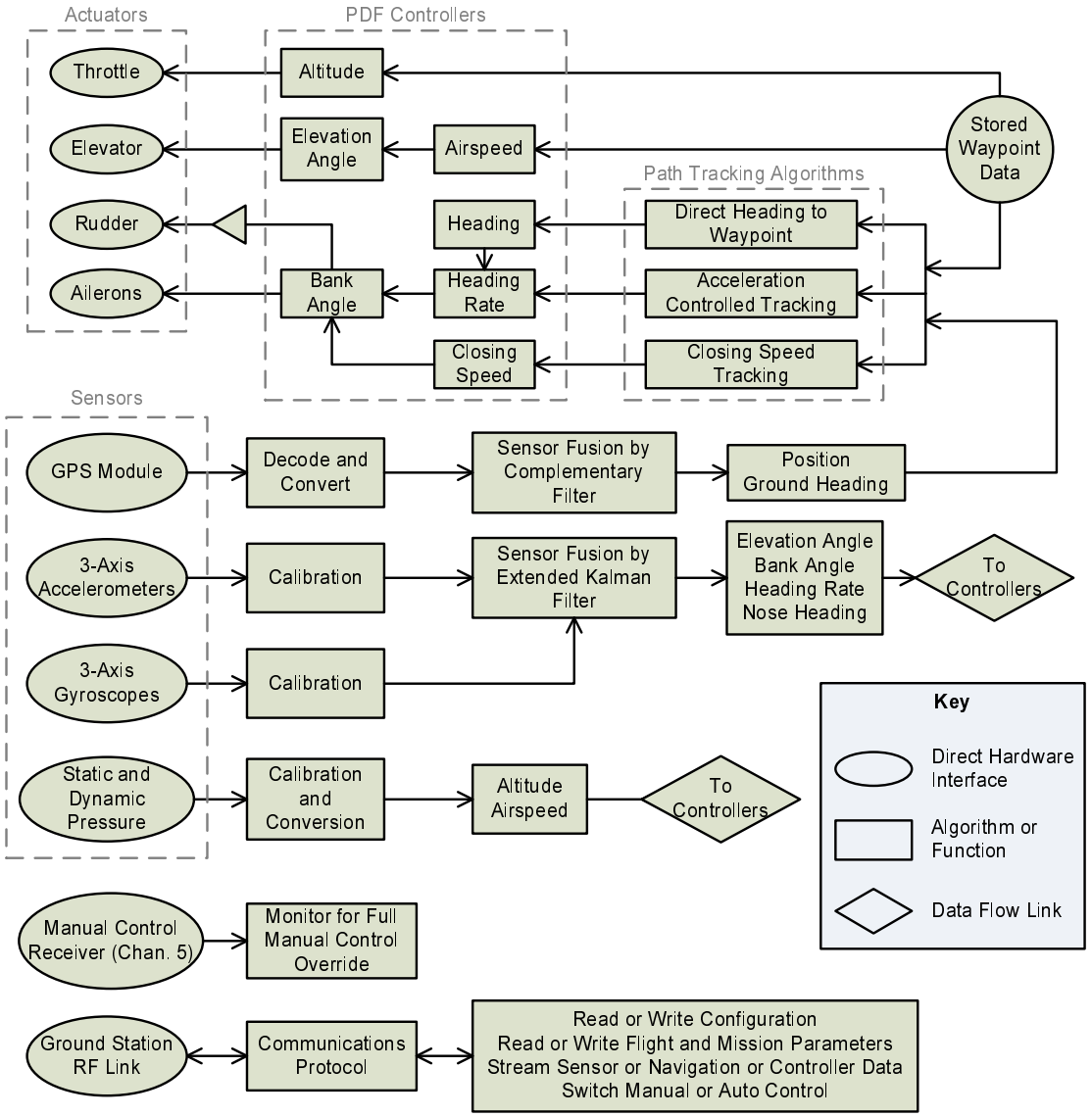


FIGURE 3.12: Autopilot software structure

of the round hardware interface symbols (Figure 3.12) to either true hardware or simulated interfaces.

3.3.2 Communications Protocol

Due to the relatively low data rate offered by the RF modem (Section 3.2.3) the communications protocol linking the autopilot to the ground station emphasises high information density, achieved using binary representation and minimal packet structure overhead. The use of binary encoding removes the option of human-readable communication with the autopilot without custom ground station software, though such software is required by the project in any case. Each

data packet is identified by a single 8-bit reference and typically has a payload no longer than 32 bytes. Many data packets do not require payload data; their reception alone is sufficient information. The complete packet structure is as follows:

<DLE><ID><payload><DLE><ETX><CRC>

Where:

- <DLE> is the byte value 16
- <ETX> is the byte value 3
- <ID> is any 8-bit value (except 16, 3 or 0) used to identify the packet type
- <payload> is any number of payload bytes
- <CRC> is an 8-bit CRC of all previous bytes in the packet

While the payload may contain any byte value, values equal to <DLE> have an extra <DLE> byte added after them during transmission. Similarly, the receiver strips the second <DLE> byte (if present) prior to further use. This enables the true packet termination byte <ETX> to be flagged only after an odd number of <DLE> bytes. The adoption of this scheme offers a degree of protection against incomplete packet reception or reception starting halfway through a packet transmission, since incorrectly initiated or terminated packets will be ignored. There is no need to specify the payload length prior to transmission, allowing varying length packets with the same identification and the dynamic construction of packets during transmission. In addition, the packet ID of zero is considered a reset communications command. This enables the transmitted sequence <0><DLE><0> to always reset the receiver state machine, regardless of the current packet decoding state. The package structure is based on the Trimble Standard Interface Protocol [184] used to communicate with the GPS modules, though has been extended by the CRC, reset communications sequence and custom state machine to form a novel protocol suitable for UAV application.

The packet described is transmitted by the RF modem through a series of data chunks themselves formed by the modem's own low-level packets. These low-level packets contain sufficient header information for the modem to handle missing or corrupt low-level packets and request a repeat transmission. To aid this, the modem packets include 16-bit cyclic redundancy check (CRC) information. The RF

transmission protocol is transparently handled by the modem, so the autopilot and ground station software can simply expect the behaviour of a standard serial data link. The downside to this transparency lies with the inability of the autopilot or ground station software to know whether a missed low-level packet has inadvertently caused a section of its own data packet to go missing (since this top-level data packet may have been fragmented into multiple low-level modem packets). Therefore, though the modem uses a 16-bit CRC, the autopilot and ground station data packets also terminate with their own minimum size (8-bit) CRC value. While this CRC is normally only required to detect missing sections during RF transmission, it also provides a degree of error checking when communication is performed using a direct serial cable connection.

3.3.3 Data Streaming

While the RF modem's transmission rate is sufficient for uplink (ground station to autopilot) requirements (predominately command updates), downlink requirements often involve real-time monitoring of aircraft and autopilot performance. Such data streams typically require the rapid update of multiple parameter values. While an ideal downlink would stream all measured parameters as often as they are updated by the autopilot, such behaviour cannot be achieved with the available transmission rate. Instead, the available parameters are categorised and split into several independent streams. Only one stream may be transmitted at a time, though the amount of data carried by each stream's payload would vary. For categories where high frequency data is of particular interest, each data stream payload contained only the few essential parameters directly associated with its analysis. For example, one data stream would focus on an individual PDF controller, allowing rapid update of only its input, output and integral parameters. Other categories demanded larger payloads but less frequent updates, enabling a wider range of parameters to be compared simultaneously. An example would be the navigation data stream. The complete list of data streams are shown in Table 3.1. Packet transmission rates are set by integer divisors of the 70 Hz flight control update.

Since the majority of parameters are internally updated by the autopilot at a higher rate than is possible to transmit, digital filtering is performed by the autopilot on all data streams. The software implements a finite impulse response (FIR) filtration scheme shown by Equation 3.3. For each update n the discrete filtered output Y_n is produced from $M + 1$ coefficients B_k and $M + 1$ previous

TABLE 3.1: Data streams used in UAV to ground station downlink

Name	Payload size (bytes)	Packet transmission rate (Hz)
Controller	13	17.5
Flight	28	7
Navigation	35	5.83
Sensor	20	5.83
Optimisation	9	8.75
Extended Kalman Filter	36	5.83
Closing Speed Tracking	35	5.83

input samples X_{n-k}. The filter coefficients B_k determine the filter response and were designed interactively using MATLAB.

$$Y_n = \sum_{k=0}^{M} B_k X_{n-k} \tag{3.3}$$

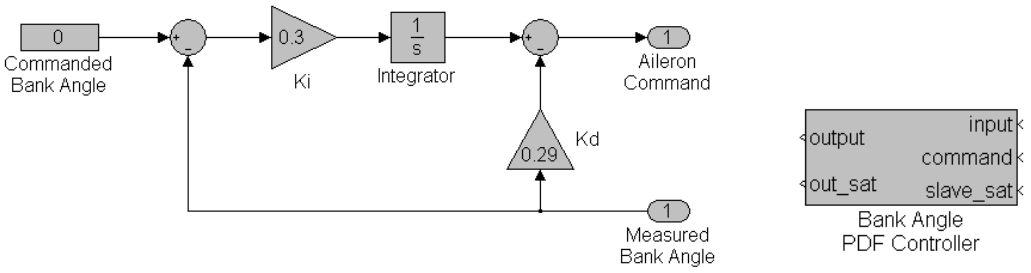
Different coefficient sets with unique low-pass cutoff frequencies are implemented for each packet transmission rate (Table 3.1) to maximise data stream bandwidth while ensuring the Nyquist sampling criteria is upheld. Since the output Y_n of this decimation filter will only be calculated every A multiples of the flight control update (where A is 70 Hz divided by packet transmission rate) the processing overhead is reduced by spreading this output calculation evenly across intermediate flight control updates, eliminating significant burst loads on the processor.

Integer constants C and D are defined by Equation 3.4 where C is the smallest number of FIR iterations per flight control update and D is the update number after which $C + 1$ iterations are made instead.

$$\begin{aligned} C &= \left\lfloor \frac{M}{A} \right\rfloor \\ D &= A(C+1) - M \end{aligned} \tag{3.4}$$

An input sample storage buffer S is created with length M and filled with the last $M - A$ samples. A counter i is incremented at the start of each flight control update, starting from 1 and counting to A before being reset. At each update i, current input sample X_i is stored in S by Equation 3.5.

$$S_{M-A+i-1} = X_i \tag{3.5}$$



(a) Block (b) S-Function

FIGURE 3.13: Simulink representations of a PDF controller

Partial output κ_i is then calculated by Equation 3.6.

$$\kappa_i = \begin{cases} \sum_{k=M-DC-1-(i-D-1)(C+1)}^{M-DC-(i-D)(C+1)} B_k S_{M-k-1} & i > D \\ \sum_{k=M-Cn}^{M-C(n-1)-1} B_k S_{M-k-1} & \text{otherwise} \end{cases} \tag{3.6}$$

When $i = A$ filter output Y_n is the sum of the partial outputs (Equation 3.7) and is transmitted with the data stream packet. i is reset to 1, and the contents of S shifted back by A indexes (Equation 3.8). In the practical implementation a circular buffer is used for S, removing this requirement.

$$Y_n = \sum_{i=1}^{A} \kappa_i \tag{3.7}$$

$$\begin{aligned} S_k =& S_{k+A} \\ & \text{where } k = 0 \ldots (M - A - 1) \end{aligned} \tag{3.8}$$

3.3.4 Controllers

Following the literature review the PDF controller scheme is adopted. This controller was felt to offer improved performance and requires fewer gain parameters than the PID, potentially reducing tuning time and saving microcontroller memory.

With reference to Figure 2.3 in the literature review, a corresponding Simulink block representation of a PDF controller is shown in Figure 3.13(a). In this case, the process state bank angle is controlled using aileron deflection. When implemented as C code an anti-wind-up limiter was included to saturate the integral value once the commanded output reaches its limits. In addition, master controllers freeze their integration should a slave controller output saturate. Use

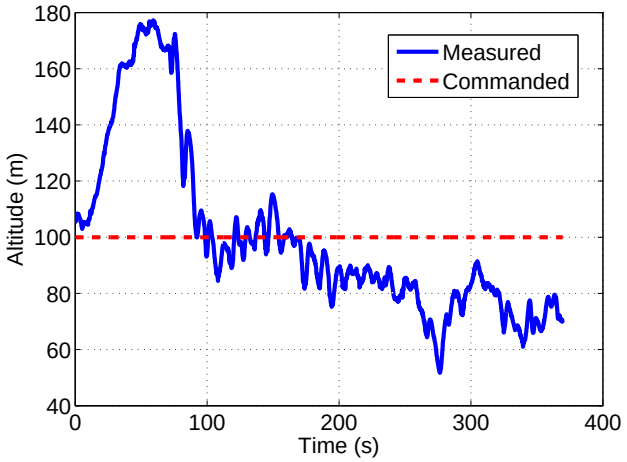


FIGURE 3.14: Poor altitude control of the Trainer 60 during flight test 19

of these standard techniques improves the controller response in the presence of limited range outputs. The Simulink representation of this cascade-capable implementation is shown in Figure 3.13(b) and pseudo-code presented in Appendix C.

Individual controllers are arranged as shown in the software overview (Figure 3.12). Both the software and Simulink autopilot implementation is capable of assigning a user-defined command value to any of the controllers and activating the relevant control path. This allows user defined commands such as hold bank or elevation angle to override higher-level controller requests.

While early simulations suggested airspeed control by throttle and altitude control by elevation angle would lead to a more responsive and stable behaviour, initial practical flight tests showed poor performance (Figure 3.14) even after in-field controller gain tuning. As a result the control path was switched to control altitude by throttle and airspeed by elevation angle. This switch resulted in considerably improved behaviour during real flight (Figure 3.32(c)). The scheme is also considered safer, which is particularly relevant during the early stage flight testing with unproven controller gain values. The autopilot will respond in a similar manner to a human pilot in the event of sudden height loss (increase engine power) or sudden drop in airspeed (lower the nose). In the original control scheme, the autopilot would command a high elevation angle at low altitude (in an attempt to climb) which may stall the aircraft even with full throttle applied. Of particular concern would have been the autopilot's response to an engine failure, with this response to the descending altitude almost certainly stalling the aircraft.

The literature review and early simulations do however suggest the original control scheme may lead to a more accurate altitude hold once the aircraft is established

in straight and level cruise. Reverting to this control scheme under such conditions whilst maintaining the safer control scheme at all other times (similar to MicroPilot and Piccolo operation) may offer improved overall performance. However, during the current phase of flight testing the majority of flight operations are outside cruise conditions and as such this switching method has not been implemented.

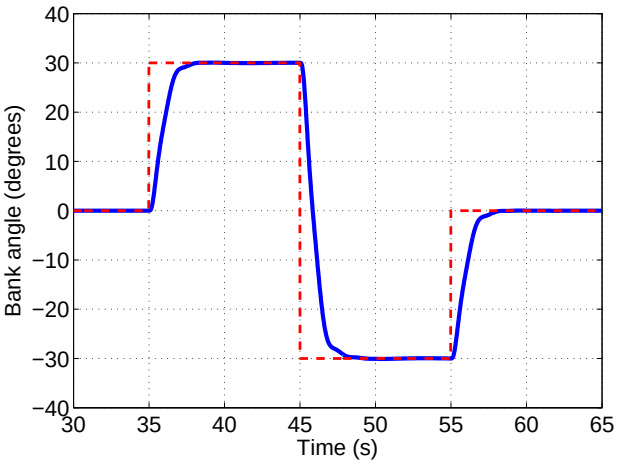
The aircraft is controlled with reference to true airspeed (TAS). This measurement was deemed more appropriate than CAS due to its increased relevance to a ground-based observer and for navigation. TAS is derived from CAS by correcting for changes in air density due to altitude and temperature, as shown by Equation 3.9. As for altitude estimation (Equation 3.1), the specific gas constant R_d of dry air is taken as 287.053 $Jkg^{-1}K^{-1}$. Both dynamic pressure P_{dyn} and static pressure P_{static} have units of Pa.

$$V_{TAS} = \sqrt{\frac{2P_{dyn}R_dT}{P_{static}}} \tag{3.9}$$

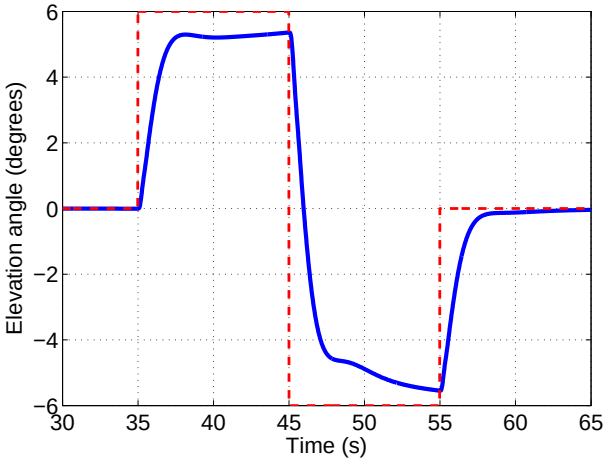
Since the current autopilot hardware does not measure outside air temperature, T is assumed to be fixed at T_0 (the ISA value of 288.15 K). Appendix D confirms this assumption is acceptable under expected flight conditions.

K_d and K_i gain pairs were manually chosen for each controller using an trial and error process with observation of the simulated aircraft response. True aircraft states are passed to the controllers during tuning and no wind is simulated to ensure the observed response is due to the controlled behaviour alone. Low-level controllers were tuned first (Figure 3.15) with control surfaces of untuned controllers fixed to trim positions. Once satisfied by their behaviour, the mid-level controllers were tuned (Figure 3.16). The closing speed controller is dealt with separately in Section 5.4. Finally the high-level heading controller was tuned, though this is a proportional only controller so only one gain was required (Figure 3.17). While this tuning processes was performed a number of times following system design changes and practical flight results, only the results using the most recent controller gains are shown. These correspond with the complete set of gains used for Flight Test 20 (Section 3.5).

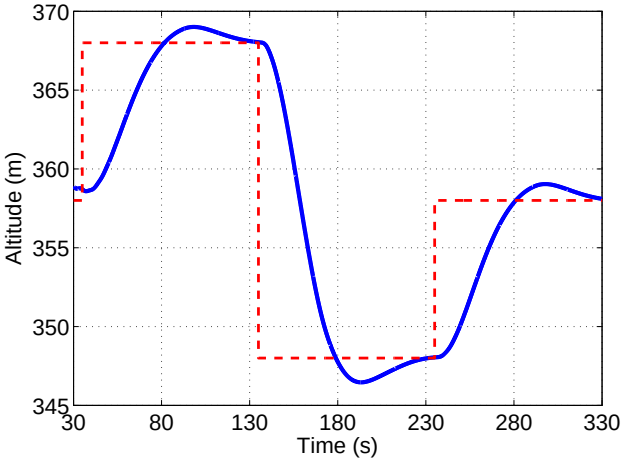
Bank and elevation angle control response is seen to be relatively quick ($\sim$2 seconds response to a step command), as required for aircraft stability particularly in the present of wind gusts. The undershoot observed in the elevation control was found to benefit airspeed control response, so was tuned accordingly. This small offset (eventually corrected by the controller's integral component) was considered acceptable since user-requested elevation angles are rarely set in practise. Due to the rapid response of the bank angle controller, subsequent higher-level master



(a) Bank angle

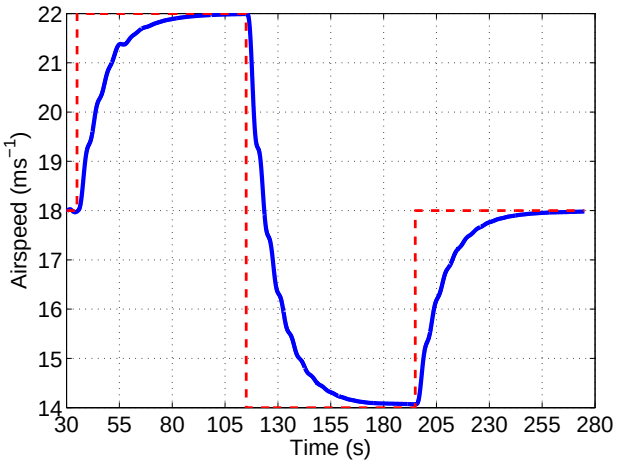


(b) Elevation angle

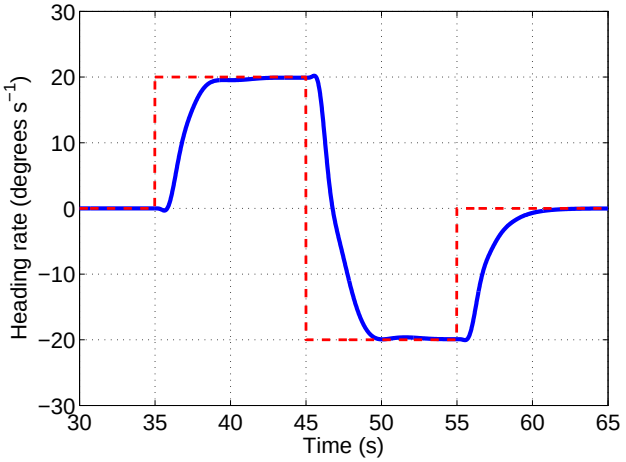


(c) Altitude

FIGURE 3.15: Simulated low-level controller response for the Trainer 60



(a) Airspeed



(b) Heading rate

FIGURE 3.16: Simulated mid-level controller response for the Trainer 60

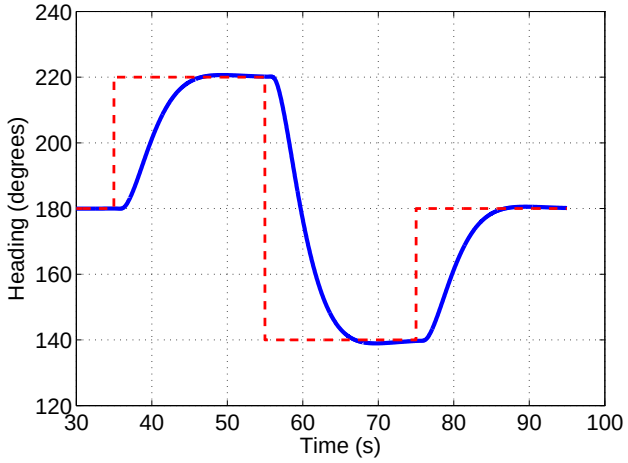


FIGURE 3.17: Simulated heading controller response for the Trainer 60

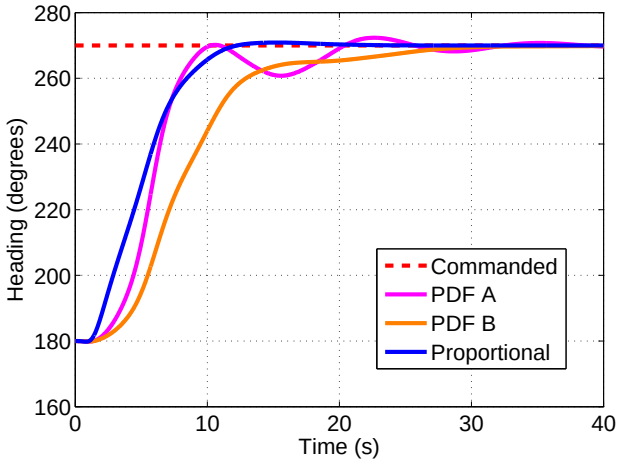


FIGURE 3.18: Comparison between PDF and proportional heading control for the simulated Trainer 60

controllers (heading rate and heading) are capable of achieving similarly quick response. Such a response is required for accurate path tracking. An initial delay is however observed in the response for these higher-level controllers due to the propagation delay down the control path, caused by the finite response time of the slave controller following a change in commanded set point. This consideration lead to the selection of a proportional only heading controller, since simulation results suggested the additional delay introduced by a further integral term of a PDF controller degraded performance in comparison. Figure 3.18 shows two PDF responses in comparison to a proportional response. An increased response delay is seen for both PDF responses. The PDF controller tuned for a fast response (PDF A) produces unavoidable oscillation. When tuned for a slow response (PDF B) the oscillation is corrected though the control becomes slower than the proportional controller.

It is apparent the tuning process has selected a particularly slow response for airspeed and altitude. This was required to reduce Phugoid-related oscillation, detailed in Appendix E. While the current response is considered adequate for test flight purposes, improvements to the current altitude control scheme may be gained from future research should the project require enhanced performance in this area. One possible route may be through gain scheduling (such as that provided by MicroPilot and Piccolo) due to the relationship observed between current airspeed and altitude controller stability in Appendix E. Disadvantages of such a technique are the need for unbiased and low noise airspeed readings for accurate scheduling and the requirement for multiple altitude controller gain pairs to be identified and tuned. An alternative route may be the use of energy flow

prediction, such as that adopted by the newer (version 2) Piccolo autopilot [149]. The algorithm is proprietary, though is believed to adjust throttle and elevator together to match an energy flow calculated from the desired airspeed and vertical climb rate. Disadvantages would include the additional complexity (and associated risk), required knowledge of the aircraft's current mass and marked departure from the current control scheme. As a result, the current altitude and airspeed control scheme has been retained for this research phase in light of the successful flight test results.

While the controller gains selected are believed to be suitable for test flights of the real Trainer 60, the selection process still requires a high degree of manual trial and error iterations. Inevitable differences between the simulated and real Trainer 60 dynamics will add further inaccuracies to these chosen gain values. As a result, Chapter 4 details research towards an automated in-flight tuning procedure.

3.3.5 Sensor Fusion

Following the review of sensor fusion literature in Section 2.4, two sensor fusion algorithms are ran in series by the autopilot. An extended Kalman filter (EKF) is chosen to provide the attitude and heading reference system (AHRS). A simpler complementary filter is used as the position reference system (PRS) as well as ground heading and speed estimation.

Early autopilot development used a single complementary filter for all sensor fusion requirements. While considerably quicker and easier to implement, an important disadvantage of this technique was the inability of the filter to estimate states not directly measured. Such states include the gyroscope bias values for the AHRS. While these bias values can be calculated from the mean gyroscope readings while the aircraft is at rest, the same cannot be achieved while the aircraft is in flight or onboard a research vessel in heavy seas. Previous study of the sensors' output and documentation had highlighted the continuously varying nature of the gyroscope's bias. The gyroscopes' datasheet highlights changes in temperature as a primary factor of this variance. As a result a continuous bias estimation ability was sought, in particular due to the integral nature of the angle estimation leading to large errors accumulating from relatively small bias effects. The switch from a complementary filter to extended Kalman filter was performed for similar reasons by Saripalli et al [119] referenced during the literature review. The PRS does not require the estimation of unmeasured states, and following satisfactory

simulation and test flight results the complementary filter has been retained for this functionality due to its significantly lower computational requirements.

3.3.5.1 Extended Kalman Filter

The AHRS requires an internal representation of aircraft orientation. While several advantages of quaternions were identified by the literature review (Subsection 2.4.4), the current EKF implementation retains Euler representation. Euler angles were favoured during the initial development of the filter due to their ease of visualisation (also highlighted in Subsection 2.4.4) aiding algorithm development and code debugging. Following the filter's successful operation in simulation and field trials, little incentive was provided for rewriting for quaternions. This was due to experimental results showing the autopilot was capable of executing the Euler EKF iteration at satisfactory speed and the Euler singularities were not experienced during practical flight since bank or elevation angles in excess of ±80 degrees would have been required.

The EKF therefore estimates six states, x, shown in Equation 3.10. The first three represent Earth-referenced bank, elevation and heading angles respectively. The final three represent roll, pitch and yaw gyroscope biases respectively.

$$x = \begin{bmatrix} \phi \\ \theta \\ \psi \\ b_p \\ b_q \\ b_r \end{bmatrix} \tag{3.10}$$

For this application, control inputs u are the body-referenced angular rates (from the gyroscopes). Process noise sources w are the angular rate readings and the bias estimates themselves. Since each gyroscope sensor is the same component make, each gyroscope is assumed to have the same noise standard deviation n_{gyro},

as are each bias estimate n_b. Equation 3.11 shows the assignment of u and w.

$$u = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad w = \begin{bmatrix} n_{gyro} \\ n_{gyro} \\ n_{gyro} \\ n_b \\ n_b \\ n_b \end{bmatrix} \tag{3.11}$$

With reference to Equation 2.1 in the literature review, the process model is described by function f in Equation 3.12 given EKF iteration time period Δt. Body-referenced angular rates are converted to the Earth reference frame prior to integration.

$$f(x,u,w) = \begin{bmatrix} x_0 + \Big(u_0 - w_0 - x_3 + \big((u_1 - w_1 - x_4)\sin x_0 \\ \qquad + (u_2 - w_2 - x_5)\cos x_0\big)\tan x_1\Big)\Delta t \\ x_1 + \Big((u_1 - w_1 - x_4)\cos x_0 - (u_2 - w_2 - x_5)\sin x_0\Big)\Delta t \\ x_2 + \Big(\big((u_1 - w_1 - x_4)\sin x_0 \\ \qquad + (u_2 - w_2 - x_5)\cos x_0\big)\sec x_1\Big)\Delta t \\ x_3 + w_3\Delta t \\ x_4 + w_4\Delta t \\ x_5 + w_5\Delta t \end{bmatrix} \tag{3.12}$$

Direct state measurements are provided by the accelerometers and GPS ground heading ψ_{GPS}. When the aircraft is at rest Equation 3.14 relates bank and elevation to accelerometer readings (A_x, A_y) and gravity (g taken as 9.81 ms^{-1}).

$$\theta = \sin^{-1}\frac{A_x}{g} \tag{3.13}$$

$$\phi = \sin^{-1}\frac{-A_y}{g\cos\theta} \tag{3.14}$$

Other approaches remove the need for g and prior knowledge of elevation in the bank angle estimate by using the z-axis accelerometer A_z. However, the advantage of Equation 3.14 is the reduction in the number of sensor measurements used thus considerably reducing computational processing. This is primarily due to the need to invert an n $\times$ n matrix during the EKF calculation, where n is the number of

measurements taken. The measured states z are shown in Equation 3.15.

$$z = \begin{bmatrix} A_x \\ A_y \\ \psi_{GPS} \end{bmatrix} \tag{3.15}$$

The use of only three measurements results in a general 3×3 matrix that can be inverted relatively easily in code by dividing its adjoint matrix with its determinate.

Though the EKF should reject short-term disturbances in these measured values as the aircraft manoeuvres in flight, longer-term external forces will cause significant bias to the EKF estimate. The primary cause of such disturbance is expected to be centripetal force as the aircraft turns. Centripetal acceleration A_c in ms^{-2} is described by Equation 3.16.

$$A_c = \omega_a V \tag{3.16}$$

ω_a represents the angular rotation in radians per second and V is the velocity in ms^{-1}. The velocity vector, angular rotation axis and acceleration vector are orthogonal. Since the orthogonal gyroscope and accelerometer sensors are aligned to the same axis, the gyroscope rate measurements can be used to directly offset the corresponding accelerometer reading without knowledge of the aircraft's orientation. Outside of transient manoeuvres, the velocity of the aircraft is expected to be directed along the x (longitudinal) aircraft axis. This assumption results in no centripetal acceleration being recorded by the x-axis accelerometer A_x, since this would require velocity components in the y or z direction. In addition, true airspeed V_{TAS} derived from the Pitot tube parallel to the x-axis remains a valid velocity measurement for centripetal calculation. Since the z-axis accelerometer A_z is not actually used in the EKF measurements, only the centripetal acceleration on A_y need be considered. The x and z axes angular rates, p and r respectively, can both contribute to this acceleration. However, expressing centripetal acceleration in terms of ω_a and radial displacement R_a in m (Equation 3.17) shows this acceleration will be small if R_a is small.

$$A_c = \omega_a^2 R_a \tag{3.17}$$

Due to the action of the ailerons it may be assumed the aircraft will always roll around its x-axis, and that the A_y sensor will lay close to this axis. As a result, the radial displacement of A_y from any x-axis rotation p will always be small - and so will be the affect on A_y. Additionally changes in bank angle are expected

to be relatively short-lived, with p remaining zero outside of transient states. For these reasons, only r is considered to influence A_y by the addition of centripetal acceleration during sustained turns as in Equation 3.18.

$$A_c = rV_{TAS} \tag{3.18}$$

Returning to the EKF measurement function h (Equation 2.2 in the literature review), Equation 3.19 relates A_x, A_y and ψ_{GPS} to the estimated states x, taking into account centripetal force.

$$h(x, v) = \begin{bmatrix} g\sin(x_1) + v_0 \\ V_{TAS}(r - x_5) - g\cos(x_1)\sin(x_0) + v_1 \\ x_2 + v_2 \end{bmatrix} \tag{3.19}$$

Since the autopilot hardware does not include magnetometers, a compass heading is not available and as such the GPS ground heading is used instead. The use of Earth-frame ground heading rather than body-frame aircraft heading (such as would be provided by a compass) has the disadvantage of not being directly linked to heading rate as recorded by the gyroscopes, particularly in the presence of wind. In addition the velocity data from the GPS is lagged by approximately one second, causing an additional offset in the two headings during rotation. Under such conditions, the time update model does not reflect the measurement update. The EKF attempts to correct these discrepancies by modifying the gyroscope bias values, though since the error is not due to gyroscope bias such changes are invalid. This behaviour can be improved by assigning a lower noise level to the gyroscope and bias sources than the GPS heading source (typically $n_b \ll n_{gyro} < v_2$), reducing the tendency of the EKF to modify the bias due to short-term discrepancies in aircraft and ground heading rates. Care is taken not to reduce the value of n_b too low so as to prevent the EKF from tracking genuine changes in gyroscope bias or to result in too slow convergence on the bias values upon start-up.

The remaining estimation calculations follow the well documented and generic EKF process described by Subsection 2.4.2. Due to the limited precision of the 32-bit floating point values used by the hardware processor, the error covariance P update is re-written to the *Joseph stabilised version* as described by Equation 2.13 in the literature review. While this form requires more calculations than the original, occasional spikes in the EKF output observed during simulation were successfully eliminated using this form.

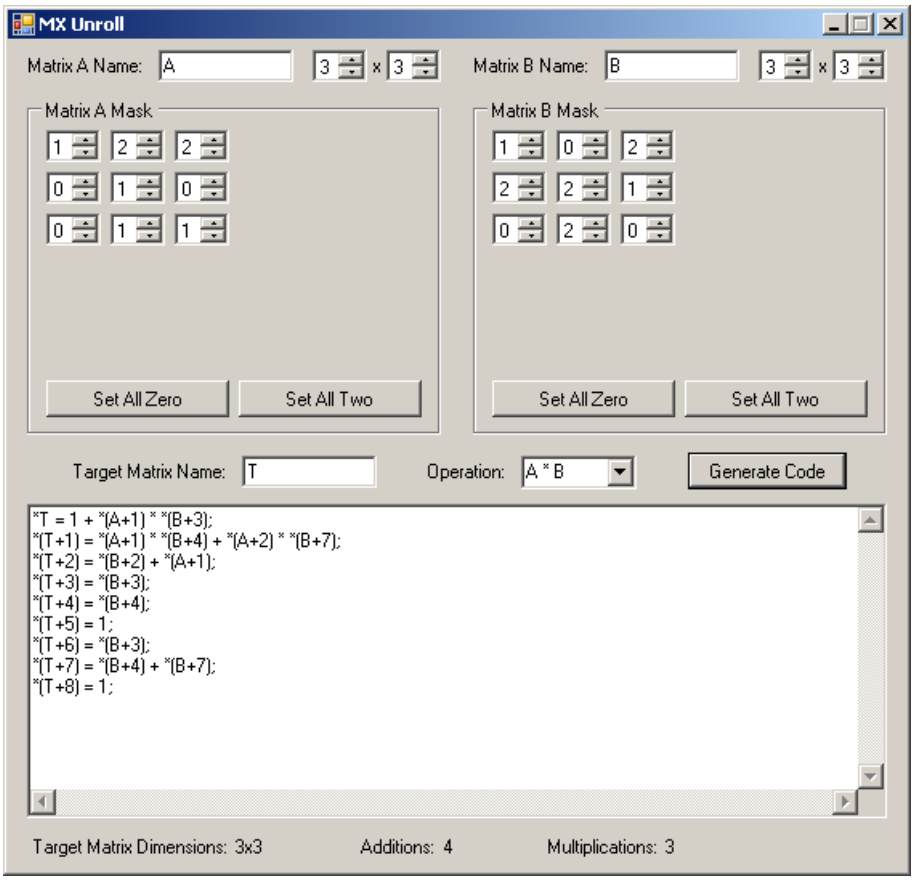


FIGURE 3.19: Optimised C-code generator for matrix operations user interface

The EKF calculation requires a number of matrix multiplications. Many of these matrices contain elements that will always be zero or one. Therefore, rather than implement a generic matrix multiplication function a separate program was written to generate C code specifically optimised to a particular matrix multiplication. The GUI allows the assignment of each matrix element to zero, one or two (the latter representing any other value). One of the matrices can optionally be transposed prior to multiplication. The output uses pointer arithmetic (rather than indexing) to ensure the code is compiled to minimally-sized machine code. The program's operation is illustrated in Figure 3.19. The use of this novel technique allowed the EKF calculation to execute on the onboard processor at the desired 70 Hz update rate, which had not previously been possible when using a generic matrix multiplication function.

A further optimisation relates to the lower (1 Hz) update frequency of the GPS ground heading in comparison to the other measurement updates (70 Hz). At times, the GPS data may also not be available. As a result, the computationally-expensive general 3×3 matrix inversion is not required every update. At other

TABLE 3.2: Extended Kalman filter noise values

Noise source	Description	Value
n_{gyro}	Gyroscope sensor noise	0.001
n_b	Uncertainty of estimated gyroscope bias	0.00005
v_0	Uncertainty of A_x sensor as θ estimate	0.05
v_1	Uncertainty of A_y sensor with centripetal acceleration correction as ϕ estimate	0.2
v_2	Uncertainty of GPS ground heading as ψ	0.005

times, the matrix to be inverted becomes a general 2×2 matrix in the upper-left of an otherwise zero 3×3 matrix, except for a single non-zero element in the bottom-right corner. This allows the full inversion result to be performed in-place with only a simple inversion of the general 2×2 matrix and the trivial inversion of the single bottom-right element, resulting in a 13% EKF processing speed increase (Appendix B).

Though initially based on noise statistics collected from the relevant sensors, a trial and error procedure was used modify the EKF noise values during simulation in a similar manner to the selection of PDF controller gains. This tuning technique is standard in the literature [1, 106]. Deviation from the pure statistical noise measurements tunes the EKF behaviour according to a qualitative assumed "trust" associated with each estimated state and measurement. The noise values chosen represent a balance between the filter's flexibility to adopt to genuine state changes (higher noise) and reject short-term disturbances (lower noise). The values selected are shown in Table 3.2.

EKF estimation results from a 2 minute simulation of the Trainer 60 are presented in Figures 3.20–3.23. Mean absolute error (MAE) for the estimated states are shown in Table 3.3. The first 20 seconds are ignored for the MAE to allow for the EKF states to settle from their initial values and better represent the long-term EKF performance.

The results show bank and elevation angle estimates to be very satisfactory for this simulation. Heading estimation is acceptable, though markedly worse than the previous two. A similar result is shown for the bias estimates, with b_r (most strongly influenced by heading rate during normal flight) under-performing when compared with the other two. The explanation is illustrated by Figure 3.24. The

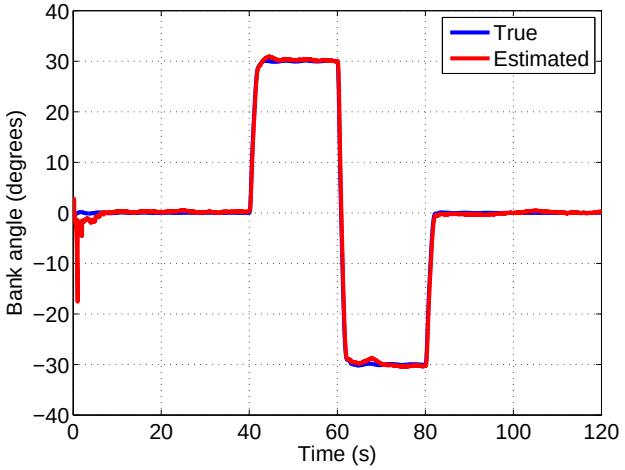


FIGURE 3.20: True and estimated bank angle from Trainer 60 simulation

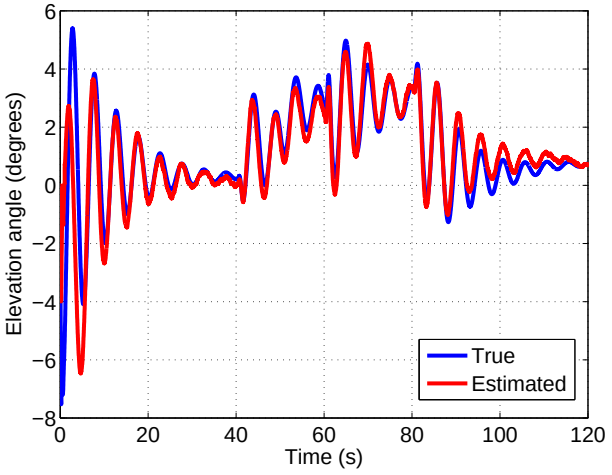


FIGURE 3.21: True and estimated elevation angle from Trainer 60 simulation

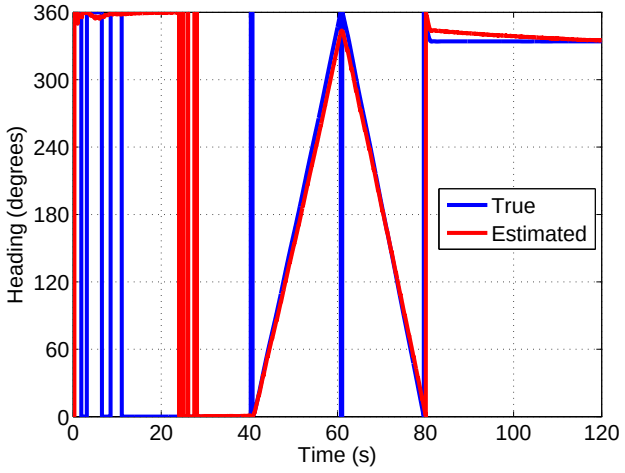


FIGURE 3.22: True and estimated heading from Trainer 60 simulation

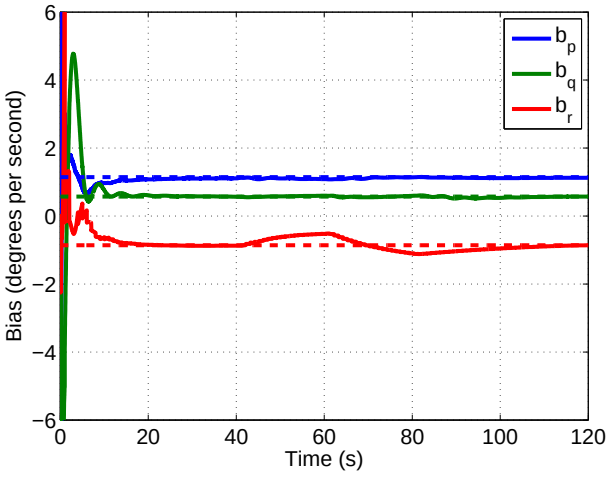


FIGURE 3.23: True (dashed line) and estimated (solid line) gyroscope biases from Trainer 60 simulation

TABLE 3.3: EKF mean absolute errors in state estimates during Trainer 60 simulation

State	Mean absolute error
ϕ	0.325 degrees
θ	0.283 degrees
ψ	5.40 degrees
b_p	0.0319 degrees per second
b_q	0.0161 degrees per second
b_r	0.118 degrees per second

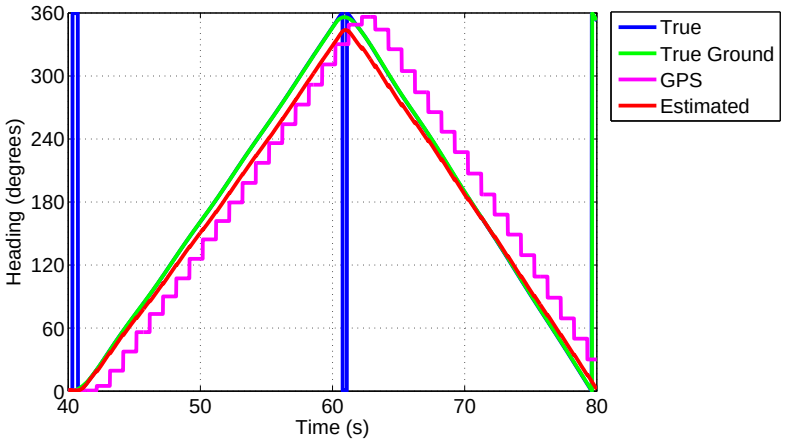


FIGURE 3.24: Heading comparison during aircraft turns in the Trainer 60 simulation

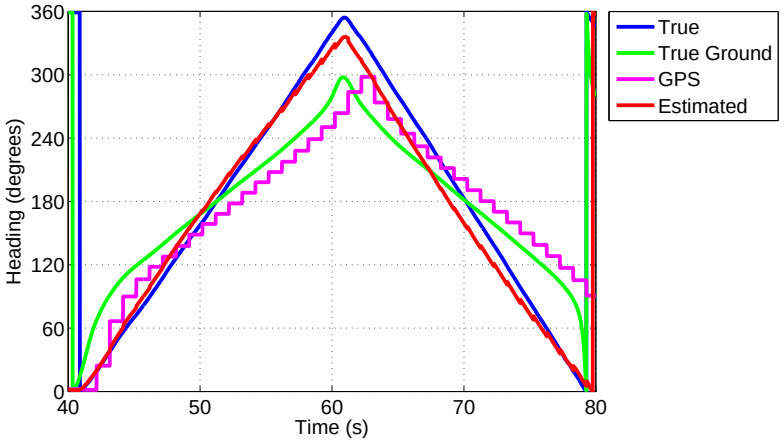


FIGURE 3.25: Heading comparison during aircraft turns in the Trainer 60 simulation with a 15 ms^{-1} northerly wind

figure focuses on the simulation time where the aircraft is turning (40 to 80 seconds) and shows true ground as well as GPS measured heading. True heading and true ground heading are seen to be almost identical, as expected given the aircraft is performing well-coordinated turns under zero-wind conditions. The staircase appearance of the GPS heading is expected due to its slower update rate, but its $\sim$1 second lag is also quite apparent. As previously highlighted, it is this lag that is causing the errors in the EKF heading estimate. Figures 3.23 and 3.24 show the EKF is performing correctly by adjusting its estimated bias values, predominately b_r, in an attempt to align its predicted heading to the measured GPS heading.

Since the error values chosen for the EKF place greater trust in the angular rates than the GPS measurement, the estimated heading still tracks changes in true heading better than the GPS and remains a better estimate of transient heading conditions. The same holds when a strong wind (15 ms^{-1} northerly) is simulated, producing the heading comparison in Figure 3.25. True heading and true ground heading are seen to differ significantly, increasing the measurement error of the GPS heading for the EKF. As expected Table 3.4 shows an increase in MAE for heading and b_r estimates, though bank and elevation angle estimates remain satisfactory.

The results highlight the disadvantages of using GPS heading over magnetic compass heading for EKF heading measurement, specifically the delay and different reference frames. In addition, the lack of a heading measurement while the aircraft is stationary on the ground (since velocity is required for a GPS heading calculation) significantly reduces the EKF's ability to track b_r. The fitting of magnetometers to the autopilot hardware is therefore recommended. Alternatively, the

TABLE 3.4: EKF mean absolute errors in state estimates during Trainer 60 simulation with a 15 ms^{-1} northerly wind

State	Mean absolute error
ϕ	0.557 degrees
θ	0.417 degrees
ψ	20.2 degrees
b_p	0.0236 degrees per second
b_q	0.109 degrees per second
b_r	0.354 degrees per second

use of a GPS module capable of a higher frequency update (and therefore smaller delay in heading) should be considered. With the existing hardware components, the use of the EKF to satisfactorily estimate bank and elevation angle has however been validated in simulation. Validation during practical flight tests is less precise due to a lack of a true attitude reference, discussed further in Section 3.5.

3.3.5.2 Complementary Filter

The operation and tuning of the complementary filter for position P_x, P_y and ground heading ψ_g estimation is straight forward in comparison to the EKF. The high–frequency components of position and ground heading are estimated by Equation 3.20, with update period Δt.

$$\begin{aligned} \psi_g &= \psi_g + (\dot{\psi} + \ddot{\psi}_g)\Delta t \\ P_x &= P_x + (\dot{P}_x + \ddot{P}_x)\Delta t \\ P_y &= P_y + (\dot{P}_y + \ddot{P}_y)\Delta t \end{aligned} \tag{3.20}$$

Current heading rate $\dot{\psi}$ is calculated during the EKF process. Ground speed V_g recorded by the GPS is converted to the Earth reference frame using the ground heading estimate, to produce the velocity components in Equation 3.21.

$$\begin{aligned} \dot{P}_x &= V_g \sin\psi_g \\ \dot{P}_y &= V_g \cos\psi_g \end{aligned} \tag{3.21}$$

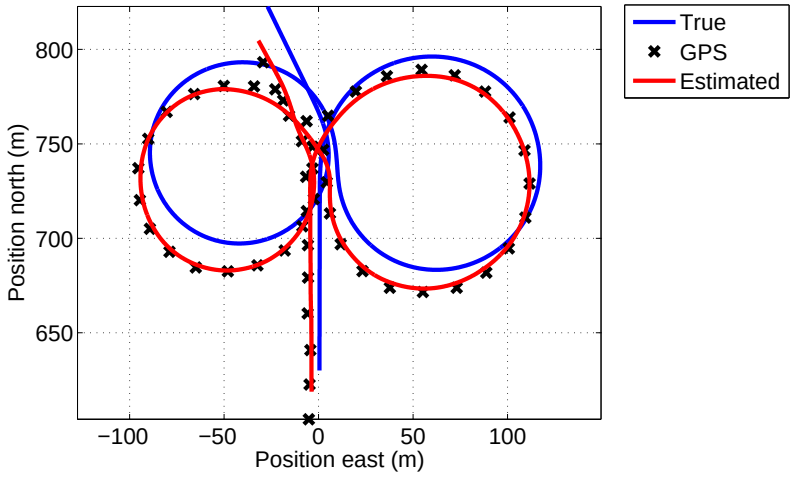


FIGURE 3.26: Position comparison during aircraft turns in the Trainer 60 simulation

The low–frequency GPS updates produce the correction factors $\ddot{\psi}_g$, $\ddot{P}_x$ and $\ddot{P}_y$ (Equation 3.22).

$$\begin{aligned} \ddot{\psi}_g &= K_h\big((\psi_{GPS} + 1.1\dot{\psi}) - \psi_g\big) \\ \ddot{P}_x &= K_p(P_{x,GPS} - P_x) \\ \ddot{P}_y &= K_p(P_{y,GPS} - P_y) \end{aligned} \tag{3.22}$$

The equation for $\ddot{\psi}_g$ attempts to compensate for the 1.1 second delay in GPS heading by using the estimated heading rate $\dot{\psi}$. While this technique generally works well during steady heading rate values (such as straight flight or established turns), the compensated result can have a greater error than the original GPS measurement outside of these conditions or when the estimated heading rate contains significant error. Experimental results suggested attempting to implement a similar prediction in the measurement update of the EKF would lead to instability. A possible explanation would be the feedback from the EKF's bias estimates influencing its estimated heading rate and in turn affecting its measurement. This predicted heading measurement would influence its new bias estimate, and so forth. Either way, it has been found better to keep the EKF estimates body-referenced - leaving the complementary filter to estimate the Earth-referenced states.

The complementary gain values K_h and K_p are manually tuned in simulation to provide a compromise between high and low frequency sensor data. Selected values of 0.8 and 0.3 respectively were chosen, with higher values representing greater trust in the GPS data. Position results from 5 seconds either side of the Trainer 60 banking during the previous (zero wind) simulation are shown in Figure 3.26. The offset observed is due to the realistic output of the Aerosim

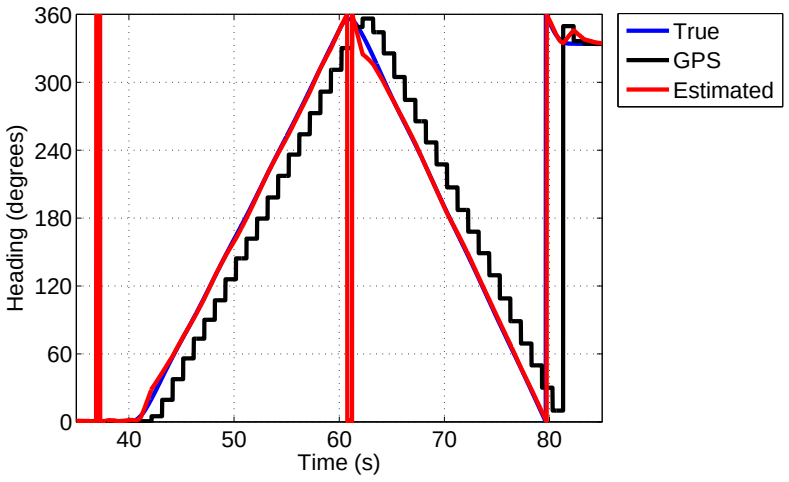


FIGURE 3.27: Ground heading comparison during aircraft turns in the Trainer 60 simulation

GPS receiver block. The complementary filter position estimate follows the GPS positions well, though due to the GPS offset itself a MAE comparison to the true position is not a fair judge of filter performance. Instead it is observed that the estimated position smoothly interpolates between the GPS positions, following a similar dynamic behaviour to the true position even using the estimated (rather than true) heading rate from the EKF. Figure 3.27 also demonstrates a good performance by the complementary filter, with smooth interpolation and apparent elimination of the GPS ground heading delay under steady-state conditions.

Should the GPS signal be lost, the complementary filter has no low–frequency update and the correction factors are set to zero. The high–frequency updates continue using (true) airspeed instead of ground speed, and the integral of the estimated heading rate as the ground heading. Figure 3.28 repeats the previous simulation but with the GPS signal lost between 35 and 80 seconds, corresponding to the start time of the figure till 5 seconds before the end time of the plotted data. Only the estimated position in each case is shown. The MAE during the 45 second period of GPS loss is 4.2 m, with a maximum error of 8.8 m. Given the moderate change in bank angle and heading rate during this time, and that the maximum error is less than half the distance travelled by the aircraft in a second, the performance is quite satisfactory. Even with the inevitable increase in position error should GPS loss occur in the presence of wind, the simulation results suggest reasonably accurate position estimates should remain during short term (<30 seconds) GPS loss. The position estimate is seen to quickly converge on the original track once GPS lock is restored in the final 5 seconds.

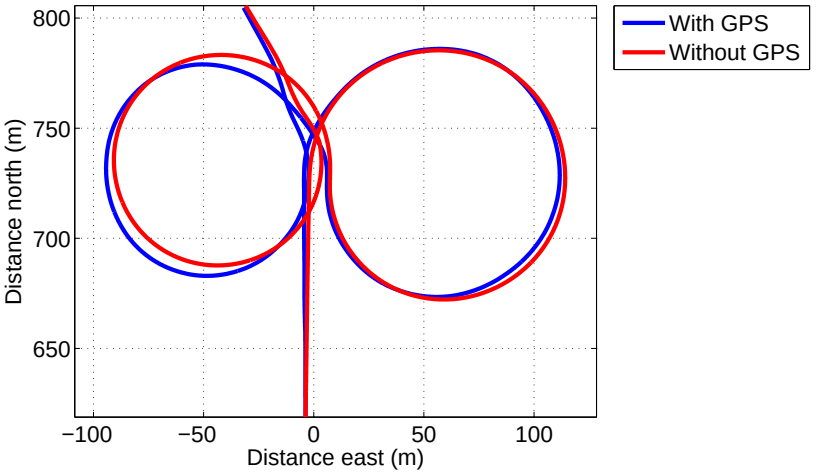


FIGURE 3.28: Position comparison during aircraft turns with and without GPS lock in the Trainer 60 simulation

3.4 Ground Station

3.4.1 Network Server

A server application has been developed as a bridge between the wireless modem (in direct communication with the UAV) and a TCP/IP network. This allows UAV operator software to execute on remote computers not physically attached to the modem. Since the optimal position for the ground station and modem is not necessarily the best location for the UAV operator's terminal, this flexibility allows both requirements to be met. This is a particular advantage during ship-based operation, where the UAV operator may not be on the deck but the UAV communications system may be. A further advantage of the server application is its ability to interface multiple network clients through the single serial link to the wireless modem, allowing the potential for real-time data sharing and multiple UAV operators. Incoming data from the client is held until a complete (and CRC-verified) communications packet is received (Section 3.3.2). Complete packets are then queued for serial transmission to the wireless modem, preventing packet fragmentation due to overlapped client data.

While multiple connections have not been utilised during this research, three areas are highlighted as particularly worthy of further study. The first is the concept of a backup UAV operator terminal for immediate use should the primary become inoperable. The second relates to a web-based monitoring system, allowing real-time UAV data to be viewed by interested parties simply using a web browser.

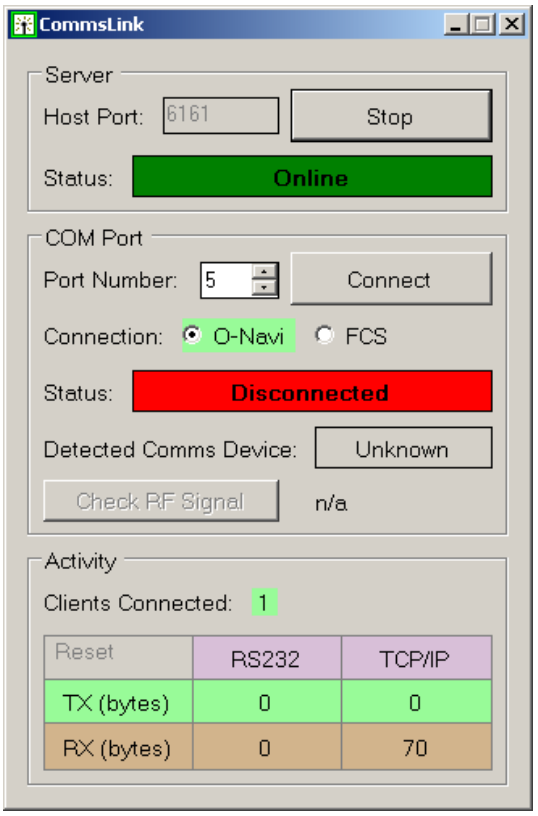


FIGURE 3.29: Network server GUI

The final suggested application involves a hand-held PDA networked via a 802.11 wireless connection. Such a device could be used to issue basic commands in the field (such as change of target waypoint) by a third party in addition to the more complex ground station monitoring and control software required by the primary UAV operator.

While these advanced features have yet to be incorporated, the underlying client/server architecture has been successfully incorporated in both the lab and in-field environments and is at present the sole method in use. The server GUI is presented in Figure 3.29

All PC-based applications developed during this project run on the Microsoft® .NET framework. While currently run on the Windows® operating system, the existence of a compatible and cross-platform framework [185] maintains the option of running under the Linux environment. Should it be necessary to run key components under this alternate operating system, the choice of the .NET framework for development will greatly increase code-reusability. In addition the framework is supported by a high proportion of PDA's, enabling the relatively straight-forward targeting of such devices for the advanced multiple-client uses previously suggested.

3.4.2 User Interface

Ground station client software has been developed to provide the UAV operator with full access to the autopilot functionality. Due to the large number of features available, these have been categorised and grouped together on single panels accessible via a tabbed interface. As a result, the operator can quickly locate required functions and is not overwhelmed by a large number of simultaneously displayed components. An exception to the dynamic display are a small number of safety-critical components whose viability is considered essential at all times. Such components are the manual or automatic control switches, connection status and autopilot status displays. In addition, a timer is permanently visible to aid estimates of remaining flight time.

To reduce confusion, a common colour-coding runs throughout the GUI. Any component highlighted in red shows it is currently under manual control; any component highlighted in green is under automatic control. In addition, such highlighting will only occur upon confirmation from the autopilot that it has switched to that state. For example, the user clicks the manual ailerons button and the request is transmitted to the autopilot. The button remains grey until an acknowledgement is received from the autopilot, upon which the button is highlighted red. In addition the button can be clicked multiple times regardless of current highlighting, to allow repeated transmission of the command should the communications link be poor.

Figure 3.30 shows a screen capture of the ground station GUI. The tab is shown in the top left (sensors tab illustrated), with the manual or automatic control switch panel to the right (manual ailerons and rudder illustrated) and connection and autopilot status panel at the bottom.

To reduce code duplication and standardise the GUI, some components were designed separately and compiled to a library. This custom library allowed multiple component instances with the same functionality to be placed within the GUI. In addition, the same components could be quickly incorporated in any application written for the project, potentially saving development time in other areas. Of particular importance was the custom moving map component developed as part of this research. While some commercial implementations exist, the closed-source nature of the libraries meant any modification to their behaviour required to meet research and operational needs would require collaboration with the third-party,

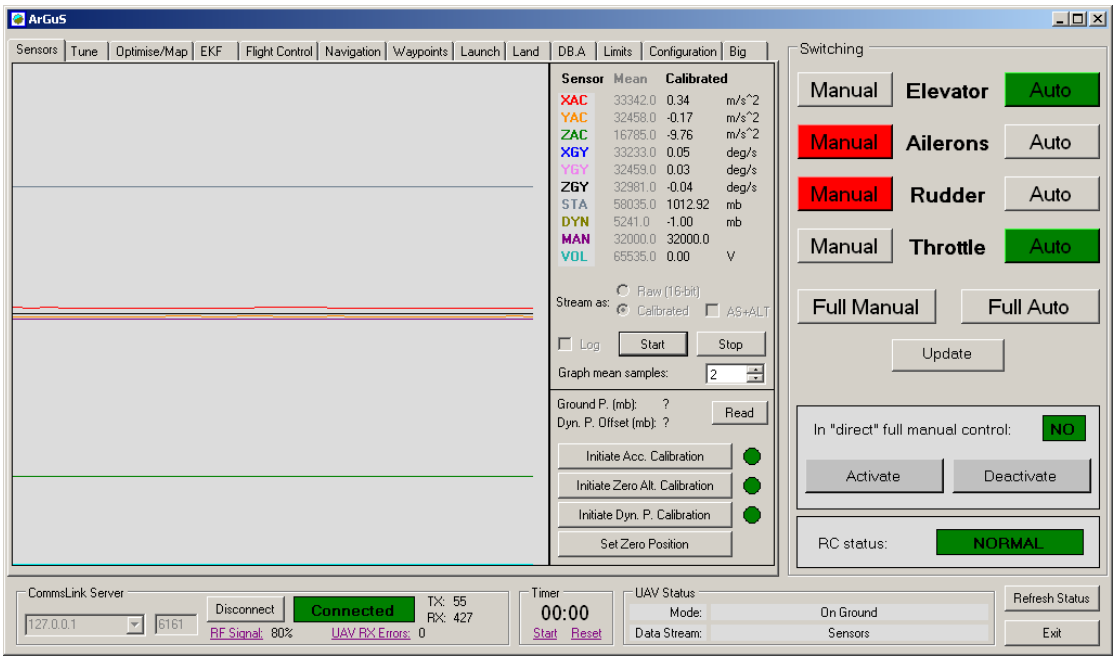


FIGURE 3.30: Ground station GUI layout

thereby risking extended development time and potential extra expense. Developing the desired components in-house has provided maximum flexibility without licensing or unexpected development delay. Figure 3.31 shows the moving map component embedded in the ground station GUI's waypoint editing tab. A correctly scaled aerial image of the flight test sight is shown as a partially transparent overlay, with full waypoint editing and map pan, zoom and grid abilities. The same component is duplicated and visually customised in the real-time navigation display tab. Waypoints and additional symbols can be placed on the map using latitude and longitude references, which are converted and drawn to local screen coordinates automatically by the component.

Aside from commanding and displaying the desired autopilot data streams in real-time, all incoming data can be logged by the ground station to comma-separated value files (compatible with both Microsoft Excel® and MATLAB). Such logging is essential for off-line analysis following a test flight. Waypoint layouts and all configurable autopilot parameters can be saved to and loaded from file by the ground station software, enabling their archive between planning sessions and fully restoring autopilot settings should the hardware memory be erased.

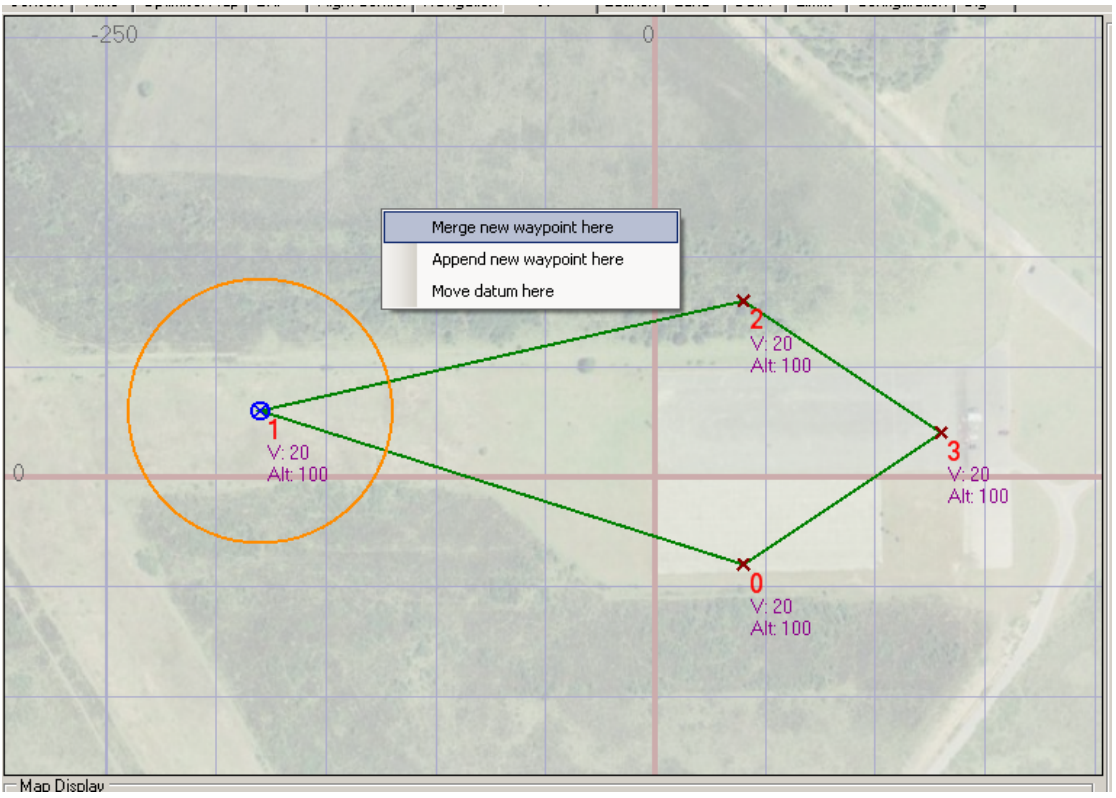


FIGURE 3.31: Custom moving map component embedded in ground station GUI

3.5 Flight Test Results

Practical flight tests required careful planning and good communication with the ground team, in particular between the test pilot and UAV ground station operator. As a result the aims and tasks of each flight test was discussed and clarified by all parties prior to take-off and the ground station itself set up in close proximity to the test pilot's standing area. The test pilots themselves were experienced model aircraft flyers capable of safely piloting the aircraft and in particular quickly and appropriately recovering the aircraft should the autopilot be likely to place itself or those nearby in any risk. All relevant Civil Aviation Authority (CAA) [186] requirements and guidance were identified and observed, in particular those in CAP658 (Model Aircraft: A Guide to Safe Flying) and CAP722 (Unmanned Aircraft System Operations in UK Airspace Guidance).

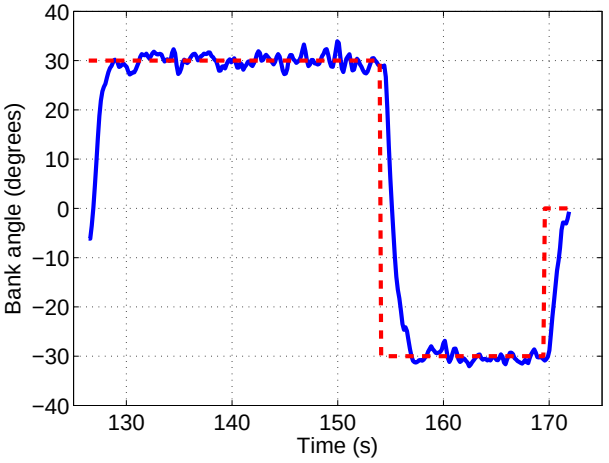
22 in-field tests have been performed to date, with each test typically consisting of 2–3 20 minute flights. Data from the two most recent flight tests taken specifically to validate the controller operation (tests 19 and 20) will be the predominate data source presented in this section.

Flight data presented by Figure 3.32 demonstrates the operation of the low-level bank, elevation and altitude controllers. Figure 3.33 presents flight data for the mid-level airspeed and heading rate controllers. Figure 3.34 presents the high-level heading hold response.

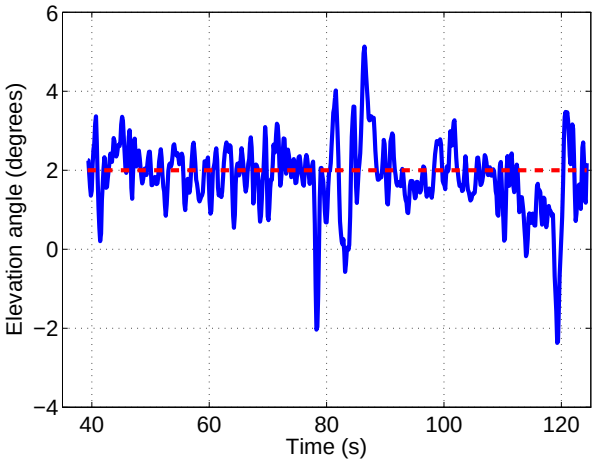
In all cases the flight test data shows higher noise levels than the equivalent controller response simulations of Figures 3.15–3.17. An increase is expected given these simulation results excluded wind and sensor noise to aid controller tuning. True states had been fed to the controllers rather than the less accurate AHRS estimations for the same reason.

Figure 3.35 compares bank and elevation angle error for realistically simulated and real flight data. The simulation was run as software-in-the-loop to ensure identical sample times and digital FIR filtering, and states recorded using the ground station client. Statistical analysis of the hardware accelerometers and gyroscope readings suggested a mean noise variance of 42 μW when the autopilot was on the lab bench. This noise variance is incorporated in the realistic sensor simulation block. A 5 mph wind was simulated with a 1 mph variance to approximate flight tests conditions. Over the 20 second sample windows, real bank angle MAE is 14 times higher than simulated bank angle and real elevation angle MAE is 3 times higher than the simulation. It is apparent significant noise sources are present in the real data that are not modelled by the simulation. Such sources may be propeller wash, engine vibration and in some cases interference from other control surfaces under manual control by the test pilot.

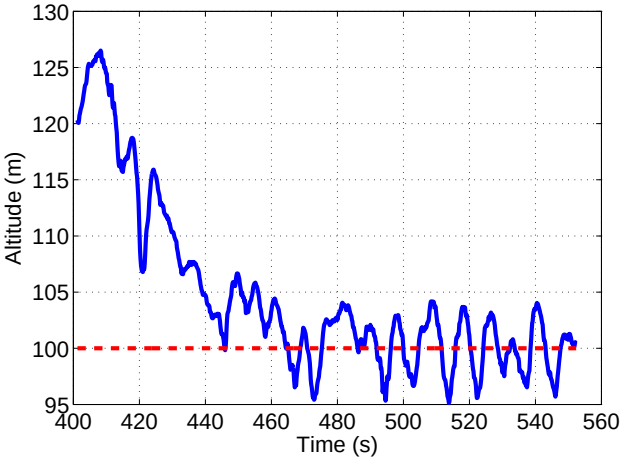
Even with these additional noise sources, the overall low-level controller response recorded during flight tests (Figure 3.32) remains close to that obtained from simulation (Figure 3.15). Response times to step changes in command are similar for bank and altitude data, though step response to elevation angle has not been recorded with the current gain set. With reference to Figure 3.32 the MAE values for selected steady-state periods are shown in Table 3.5. The values are considered within acceptable tolerances, though the apparent oscillation observed in the altitude response is an area for improvement. This oscillation, with approximately 8 seconds time period, may be due to the Phugoid effects identified during simulation in Appendix E. Airspeed recorded during this time is overlaid in Figure 3.36 and shows a similar swapping of airspeed with altitude demonstrated in Figure E.1. Whether this is due to underlying Phugoid motion of the airframe or a result of mismatched controller gains cannot be directly determined from the available data. The altitude overshoot predicted in the simulation is not apparent



(a) Bank angle hold during flight test 20

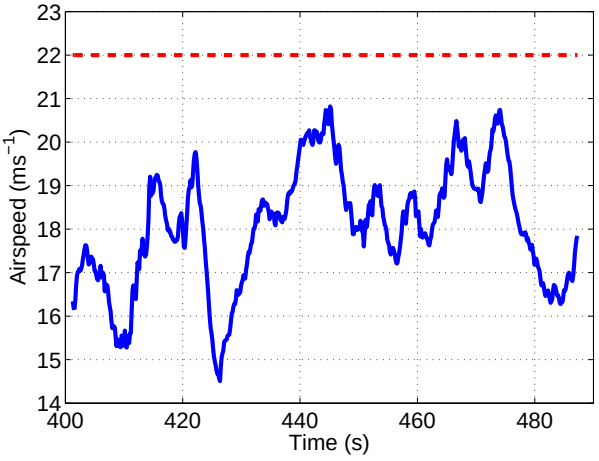


(b) Elevation angle hold during flight test 19

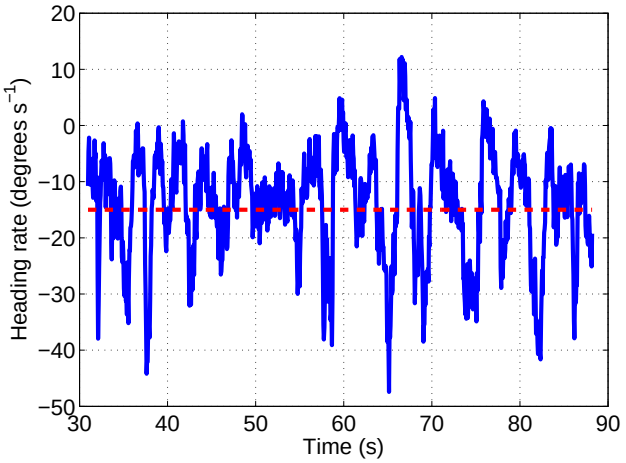


(c) Altitude hold during flight test 20

FIGURE 3.32: Flight test data during low-level control of the Trainer 60



(a) Airspeed hold during flight test 20



(b) Heading rate hold during flight test 13

FIGURE 3.33: Flight test data during mid-level control of the Trainer 60

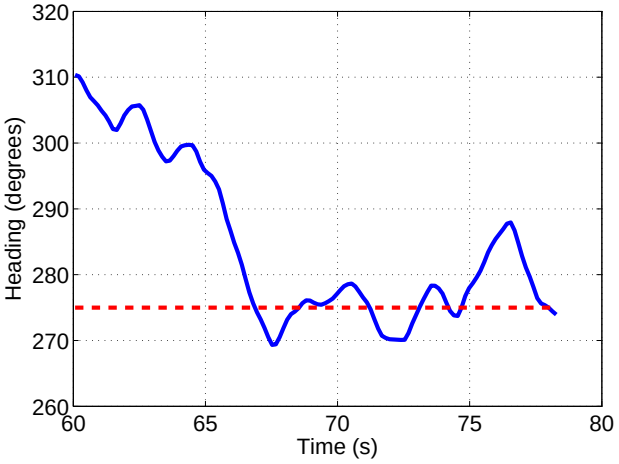
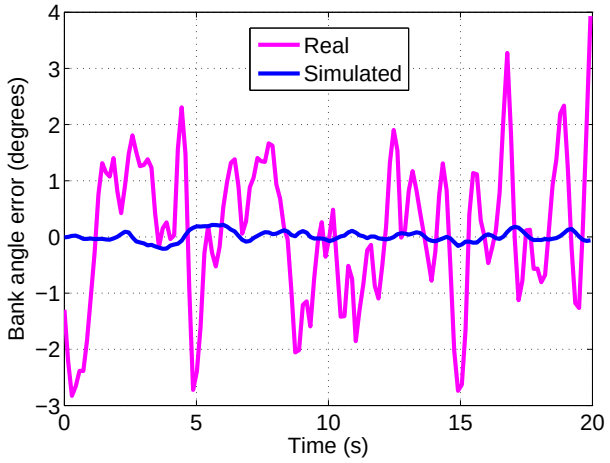
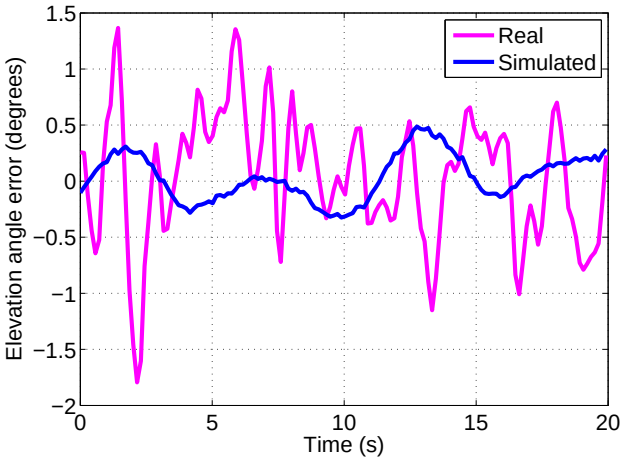


FIGURE 3.34: High-level heading control of the Trainer 60 during flight test 14



(a) Bank angle



(b) Elevation angle

FIGURE 3.35: Comparison between realistic simulation and real flight data of the Trainer 60

TABLE 3.5: Mean absolute errors of low-level controller response recorded during steady-state flight of the Trainer 60

Controller	Measurement period (s)	Mean absolute error
Bank angle	130 - 150	1.0 degrees
Elevation angle	40 - 125	0.66 degrees
Altitude	470 - 550	2.1 metres

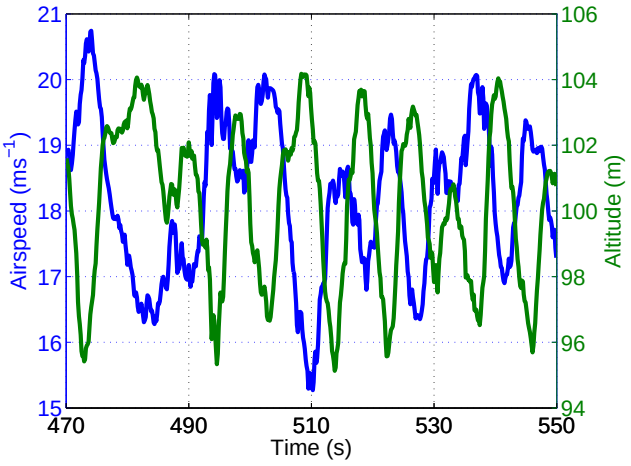


FIGURE 3.36: Altitude and airspeed of the Trainer 60 during flight test 20

in the flight data, though the ∼2 m overshoot may have been obscured by the noise sources.

The mid-level airspeed control response itself (Figure 3.33(a)) appears sufficiently stable for safe operation during flight tests, provided there is sufficient buffer between the variance observed and the aircraft's stall speed. However, the performance falls short of that suggested during simulation (Figure 3.16(a)) even taking into account the additional noise sources identified. In particular the transient response appears much slower than predicted, with the airspeed remaining too low for the duration of the measured flight. Given the slave elevation angle controller response has been shown to be satisfactory, this behaviour is likely to be due to discrepancies between the airspeed controller gains appropriate for the simulation and those necessary for the real Trainer 60. The elevation angle limits defined in software for the airspeed controller output are set to a relatively small ±9 degrees, to prevent the airspeed controller commanding extreme elevation angles that could quickly endanger the aircraft. Observation of the measured elevation angle during this time shows the controller output was rarely saturated however, again suggesting the key improvement lays in adjusting the airspeed controller gains. Such tuning has yet to be field-tested due to a greater emphasis on path tracking during recent flight tests and the current airspeed control being stable, if unsatisfactory.

The high variance in measured heading rate during control (Figure 3.33(b)) is due to external noise sources rather than a poorly tuned heading rate controller. Such a conclusion is inferred by the similar variance recorded during a gentle turn under full manual control by the test pilot (Figure 3.37). The high frequency noise observed could not be corrected by the heading rate controller since the latency in

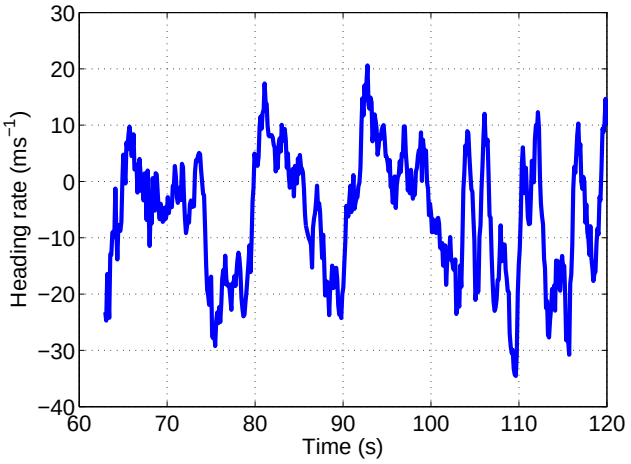


FIGURE 3.37: Measured heading rate during a gentle manual turn of the Trainer 60 during flight test 9

TABLE 3.6: Mean absolute errors of mid-level controller response recorded during steady-state flight of the Trainer 60

Controller	Measurement period (s)	Mean absolute error
Airspeed	405 - 485	3.9 ms^{-1}
Heading rate	35 - 85	8.1 degrees s^{-1}

adjusting the aircraft's heading rate by bank angle and in turn aileron deflection is too great. The controller has however managed to hold the mean heading rate to only 1.0 degree s^{-1} from that commanded during the 57 second sample in Figure 3.33(b). With reference to Figure 3.33 MAE values for the mid-level control are shown in Table 3.6.

The high-level heading controller response is shown in Figure 3.34 to have a similar response rate to the simulation results (Figure 3.17). During the steady-state period between 67 and 78 seconds, the MAE from commanded heading is 3.5 degrees. A longer period of autonomous heading hold would have been preferred, though the small size of the site used during flight test 14 limited the time the aircraft could fly on a constant heading before considered out of the site limits. Though attempts were made to change the heading command to keep the aircraft within the boundary, the test pilot was required to take control before a sufficient turn had been achieved. The controlling action of the heading controller is however quite apparent in the flight data. Since path following would be unlikely to use the heading hold controller itself, the recorded response was considered satisfactory.

The flight data taken for heading rate and heading controller analysis has come from earlier test flights due to the emphasis on path tracking performance in the more recent flight tests. Since heading rate and heading control is not used during this flight configuration, no subsequent collection of such data has been recorded. The gains for both controllers however remain unchanged and as such the earlier data remains valid.

The conclusions drawn from the flight data analysis have assumed the estimated states from the AHRS and pressure calculations accurately reflect the true aircraft states. Since no independent method of logging these states is available, trust in these estimates is based upon direct measurements recorded by autopilot hardware in the laboratory environment, the encouraging simulation results for bank and elevation angle estimates and visual confirmation from the ground observers. The latter represents the most direct evaluation which, while accurate quantitative assessment is not possible, trends in state values have been reasonably correlated to those reported by the autopilot. Accurate assessment of practical flight test data is further hampered by the unpredictable flying environment, requiring longer state observations under varied flight conditions to separate internal estimation and control error from external disturbance.

The flight data recorded by the ground station and used for all analysis has been low-pass filtered by the autopilot prior to transmission, as described in subsection 3.3.3. The data stream FIR filter has a -6 dB gain at 3.5 Hz. While the transient response of all the controllers is not expected to contain significant components above this frequency, it is possible some relatively high-frequency oscillations in the airframe would be notably attenuated in the flight data. Visual observation of the aircraft in flight have not however identified any such oscillations. High-frequency noise such as electrical noise or vibration due to the engine would also be mostly removed from the flight data stream, though such noise is of little interest in assessing and tuning the autopilot control response.

Overall the current control scheme and choice of gain values is considered adequate for fully autonomous flight under local test conditions. Such a flight was achieved for 70 seconds at the end of flight test 20, with the autopilot controlling all control surfaces and throttle to attain a constant airspeed, altitude and bank angle. The flight data suggests further tuning of the altitude, airspeed, heading rate and heading controllers is recommended.

3.6 Conclusion

A complete simulation model of the Trainer 60 has been created with aerodynamic, inertial and engine components. The model has been integrated with Simulink to produce a modular simulation environment arranged as a clear schematic. The separation of core autopilot algorithms from hardware interface code has allowed identical autopilot functionality on both simulation and hardware platforms. This allowed the rapid code development and debugging of both autopilot and ground station software due to the ease of data logging, repeatability and controlled environment of the simulation. The same features enable the manual tuning of controller gains to the simulated Trainer 60 in a risk-free environment.

A custom daughter board was produced to extend the performance of the O-Navi avionics hardware. Key features of the daughter board were the increase in altitude and airspeed resolution deemed necessary for robust control, a non-volatile memory and the ability to switch between manual and autopilot-controlled actuator signals. The switching hardware added the important safety feature of an emergency manual override that operates independently of the autopilot and ground station. The COTS Trainer 60 model aircraft required only minor modifications to act as a suitable platform for autopilot testing.

Simulation results demonstrated the novel use of PDF controllers for UAV control. Acceptable control response was obtained for all controlled states following manual gain tuning using a trial and error technique. Real flight tests showed improved performance for altitude control by throttle and airspeed control by elevation angle as opposed to the airspeed by throttle and altitude by elevation angle. This choice was also considered safer, since the autopilot would be less likely to stall the aircraft if a rapid increase in altitude was required or the engine failed.

EKF and complementary filter algorithms operate in series to provide AHRS and PRS functionality. The EKF's ability to estimate orientation and gyroscope bias was verified in simulation prior to real flight testing. A custom application to generate optimal matrix calculation code allowed the EKF iteration to be calculated sufficiently quickly by the embedded processor. The use of the *Joseph stabilised version* for updating the EKF error covariance increased the algorithm's stability. The complementary filter estimates were also verified by simulation. The filter's ability to continue satisfactory position estimates in the absence of GPS lock and provide a rapid correction once the lock is reacquired was demonstrated.

A custom communications protocol was developed to link the autopilot and ground station. Digital FIR filters are employed by the autopilot software to minimise aliasing when streaming data to the ground station client, given the autopilot's state update rate exceeds the available communications bandwidth. The ground station consists of a network server and UAV operator client software. The ability to interface with the UAV over a network removed the need to run the client software on the same computer that was physically attached to the wireless modem. The potential for multiple client connections to a single autopilot also increased the system's flexibility. The UAV operator software was developed in-house to ensure access to all the features of the custom autopilot and support the expected oceanographic mission requirements. A GUI was provided with data presentation and autopilot instruction aimed at the non-expert user.

Practical flight test results broadly agreed with those suggested by the simulation, though additional unmodelled noise sources were present. The successful operation of a complete UAV system has been recently demonstrated with the autopilot taking full control of the aircraft for 70 seconds, fulfilling key research objectives. Further mission-specific autonomous control is demonstrated in Chapter 5. Analysis of the flight data supports the need for further tuning of the controller gains, in particular to compensate for differences between simulated and real Trainer 60 behaviour. A novel tuning technique designed to achieve this is the subject of the next chapter.

Chapter 4

Controller Tuning

4.1 Introduction

The literature review identified a manual trial and error technique as the typical method of tuning controller gains, even for the industry leading Piccolo and MicroPilot autopilots (Section 2.6.1). The review identified some algorithmic tuning techniques for both PID and PDF controllers [137–139, 147], though none were found to be satisfactory alternatives for this UAV application due to their reliance on an accurate transfer equation or response measurement. While manual tuning provided an adequate controller response for the simulated and practical flights of the Trainer 60 (Sections 3.3.4 and 3.5) it is still not considered optimal. Difficulties are two-fold; the effect of PDF gain adjustment on the response is not as predicable as PID (Section 2.6.1) and practical issues impair the in-flight gain tuning of the real aircraft. Practical issues stem from the requirement to memorise many previously observed responses and associated gains while seeking the best settings. Such problems are countered during simulation by the improved off-line analysis tools, such as multiple graph overlays, not present on the ground station. In addition, low-frequency noise during real-time flight observation can impair visual judgement of the control response. Manual tuning also requires an expert operator with experience in PDF gain tuning, potentially restricting UAV tuning opportunities.

An automated technique was therefore identified as a key research objective, with the ability to quantify controller optimality and iteratively tune the gains in real-time. Such a technique is presented based upon the calculation of the mean squared error (MSE) between the ideal and observed response to a commanded step change.

This MSE is a measure of optimality. In this method the ideal response is modelled as a linear response which transitions to a normal cumulative distribution function (CDF). The simulation environment and Trainer 60 model described in Section 3.1 were used to generate maps of optimality versus PDF controller gains for specific manoeuvres. These maps are termed "optimality maps".

Whilst it is possible to generate the large number of step responses required for production of a high resolution optimality map for visual estimation of the optimal gain pair in simulation, this is impractical in real flight tests. Fortunately the characteristics of the optimal regions identified by simulation enable a two-dimensional maximisation algorithm to discover the optimal maxima using considerably fewer step responses. The technique is therefore practical for real-time controller tuning. A simulation of the Trainer 60 under realistic flight conditions was used to assess the tuning algorithm's performance. Results are presented using an autopilot with a response delay expected of the actual hardware, and with an extended delay. The latter provides a simple test of the algorithm's ability to adapt to differing conditions.

Further practical implementation aspects are investigated including using the computational resources of the ground station computer, linear interpolation to reduce artefacts due to under-sampling and the effect of wind and state estimation errors. The algorithm is demonstrated by tuning the real Trainer 60's bank angle controller during a practical flight.

4.2 Generation of the Ideal Response

Observation of well-tuned aircraft dynamics suggests the ideal response to a commanded step change can be modelled by a linear section transitioning to the latter half of a normal CDF. The transition occurs once the response reaches halfway between the initial and commanded values. At this point the gradient of the linear section and the CDF function are equal resulting in a smooth transition.

Thus the ideal response can be defined simply by mean μ and standard deviation σ. In this context, given a continuous time domain, μ would represent the time at which transition occurs. σ provides an indication of the response rate. For the discrete time domain, the ideal response y_i at discrete sample index i is shown in

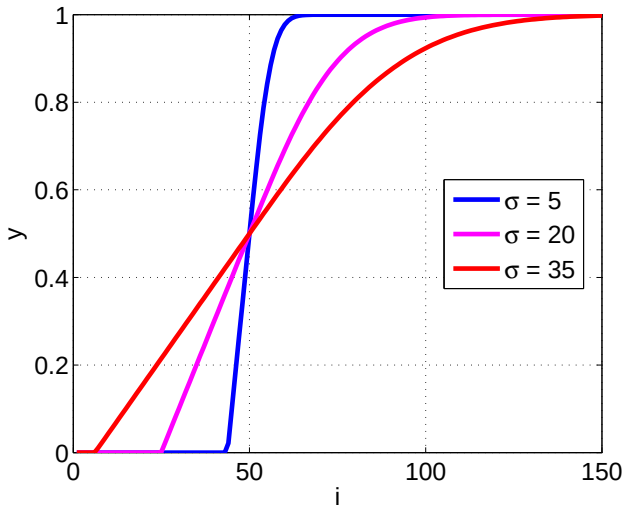


FIGURE 4.1: Ideal response with varied σ

Equation 4.1. Figure 4.1 illustrates the effect of varying σ with fixed μ.

$$y_i = \begin{cases} 0 & i \leq \mu - \dfrac{\sigma\sqrt{2\pi}}{2} \\ \dfrac{i-\mu}{\sigma\sqrt{2\pi}} + \dfrac{1}{2} & \mu - \dfrac{\sigma\sqrt{2\pi}}{2} < i \leq \mu \\ 1 - \dfrac{1}{2}\operatorname{erfc}\dfrac{i-u}{\sigma\sqrt{2}} & \mu < i \end{cases} \tag{4.1}$$

The components of this ideal response model emphasise a rapid and stable transition with a smooth round-out at the commanded value. Maintaining the linear response right up to the command value would increase the chance of overshoot by ignoring the finite transition time required by a real system to change from a non-zero to zero response gradient. A step response would be a similarly unsuitable model for a real system and would almost certainly lead to overshoot. The linear section at the start of the response (rather than an extension of the normal CDF) encourages a rapid initial response. Ideally no oscillation should occur once the response reaches the commanded value, shown by the ideal response's horizontal steady-state condition.

Simulations suggest a too slow response fails to match the initial linear transition. A too fast response will overshoot and oscillate, also failing to match the ideal model. Therefore though σ is unconstrained, it is believed a given controller can only generate responses to match a narrow range of σ and that this will occur during its best response.

Though the following algorithms assume this ideal response model, the optimisation technique itself may be used for any model whose parameters can be fitted to the measured response. Alternative models may be sought to match control criteria outside the UAV application described.

4.3 Fitting Ideal Response to Measurement

Only the shape of the ideal response is known *a priori*. To calculate its magnitude, slope and location in time the ideal response is fitted to real data. After normalisation and filtering (which reduces the effects of noise) the start, middle and endpoints of the region of the response used for fitting are identified using threshold crossing. Fitting of the linear and CDF sections to the measured data is then achieved using transformation and linear regression.

Since the optimality measurement is based upon the mean squared-error, normalisation of the measured data prior to fitting permits fair optimality comparisons between controlled states. Given the original command step was from s_0 to s_1, normalised measurement m is produced from original data d at sample index i by Equation 4.2. The normalisation to a unit step also simplifies subsequent equations.

$$m_i = \frac{d_i - s_0}{s_1 - s_0} \tag{4.2}$$

The division of measured data into linear and normal CDF regions is achieved by threshold crossing. As a result, it is recommended to low-pass filter the measured data m to produce $\hat{m}$. For this research a 32^{nd} order FIR filter was used. The filter may have a cut-off frequency below the highest frequency of interest since the mean squared-error calculation will be performed on m. High frequency oscillations caused by the controller will be of interest during tuning, but may be safely filtered out for threshold detection.

The region where $\hat{m}$ lays between the noise threshold at both the floor and ceiling of the unit step is the region used for response fitting. The normalised noise threshold τ is converted from original data units τ_d by Equation 4.3. The sample index range for the linear (i_{start} to i_{tran}) and normal CDF (i_{tran} to i_{end}) sections are determined by Equation 4.4. The use of thresholding permits the algorithm to operate without knowledge of the step command execution point, in addition

optimality becomes insensitive to controller delay.

$$\tau = \frac{\tau_d}{|s_0 - s_1|} \tag{4.3}$$

$$\begin{aligned} i_{start} &= i \qquad \hat{m}_{i-1} \leq \tau < \hat{m}_i \\ i_{trans} &= i \qquad \hat{m}_{i-1} \leq 0.5 < \hat{m}_i \\ i_{end} &= i \qquad \hat{m}_i \leq 1 - \tau < \hat{m}_{i+1} \end{aligned} \tag{4.4}$$

Linear regression is used to determine the best fit of the ideal response to the measured data. The normalised measured response m is linearised over the region of interest to produce l by Equation 4.5.

$$l_i = \begin{cases} \left(m_i - \dfrac{1}{2}\right)\sqrt{\pi} & i_{start} \leq i < i_{trans} \\ \text{erf}^{-1}\left(2m_i - 1\right) & i_{trans} \leq i \leq i_{end} \end{cases} \tag{4.5}$$

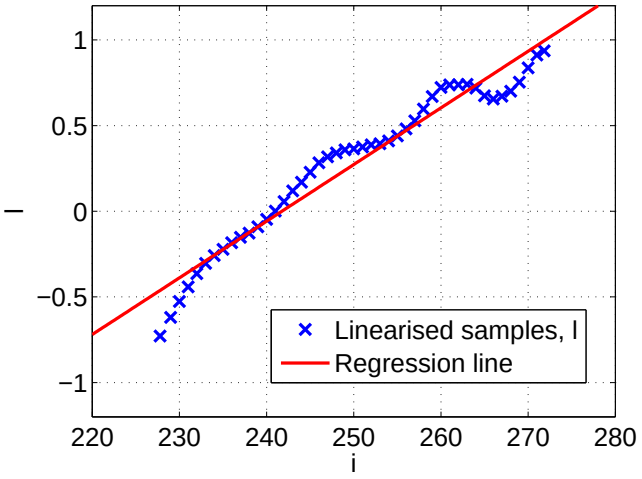
Standard linear regression is performed for l against i to produce a best fit line with gradient G and l-intercept C. Equation 4.6 relates these parameters back to estimated $\tilde{\mu}$ and $\tilde{\sigma}$.

$$\tilde{\mu} = \frac{-C}{G} \qquad\qquad \tilde{\sigma} = \frac{1}{G\sqrt{2}} \tag{4.6}$$

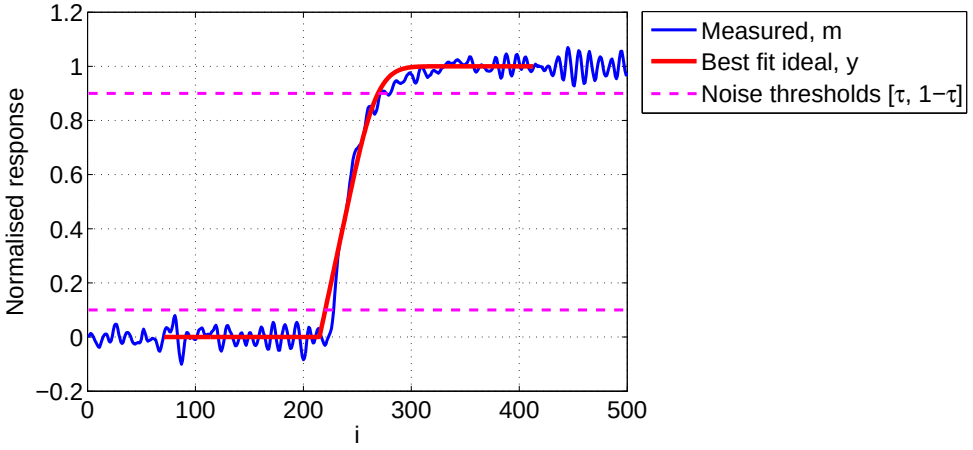
An example of an ideal response matched to the measured response from real flight test data is shown in Figure 4.2.

4.4 Calculating and Mapping Optimality

While the coefficient of correlation could be used as an indication of optimality, the region of interest within the measured response extends beyond the fitting region (i_{start} to i_{end}). Therefore the mean squared-error will be taken, bounded by $\tilde{\mu} \pm 8\tilde{\sigma}$. The use of $\tilde{\sigma}$ for range setting permits a similar proportion of transition and steady-state samples over the region of interest, improving the likelihood of a fair comparison between differing responses. Additionally, the number of discrete measurement samples used for the MSE scales with sampling rate without user intervention. Optimality has been chosen as the inverse of the MSE primarily due to the increased sensitivity this provides around the optimal region (due to smaller



(a) Linearised data



(b) Ideal response

FIGURE 4.2: Ideal response matched to test flight data

magnitude MSE). This improves the visual representation of the optimality maps when plotted as a contour map. Optimality is expressed in Equation 4.7.

$$\text{Optimality} = \frac{i_{last} - i_{first} + 1}{E} \tag{4.7}$$

where:

$$i_{first} = \left\lfloor \tilde{\mu} - 8\tilde{\sigma} + \frac{1}{2} \right\rfloor$$

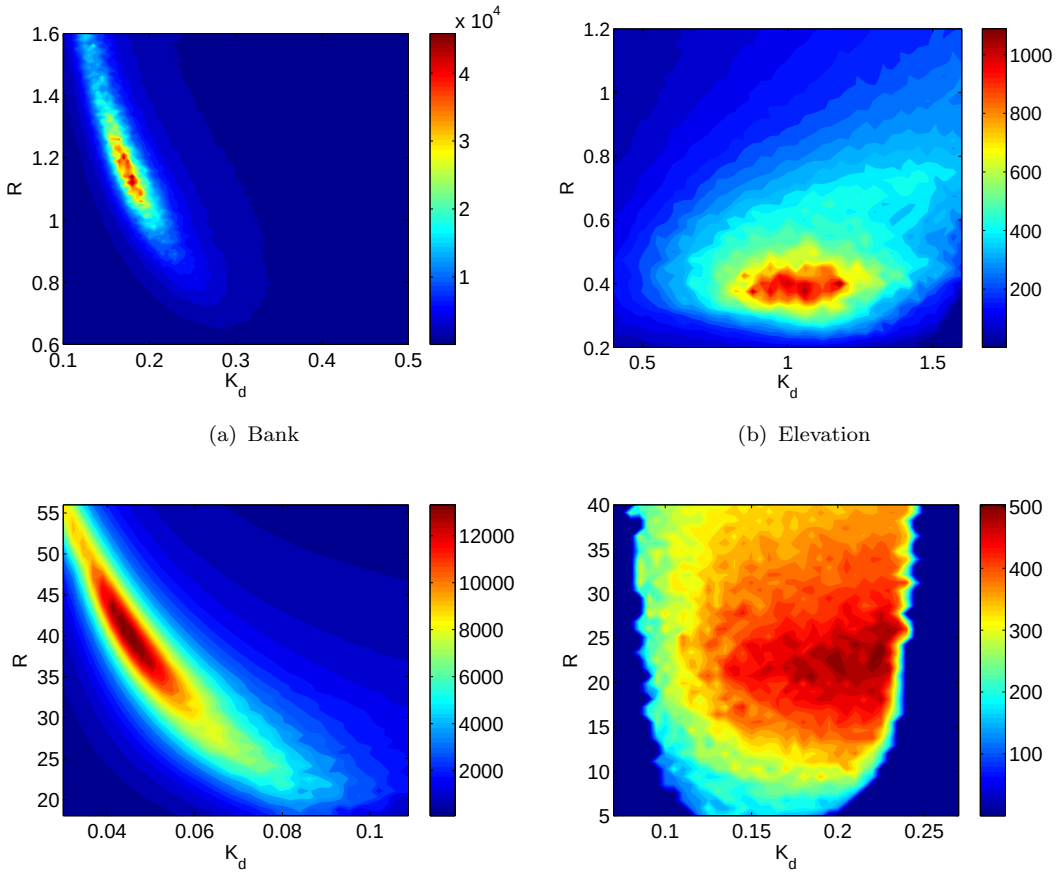
$$i_{last} = \left\lfloor \tilde{\mu} + 8\tilde{\sigma} + \frac{1}{2} \right\rfloor$$

$$E = \sum_{i=i_{first}}^{i_{last}} (m_i - y_i)^2$$

A map of optimality for any given controller may now be produced by varying the two controller gains and recording their associated response optimality. Providing there is sufficient resolution, this map should highlight any regions of high optimality and the resulting sets of optimal gain pairs. Due to the relationship between PDF controller gains K_d and K_i, it is more useful to examine the ratio $R = \frac{K_d}{K_i}$ along with K_d.

A simulation of the Trainer 60 was run with minimal environmental influence (zero wind) and all controllers (except that being tuned) commanded to hold the aircraft at cruise conditions. A C-code MEX file was written to directly interface with the simulation. This program varied K_d and R, requested a step change in commanded state and measured the response optimality. The commanded state was stepped back to its original condition and a second response optimality measured, with the average of the two optimalities being associated with the gain pair. This average was taken to reduce the effect of noise and to unbias the optimality measurement from systematic response behaviour associated with the direction of step change itself (e.g. going with or against gravity). The gains were incremented in equal spaces along a grid of predefined size and resolution to form the optimality map. The optimality maps produced for the three low-level controllers (bank, elevation and altitude hold) and a mid-level controller (airspeed hold) are shown in Figure 4.3. The aircraft response was taken from the EKF state estimates based upon realistic sensor models (Section 3.1.4). These sensors have some associated noise and bias and in turn there is some error present in the state estimation. Since these values will be all that is accessible to any real controller tuning algorithm, the majority of analysis was performed using this data. A comparison of bank controller optimality maps produced from estimated and true states is provided in Section 4.7.6.

The optimisation measurement algorithm appears sufficiently robust to have calculated values over the complete range of gains tested, with considerably different response characteristics. The algorithm required no modification when switching between controllers, except to state the two set point values and expected noise level (in measured units). All the optimality maps produced a single region of high optimality, with a general increasing trend towards it. As expected given the differing aircraft dynamics for each controlled state, each controller produced a unique contour shape and location of high optimality. The mid-level controller, airspeed, produced a significantly different optimality map compared with the low-level controllers (Figure 4.3(d)). The map shows clearly-defined K_d boundaries,



(a) Bank

(b) Elevation

(c) Altitude

(d) Airspeed

FIGURE 4.3: Optimality maps for the simulated Trainer 60's PDF controllers

and a small integral component (due to high R). It is most likely due to coupling between the airspeed controller and its slave elevation controller. Since it is the elevation controller that ultimately adjusts the aircraft's airspeed, as soon as the airspeed controller's response becomes faster than the elevation controller can achieve, oscillations in airspeed quickly become apparent. In addition, there would be a greater lag in the feedback loop between measured airspeed and airspeed controller output, favouring the smaller integral component.

Though noise introduced by the use of state estimation from noisy sensors is apparent as an irregularity in the contour plot, it has small enough amplitude not mask underlying trends. Therefore, the use of a maximisation function to seek the optimal region would be practical.

4.5 Maximisation Function

The optimality maps in Figure 4.3 each required over 3300 simulated step responses. To produce maps of a similar resolution under test flight conditions would be highly impractical. A promising alternative lays with an iterative maximisation algorithm known as Dynamic Hill Climbing (DHC) [187, 188]. The algorithm has demonstrated a robust and efficient performance against a suite of test cases, outperforming a number of genetic and adaptive simulated annealing algorithms. The authors provide a C-code implementation [188] though this has been modified to provide separate initialisation and single iteration step functions. The research presented implements *Procedure 2-1*, composed of the global random start point selection *Procedure 2-2* and local optimisation *Procedure 4-1* [188]. Since the local optimisation locates the minima, the sign of the optimality result is inverted prior to its use in the DHC iteration. A generic C-code header file was written and incorporated in an additional MEX file during simulation and directly into the ground station software used for practical flight testing.

The bank angle controller was selected for optimality maximisation as it would be the first test during a practical flight. This is due to the state's fast response and high tolerance to minor changes in flight conditions. To increase confidence that a local maxima is a global maxima, multiple start points were used (as required by *Procedure 2-1*). As a compromise between reduced response count and maximisation confidence, five start points were used in the simulation. Similarly the step threshold (the tolerance in K_d and R before a local maxima is considered) was set to 0.01. While the airspeed and altitude optimality maps suggest a marked difference in acceptable K_d and R resolution, the C-code algorithm provided in [188] only accepts a scalar step threshold. Though [187] demonstrates a step threshold vector implementation in LISP, its implementation in C is considered future work.

Only one optimal gain pair will be sought for the bank angle controller. Since the autopilot does not support gain scheduling the selection of multiple optimal gain pairs for differing flight conditions is not required. The variance of such conditions during normal flight operation is considered sufficiently small to justify this simplification.

The maximisation start and end points for each of the five runs is shown superimposed on the bank optimality map in Figure 4.4 and summarised by Table 4.1. Figure 4.5 illustrates the before and after optimisation bank angle responses for each run.

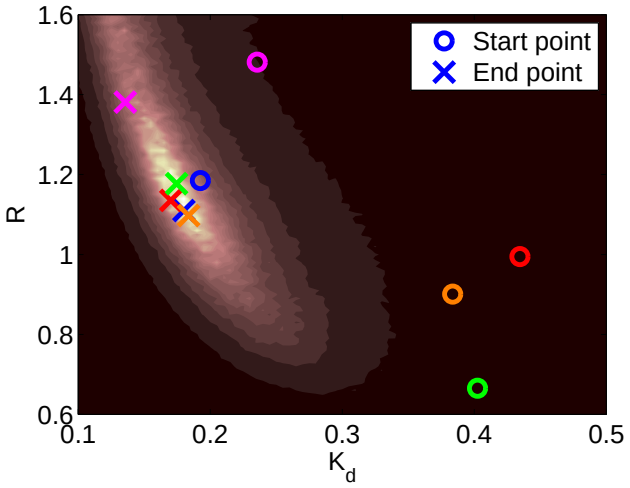


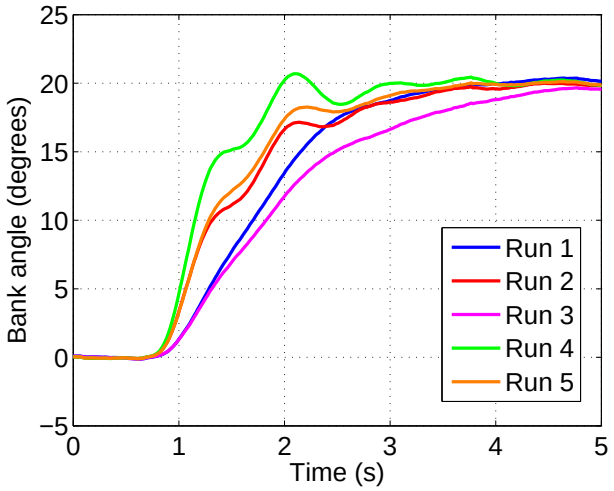
FIGURE 4.4: Start and end points for five bank angle optimisation runs

TABLE 4.1: Summary of simulated bank optimisation runs

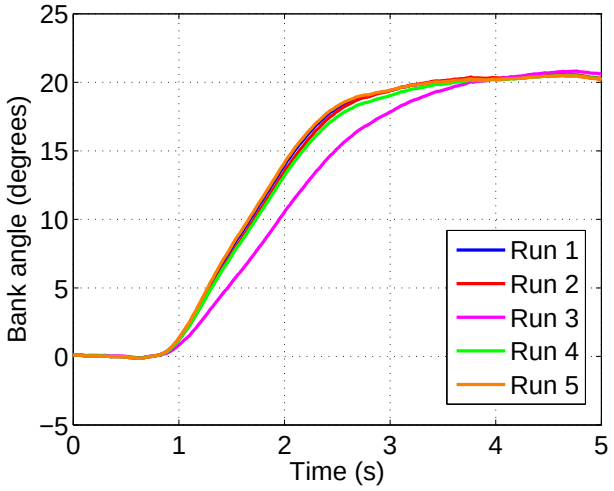
Run	Iterations	Duration (m:s)	Optimality Start	Optimality End
1	26	20:58	13708	43487
2	51	39:27	846	35078
3	22	18:34	1464	17591
4	53	41:43	575	40698
5	37	29:24	971	39836

The bank angle response was improved in all optimisation runs, observed both by higher optimality measurement and visual inspection of Figure 4.5. Though the indicated maxima did not converge on a single point, the close clustering of four out of the five end points suggests they are within the region of highest optimality. This is supported by visual comparison.

The first simulations suggested each run would take 20–40 minutes to complete in real flight tests. While this is within practical limits, the aircraft under test can only carry fuel for approximately 20 minutes flying time. Though the DHC implementation used can be paused and restarted on command (e.g. after refuelling) it is still preferable to complete an optimisation run per 20 minute flight test. This tactic reduces environmental disturbance as flight conditions (such as wind) change. The first simulations took the average of two bank optimality readings, as performed for mapping. However, in the case of bank angle this may be reduced to a single reading as aircraft bank response is very similar whether banking to port



(a) Before optimisation



(b) After optimisation

FIGURE 4.5: Before and after bank angle optimisation for each run

or starboard. This tactic halves run time with only a small increase in sensitivity to noise. This was confirmed by simulation resulting in very similar optimality locations whilst run time was little over 19 minutes (the shortest run completed in under 8 minutes). As such, the bank optimisation for the practical flight test would use this configuration.

4.6 Delay Compensation

The standard simulation model assumes a 50 ms delay between the true aircraft state and the autopilot's controller input. This delay is mainly due to the onboard

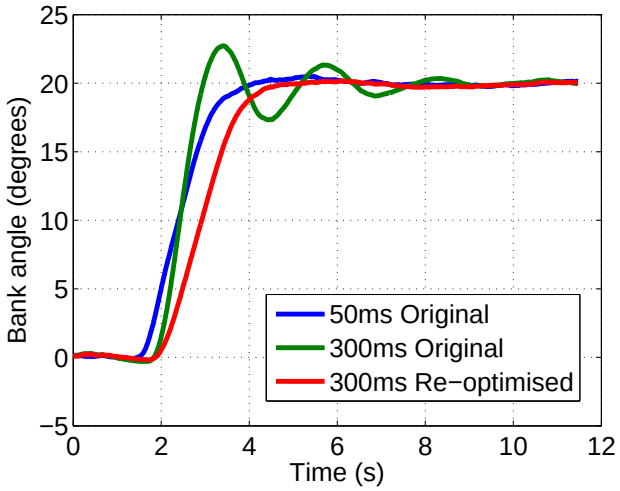


FIGURE 4.6: Optimisation with 50ms and 300ms input delay

TABLE 4.2: Optimality results for standard and extended input delay

Delay (ms)	Optimal K_d	Optimal R	Optimality
50	0.1801	1.110	43487
300	0.1176	1.683	18564

digital sampling, filtering and state estimation. To test the optimisation algorithm's ability to adapt the controller gains to differing conditions, the state delay was increased to 300 ms. This led to marked oscillations in the simulated aircraft states using gains tuned when the delay was 50 ms. Bank angle was again chosen to demonstrate the optimisation. Five optimisation runs were performed under the increased autopilot delay. It was found the optimisation boundaries required a shift to incorporate the new high optimality region, but all remaining settings were as before. The gains chosen by the run ending with the highest optimality were used for comparison. The results are summarised in Figure 4.6 and Table 4.2.

It is apparent that the optimisation algorithm successfully re-optimised the bank controller gains to the new conditions. The optimal K_d and R are both lower for the extended delay case. This was expected as it results in a slower response from the bank angle controller, reducing the tendency to oscillate in the presence of a lagged input.

4.7 Practical Considerations

4.7.1 Computation

The iterative optimisation algorithm presented may be run either on the autopilot itself or at an external ground station. The ground station must be capable of receiving real-time aircraft state information and uploading command set points and controller gain values; this would be expected functionality for the majority of ground station software. Two benefits are then introduced; the autopilot software itself requires no additional modification, and the extra processing power required to implement the algorithm may be taken up by the ground station computer. Since this computer would typically have a greater degree of spare computational capacity, for practical implementation it is suggested the algorithm be executed in this manner.

The complementary error function required in Equation 4.1 is implemented in C using a Chebyshev fitted approximation [189]. The inverse error function used for Equation 4.5 and for linear interpolation (Section 4.7.4) is implemented in C using a rational approximation based upon code by Acklam [190].

4.7.2 Flight Testing

The optimisation sequence may be paused at any point between successive iterations. This allows the repositioning of the aircraft (manually or otherwise) or even landing for refuelling to take place when necessary during an active optimisation run. Wherever possible flight conditions should be matched between successive iterations and runs.

When setting commanded set points, it is recommended they be placed on either side of typical cruise conditions to allow the aircraft to continue uninterrupted for as long as possible. For bank controller optimisation, it is suggested alternating between matched turns to port and starboard would also use the least airspace - of particular importance when maintaining visual contact during practical flight tests. Though the simulated aircraft had unlimited fuel and airspace, care was still taken when determining the set points to maintain the very long flight periods under constant conditions required to generate the optimality maps.

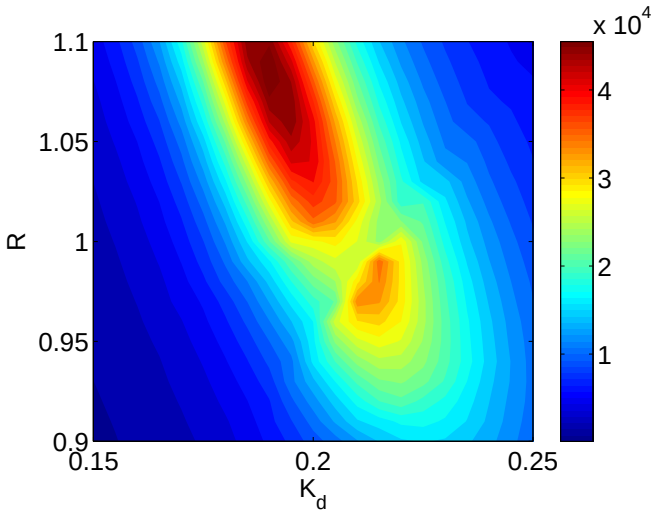
4.7.3 Filtering

During response linearisation noise becomes distorted by the non-linear inverse error function in Equation 4.5. A positive measurement offset will increase the linearised value more than the same magnitude negative offset would decrease it. This distortion means Gaussian noise can bias the regression result and in turn $\tilde{\mu}$ and $\tilde{\sigma}$ (Equation 4.6). It is therefore recommended sufficient low-pass filtering be performed on d (Equation 4.2) to remove the typically high-frequency signal components not associated with the response of the aircraft itself. The cut-off frequency for this filter would be higher than the filter already suggested for threshold crossing since in this case no frequencies of interest should be lost. Such filtering is already performed by the autopilot to maintain Nyquist criteria when streaming state data to the ground station (Subsection 3.3.3).

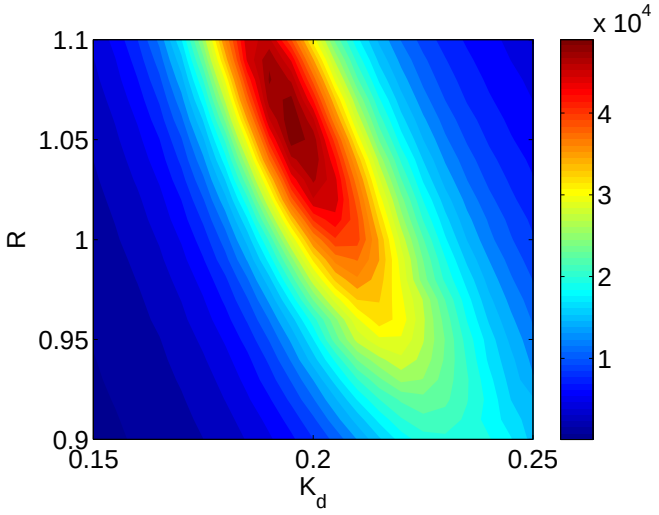
For additional protection any measured value of $0.02 < m_i > 0.98$ is ignored during linearisation when $i \geq i_{trans}$ due to the particularly high sensitivity of erf^{-1} in this region. Since the normalised noise threshold (Equation 4.3) would typically be greater than 0.02, such measurements should be very rare after low-pass filtering.

4.7.4 Linear Interpolation

Should the sample period approach a significant percentage of the controller response transition period, the fitting range (i_{start} to i_{end}) will be small. Such conditions may occur with a low data rate between the aircraft and ground station (assuming the optimisation algorithm is performed by the ground station) and the optimisation of a state with a fast response (such as bank angle). The smaller the fitting range, the greater the significance a single sample has on the regression calculation. Since the indexes are chosen by thresholding (Equation 4.4) small changes in response characteristic can trigger a unit change in index value and a resulting shift in the number of samples used for fitting. Observation of the optimality maps under these conditions show a distinct artefact along the boundary where the number of samples changes by one. Simulation of bank angle optimisation with measured data passed to the algorithm at the actual data rate of the wireless modem shows this effect in Figure 4.7(a). The extra local minima produced will decrease the chance of a maximisation run selecting the true point of highest optimality, making the reduction of this boundary effect important for both methods of optimisation.



(a) Without linear interpolation



(b) With linear interpolation

FIGURE 4.7: Use of linear interpolation to correct optimality map artefacts

A solution is found by using linear interpolation to position an extra sample at the exact threshold crossing points prior to regression. The fractional position between neighbouring sample indexes is calculated as the exact threshold crossing times λ (extending back from i_{start}) and ρ (extending after i_{end}). Each fraction is used to weigh the measured samples on either side to produce the interpolated sample point value. For the end sample, the threshold boundary and sample values are linearised in the same manner as Equation 4.5 to retain the validity of a linear interpolation. Given sample indexes $a = i_{start} - 1$ and $c = i_{end} + 1$ the fractional weights are calculated by Equation 4.8 and new sample values l_a, l_c calculated by

Equation 4.9.

$$\begin{aligned} \lambda &= 1 - \frac{\tau - \hat{m}_a}{\hat{m}_{a+1} - \hat{m}_a} \\ \rho &= \frac{\text{erf}^{-1}(1 - 2\tau) - \text{erf}^{-1}(2\hat{m}_{c-1} - 1)}{\text{erf}^{-1}(2\hat{m}_c - 1) - \text{erf}^{-1}(2\hat{m}_{c-1} - 1)} \end{aligned} \tag{4.8}$$

$$\begin{aligned} l_a &= m_{a+1} - \lambda(m_{a+1} - m_a) \\ l_c &= \text{erf}^{-1}(2m_{c-1} - 1) \\ &\quad + \rho\left(\text{erf}^{-1}(2m_c - 1) - \text{erf}^{-1}(2m_{c-1} - 1)\right) \end{aligned} \tag{4.9}$$

These sample points are inserted into the regression samples at $(i_{start} - \lambda,\ l_a)$ and $(i_{end} + \rho,\ l_c)$. Should the new samples be too close to the original start and end sample points, the original samples are discarded to prevent undue biasing of the regression function. This is achieved by discarding l_{a+1} if $\lambda < 0.5$ and discarding l_{c-1} if $\rho < 0.5$. Following the use of this technique the boundary artefacts are corrected, as shown in Figure 4.7(b). A further simulation was performed under identical conditions except for a ten times faster measurement update frequency and without the use of linear interpolation. The optimality map produced very closely represented the corrected low-frequency update version.

4.7.5 Wind

A simulation to produce a bank angle optimality map was performed with a moderate 8 ms^{-1} northerly wind. The wind was modelled to vary by strength and direction over time in a realistic manner (Section 3.1.3). Following the creation of the map, five optimising runs were performed under the same conditions. The resulting map is shown in Figure 4.8 with optimisation run end points overlaid as black crosses (the start points were unchanged from Figure 4.4). When compared to the nil-wind result in Figure 4.3(a) it is apparent the overall optimality has decreased. Such a decrease would be expected given the extra disturbance caused by the varying wind. The map is seen to contain many more local maxima, caused by the increased variation in measured optimality caused by factors other than controller gain. While these local maxima would disadvantage the maximisation algorithm, the overall shape of the optimality map appears similar to before - suggesting the underlying controller response remains apparent. Upon running the optimisation algorithm under these conditions, the run end points have a wider distribution. However, even with the considerable external disturbance they remain clustered around the true high optimality region. The two maximisation

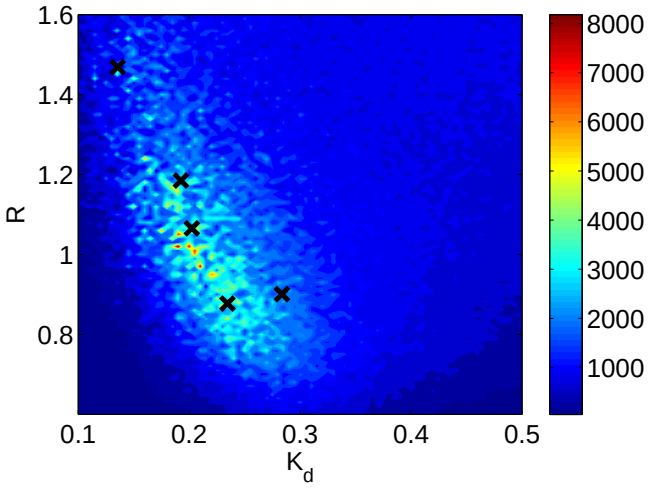
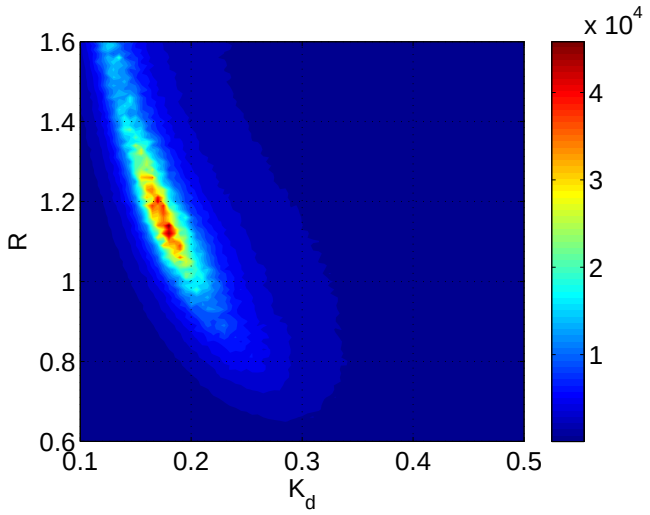


FIGURE 4.8: Bank angle optimality map with variable 8 ms^{-1} wind

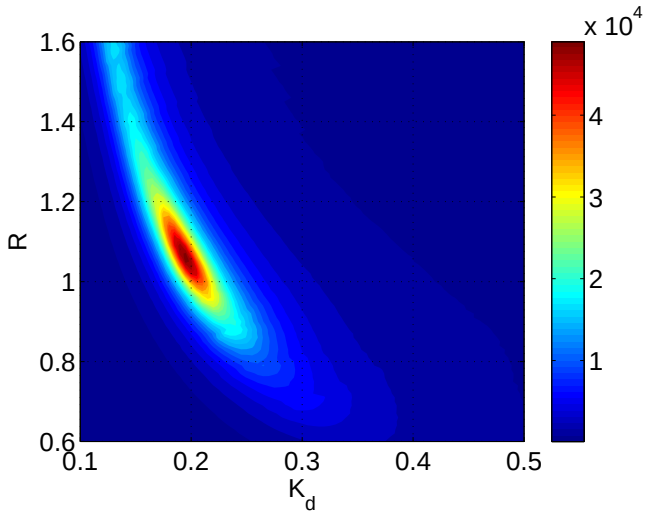
runs with the highest optimality (near the line of $K_d = 0.2$) ended very close to the nil-wind optimal point. While these results confirm increased external disturbance leads to reduced optimisation performance, they also suggest the algorithm is still capable of useful operation even in moderate wind and gusts.

4.7.6 State Estimation

All previous simulations have used the autopilot's state estimation to maintain realistic results. Comparison with the true simulation state is shown in Figure 4.9 using the bank angle controller. While the overall optimality levels are similar, a clear reduction in local maxima is observed. The smooth appearance of the true bank angle optimality map was expected due to the elimination of noise in the model. The similarity in overall shape of both maps suggest the bank angle estimate is relatively close to the true state. The slight shift towards a higher K_d and lower R suggests a faster optimal response was possible when using the true bank. This may be due to the true bank having zero delay, unlike the 50 ms delay used for the estimate, and so the controller is capable of a slightly faster response before significant overshoot or oscillation. This was confirmed in a subsequent simulation with a 50 ms delay added to the true state. The resulting optimality map demonstrated a shift back to an optimal centre visually identical to that of the estimated state.



(a) Using estimated bank



(b) Using true bank

FIGURE 4.9: Optimality maps for estimated and true bank

4.8 Flight Test Results

The optimisation algorithm proposed was incorporated in the ground station client software (Section 3.4.2). A flight test of the Trainer 60 was performed to optimise the bank controller gains. The algorithm parameters were set to those previously suggested by simulation (step threshold of 0.01, no optimality averaging, K_d boundary [0.1, 0.5] and R boundary [0.6, 1.6]). The step command switched between small port and starboard banks ($s_0 = -20°$, $s_1 = 20°$). During the flight it was found the aircraft strayed out from the permitted airspace on regular occasions. This was due in part to the turns requested by the algorithm, but also due to the uncorrected drift due to the prevailing wind. Though the algorithm coped

well with the frequent interrupts, and permitted the aircraft to remain flying to the satisfaction of the supervising pilot, the time required to complete a single maximisation run was therefore longer than expected. As a result, a landing was still required to refuel.

A complete run was achieved after 27 iterations, within the range expected from simulation. Unfortunately the local flying conditions had worsened and the decision made to abandon the remaining four maximisation runs planned. The data from the first run is presented as a contour plot in Figure 4.10. The optimal point of the maximisation run is shown as a black cross, with intermediate steps as black dots joined by a line. The start point is circled. A coarse resolution on the outer search perimeter is seen as a product of the DHC algorithm's dynamic step sizing. The route taken by the algorithm in some cases caused the same gain pair to be requested on multiple occasions, though this was considered advantageous by indirectly providing optimality averaging. By forming a contour plot using these intermediate steps, a low resolution impression of the optimality region is produced to assess the validity of the selected optimal point. Though an increased confidence in the optimality region would have been provided by additional runs, the optimal point does appear to reside in a generally increasing area of optimality. The gain values selected are similar to those suggested by the simulation, though again further runs would be required to validate this position.

The overall optimality is considerably lower than that expected from the simulation data. The wind (approximately 8 mph) would have contributed, though it is likely a number of small additional external disturbances would have been present that were not incorporated on the simulation. It is also possible that the real aircraft dynamics favour a bank response with a behaviour less like the ideal response, as noted for the optimality maps of other simulated states (such as elevation). This would imply the flight test optimisation run remains valid even while the overall optimality values are low. The trend in measured optimality is seen to increase during the maximisation run. A plot of measured optimality and linear regression line against iteration number is shown in Figure 4.11. The peak at the 17^{th} iteration corresponds to the optimal gain pair selected from the run.

4.9 Conclusion

A quantitative measurement of controller response optimality has been developed. This uses the inverse mean squared-error between an ideal and measured response

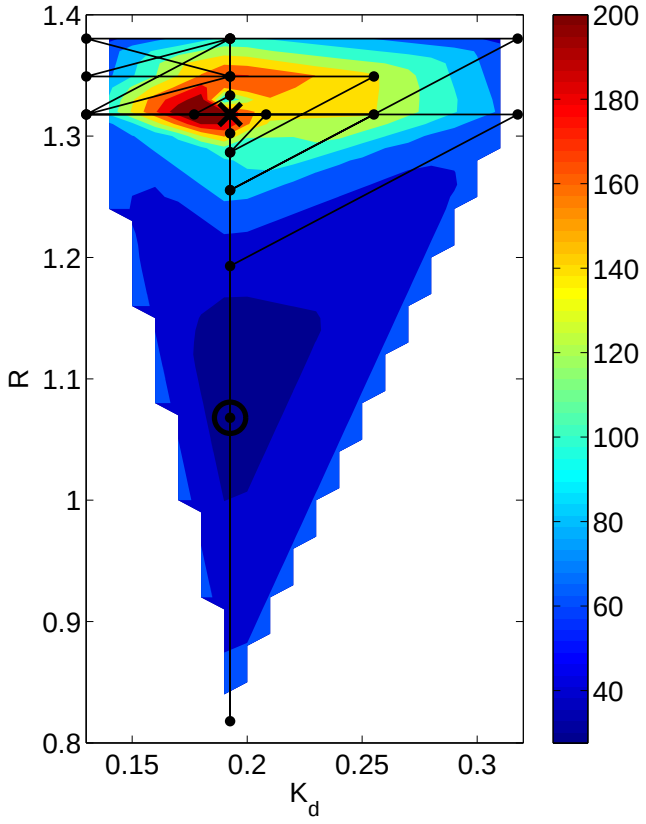


FIGURE 4.10: Contour plot of the flight test first maximisation run

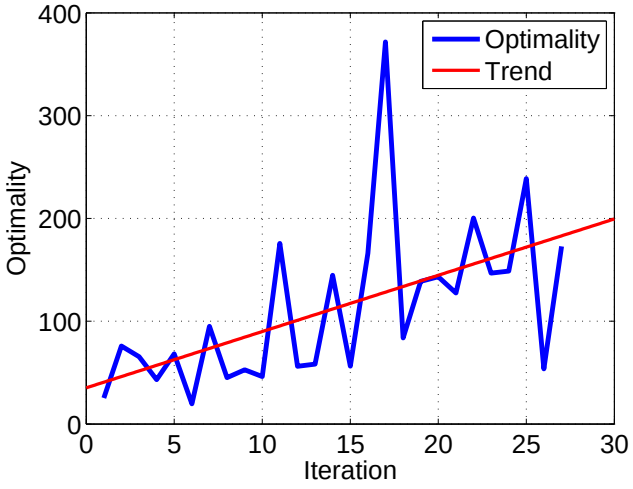


FIGURE 4.11: Optimality against iteration number during the flight test

following a step command. The ideal response is modelled as a linear section transitioning to a normal cumulative distribution function and a method for matching this to the measured data has been presented. Optimality maps produced from simulated flights highlight the effect of controller gain values on response optimality. Observation of these maps showed a single region of high optimality existed for all controllers tested. The Dynamic Hill Climbing algorithm was used to seek this point, bringing the number of measured responses required to identify this region to a practical level for in-flight tuning. This optimisation technique successfully tuned a simulated autopilot with both standard and extra response delay. A number of practical implementation issues were discussed, with the use of the ground station processor for optimality calculations and the need for linear interpolation highlighted as particularly important. The simulation environment suggested wind disturbance reduces optimisation performance though the technique should remain practical under light winds. Results for one optimisation run during a real flight test were presented, with analysis suggesting the controller response had been improved. The successful flight testing of a complete run supports the practical use of the algorithm.

4.10 Further Work

Aside from the further flight tests required to validate the operation of the optimisation algorithm presented, extensions to the algorithm itself are proposed. Though the current application only required the tuning of two gain values, the same technique may be applied to controllers of three or more tuning parameters (such as the popular PID type). Visual representation as a three-dimensional object colour-coded to optimality would be possible using a similar methodology to optimisation mapping. The DHC minimisation algorithm is itself multi-dimensional, enabling the complete optimisation technique to remain valid. The simulation of three-dimensional optimality maps and maximisation runs would however take longer, and further simulation would be required to assess the practical significance of the extra iterations required.

Chapter 5

Path Tracking

5.1 Introduction

An algorithm was sought to fulfil the research objective of reliable and accurate path tracking. Several tracking algorithms were identified during the literature review (Section 2.7), with common approaches being to command ground heading to point at the target or the use of lateral tracking. Though no algorithm was considered to offer the simplicity and reliability required, the performance of the first approach under both simulated and real conditions is provided in the next section for reference. Two novel algorithms are presented in the latter sections, one based upon heading acceleration and the second upon closing speed. The latter is considered to meet the path tracking objective. Additional flight tests demonstrate the autonomous triggering of a payload upon arrival at a waypoint, considered necessary for future missions.

5.2 Direct to Waypoint

A basic tracking algorithm points the commanded ground heading ψ_c at the target waypoint centred at (W_{Tx}, W_{Ty}), as shown by Equation 5.1. The current aircraft position is (P_x, P_y). The `atan2` macro [126] calculates the four-quadrant inverse tangent and is common to many programming languages.

$$\psi_c = \texttt{atan2}\left(P_x - W_{Tx},\ P_y - W_{Ty}\right) + \pi \tag{5.1}$$

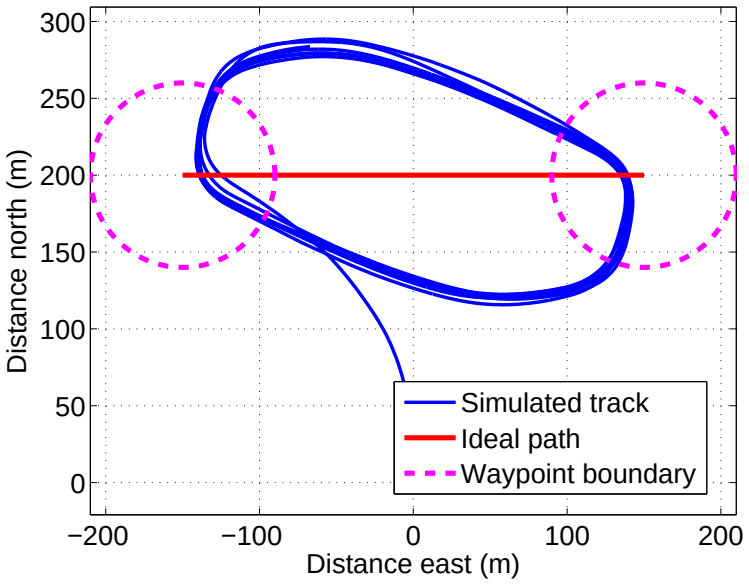


FIGURE 5.1: Simulated track of the Trainer 60 with direct to waypoint tracking

While the aircraft should eventually reach its target, it will not necessarily do so in an efficient or predictable manner. The simulated Trainer 60 track for a two-waypoint path under zero-wind conditions is shown in Figure 5.1. In this case the UAV is considered to have reached the target waypoint once it is less than 60 m from its centre. Predicted position from the PRS was used for this track, though subsequent simulation showed little variance in track shape using true values.

The delay in turning towards the next waypoint following entry into the target boundary causes an overshoot of the ideal path. Such a delay is unavoidable due to the aircraft dynamics and controller action. The algorithm makes no attempt to align the aircraft to the ideal path, resulting in the oval track observed.

An equivalent procedure was performed during flight test 11. Sufficiently light winds were present to approximate the zero-wind conditions simulated. The measured track is shown in Figure 5.2. An increase in track deviation per circuit is observed, most likely due to external factors such as light gusts of wind and thermal activity (the latter effect was reported as particularly strong by the test pilot on what was a hot day). The oval shape predicted by simulation is observed in the real flight data.

The previous flight test had used the same tracking algorithm though under higher wind conditions. The resulting track is shown in Figure 5.3, coloured according to measured ground speed. The 25 knot north-westerly wind during the flight had a marked effect on the aircraft's track. When the aircraft was heading upwind

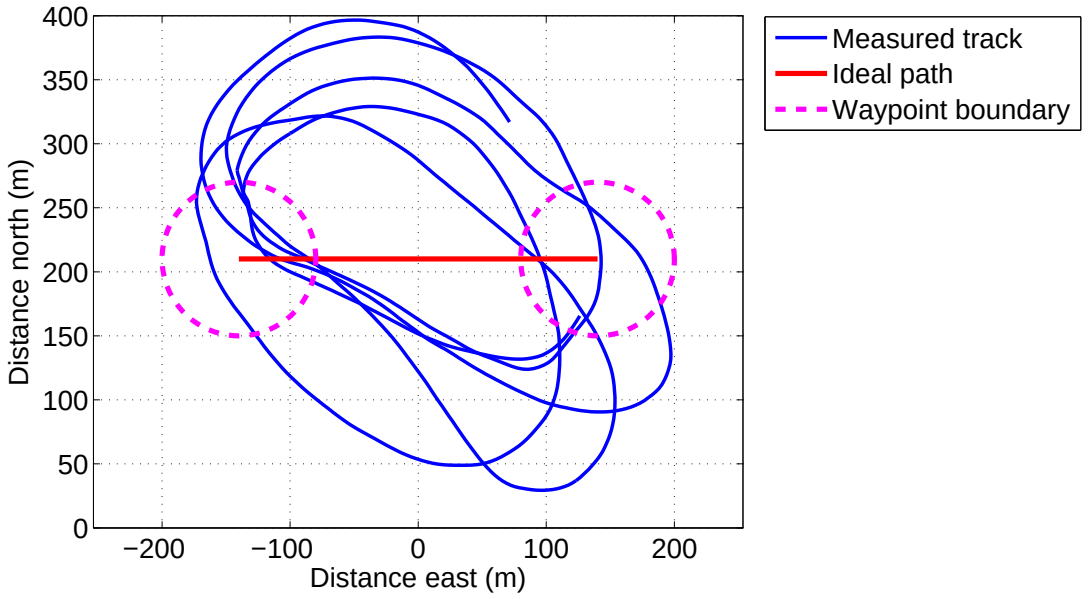


FIGURE 5.2: Measured track of the Trainer 60 with direct to waypoint tracking during flight test 11

(slower ground speed) an oscillatory motion is observed. The downwind path is more similar to the expected zero-wind condition, though the aircraft repeatedly overshoots the downwind waypoint. This behaviour was reproduced in simulation following the addition of similar wind conditions (Figure 5.4).

The oscillation and overshoot were considerably reduced when true rather than predicted ground heading was fed to the tracking algorithm. The cause of this is determined to be the discrepancy between true ground and body-referenced heading when wind is present, as discussed in Section 3.3.5. Ground heading rate is under-estimated by the PRS when turning away from the wind (the aircraft ground track is pushed back, leading to a greater change in ground heading than measured by the gyroscopes) and over-estimated when turning in to wind (the aircraft body turns, but wind continues to carry the aircraft downwind). The former effect is likely to lead to the upwind oscillation observed as the heading controller inadequately counters heading drift. The latter effect causes overshoot of the target waypoint by failing to account for sideslip. Figure 5.5 shows this discrepancy between heading rates during two simulated circuits. Downwind sections are 40–60 and 110–130 seconds.

Path tracking by simply commanding a ground heading towards the next waypoint is considered inadequate. Even if ground heading is accurately estimated, the algorithm makes no attempt to align the aircraft along the shortest straight line between waypoints which leads to an unpredictable and typically inefficient path.

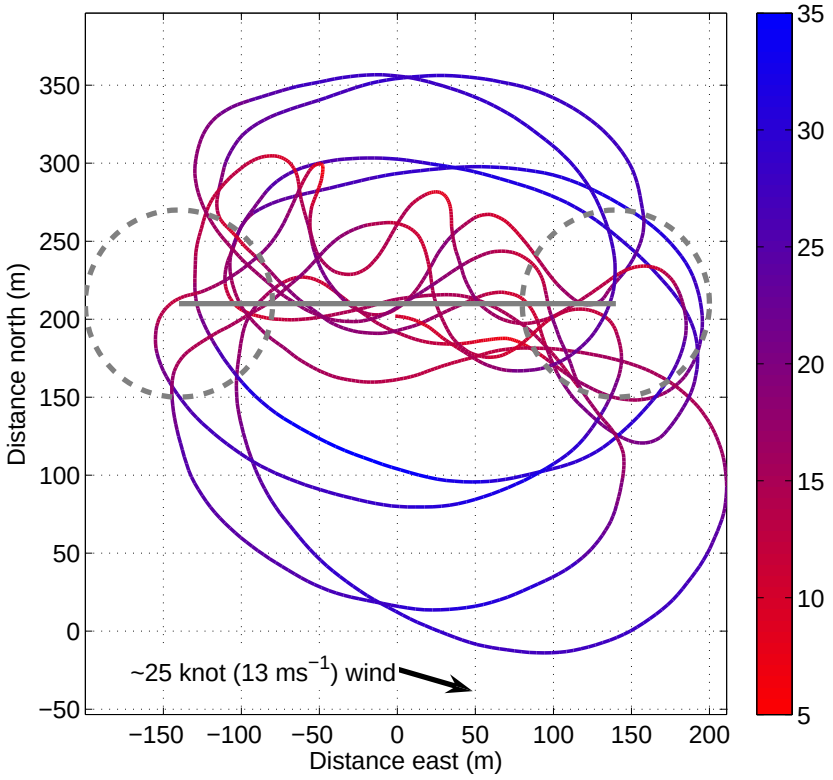


FIGURE 5.3: Measured track of the Trainer 60 with direct to waypoint tracking and coloured by ground speed (in ms^{-1})

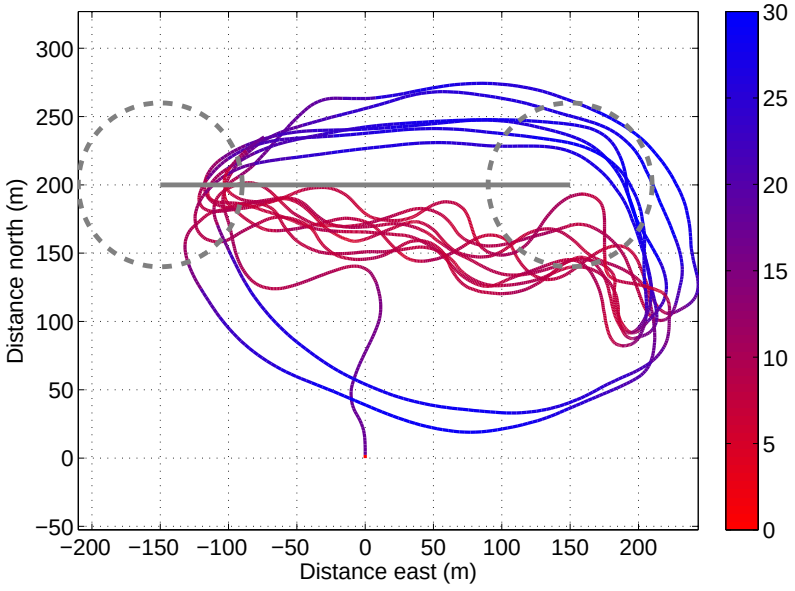


FIGURE 5.4: Simulated track of the Trainer 60 with direct to waypoint tracking and ∼15 ms^{-1} WNW wind

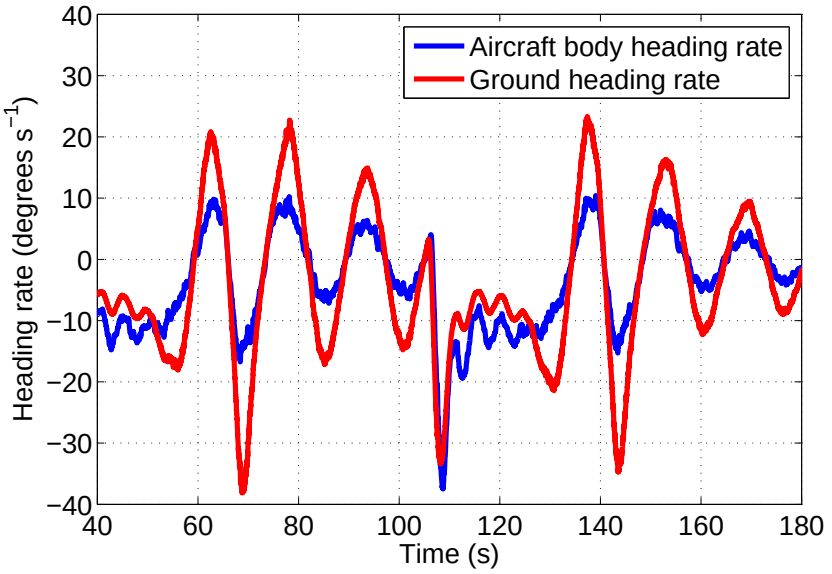


FIGURE 5.5: Comparison between simulated aircraft body and ground heading rates under ∼15 ms^{-1} wind

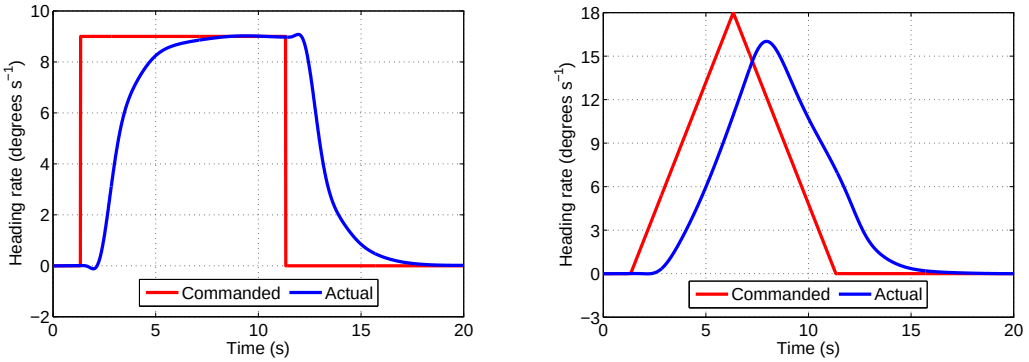
The current PRS implementation also relies on body-referenced gyroscopes for high-frequency heading prediction, leading to poor ground heading control in the presence of wind. A higher GPS update rate or prior knowledge of the wind vector would offer an improvement, though neither are possible with the current hardware implementation (the latter requiring a compass heading).

5.3 Acceleration Controlled Tracking

5.3.1 Introduction

The literature review (Section 2.7) identified Dubins paths [151–154] and the similar extremal control law [155] that require the aircraft to follow a path produced by line and circular segments. While such paths place a maximum limit on heading rate, they expect its value to instantaneously match a step command. Secondary algorithms have been developed to guide the aircraft on a realistic path round the extremal outline [31, 155]. The requirement of such algorithms increase computational requirements as a result of these impractical heading rate commands.

A novel algorithm is proposed that further differentiates heading to its second derivative, referred to as heading acceleration. Commanding heading acceleration



(a) Commanding heading rate

(b) Commanding heading acceleration

FIGURE 5.6: Comparison between commanding heading rate and acceleration

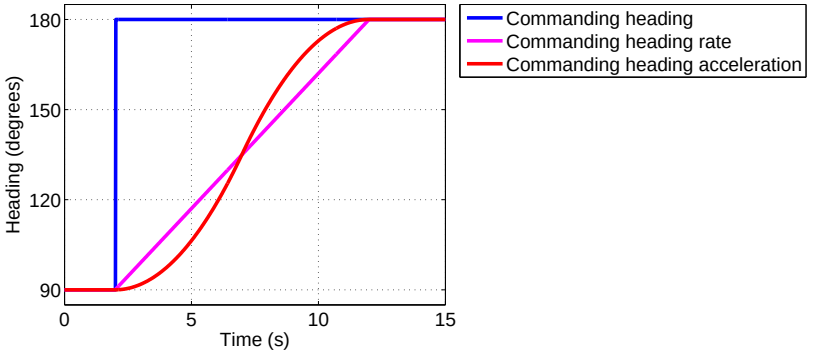


FIGURE 5.7: Expected heading using different commanded states

limits changes in the resulting commanded heading rate to realistic dynamics, allowing the aircraft to track commanded heading rate more reliably. An additional trajectory tracking algorithm should not be required.

Figure 5.6 shows simulated Trainer 60 heading rate following step changes in commanded heading rate and commanded heading acceleration. Both commanded a 90 degree turn over 10 seconds. The difference in normalised MAE for both results is not significant (though the heading acceleration result is 17% lower) but the gradient of the actual and commanded heading rate are closer matched in the latter result. This suggests the aircraft would perform better during the dynamic commanded heading rates expected during path tracking.

Figure 5.7 illustrates expected aircraft heading when commanding heading, heading rate or heading acceleration. The latter is considered most achievable due to its similarity to the Trainer 60 simulation results (Figure 3.17).

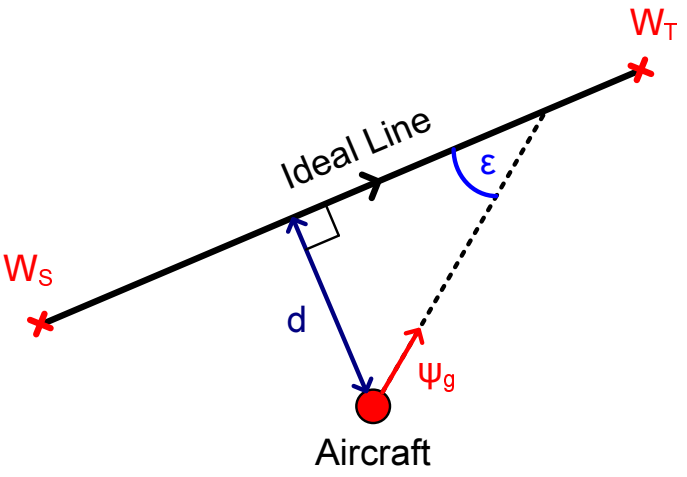


FIGURE 5.8: Tracking errors from the aircraft and ideal line between source and target waypoints

5.3.2 Algorithm Derivation

To align the aircraft to the ideal line connecting two waypoints, the proposed acceleration controlled tracking (ACT) algorithm must be able to command heading acceleration to achieve a predetermined distance and angle. Figure 5.8 illustrates the requirement to simultaneously turn the aircraft by angle ε while covering perpendicular distance d. The aircraft has an initial ground heading of ψ_g.

The angle ε_t turned by a constant heading acceleration k is found by integrating ψ twice with respect to time t, forming Equation 5.2 given initial heading rate $\dot{\psi}_0$.

$$\varepsilon_t = \frac{1}{2}kt^2 + \dot{\psi}_0 t \tag{5.2}$$

Figure 5.9 identifies the aircraft path required to align to the ideal line. Orthogonal components of the path, Δx and Δy, are illustrated and Δx shown to be equivalent to d. Δx travelled in time T is expressed by Equation 5.3 using the angle turned from Equation 5.2.

$$\Delta x = V_g \int_0^T \sin\left(\frac{1}{2}kt^2 + \dot{\psi}_0 t\right)\, dt \tag{5.3}$$

Since Δx is the same whether travelled forwards or backwards along the path, starting the calculation at the point of intercept with the ideal line and working backwards allows the initial heading rate to be set at zero and the integral

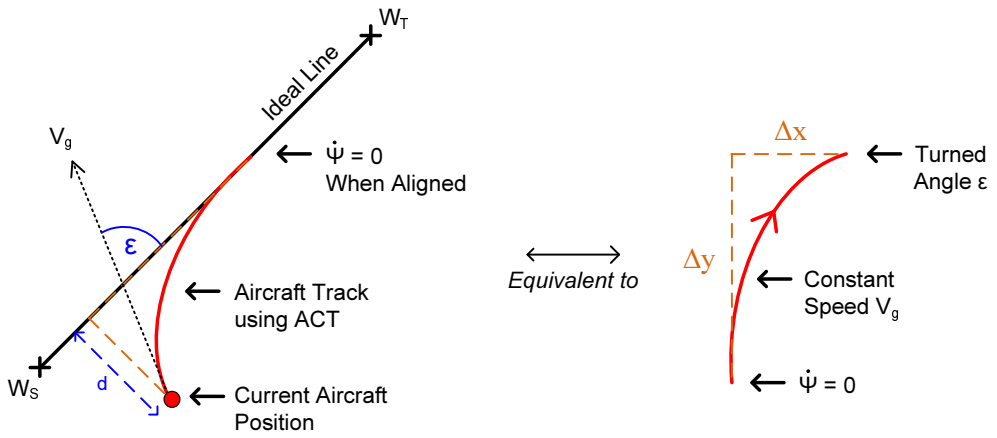


FIGURE 5.9: Identification of the desired aircraft path to align to the ideal line

expressed as Equation 5.4.

$$\Delta x = V_g \sqrt{\frac{\pi}{k}} \,\text{FresnelS}\left(\sqrt{\frac{k}{\pi}} T\right) \tag{5.4}$$

Equation 5.2 shows $T = \sqrt{\frac{2\varepsilon}{k}}$, producing Equation 5.5.

$$\Delta x = V_g \sqrt{\frac{\pi}{k}} \,\text{FresnelS}\left(\sqrt{\frac{2\varepsilon}{\pi}}\right) \tag{5.5}$$

Given $\Delta x = d$ and rearranging to Equation 5.6 shows the calculation for k to provide the required turn and perpendicular distance to align to the ideal line.

$$k = \frac{\pi \left(V_g \,\text{FresnelS}\left(\sqrt{\frac{2\varepsilon}{\pi}}\right)\right)^2}{d^2} \tag{5.6}$$

5.3.3 Practical Implementation

The Fresnel integral is defined by Equation 5.7.

$$\text{FresnelS}\,(x) = \int_0^x \sin\left(\frac{1}{2}\pi t^2\right) dt \tag{5.7}$$

Implementation in C is performed using a 9^{th} order Taylor series expansion. The expansion is combined with Equation 5.6 and simplified to produce Equation 5.8.

$$k = \frac{1}{71899960366080000} \left|\varepsilon^3\right| \times \left(\frac{V_g}{d}\left(126403200 - 9028800\varepsilon^2 + 287280\varepsilon^4 - 5016\varepsilon^6 + 55\varepsilon^8\right)\right)^2 \tag{5.8}$$

The implementation is accurate to less than $1.54 \times 10^{-5}\left(\frac{V_g}{d}\right)^2$ for $0 \leq |\varepsilon| \leq 120$ degrees and $0.1105\left(\frac{V_g}{d}\right)^2$ for $120 < |\varepsilon| \leq 180$ degrees when compared to Equation 5.6 using the Fresnel function provided by MATLAB. The accuracy for $|\varepsilon| > 90$ degrees is of less significance since the aircraft is already heading away from the ideal line and is simply required to turn towards it. Efficient coding permits Equation 5.8 to be calculated using 10 multiplies, 4 additions and 1 division. This is considered acceptably low computational requirements for real-time calculation by the autopilot.

Since ACT is designed to provide realistic heading rate commands, a saturation limit of $\pm k_{turn}$ is placed on k. Equation 5.6 suggests the k_{turn} limit is only likely to be reached when the aircraft is close to the ideal line but with a large heading error. Under such conditions the aircraft will overshoot the line, though should quickly converge providing the heading error continues to decrease.

Equation 5.6 also suggests k becomes very small as d increases, resulting in a slow convergence to the ideal path. The maximum acceptable d (d_{max}) is therefore determined by extending a line of length V_gL from the aircraft along its ground heading (Figure 5.10).

The positive constant L represents the distance required to turn a 90 degree corner at unit speed (i. e. 1 ms^{-1}) with a heading acceleration of k_{turn}, calculated by Equation 5.9.

$$L = \sqrt{\frac{\pi}{k_{turn}}}\left(\text{FresnelS}\left(\sqrt{0.5}\right) + \text{FresnelC}\left(\sqrt{0.5}\right)\right) \tag{5.9}$$

The V_gL line represents the minimum distance the aircraft requires to transition from being perpendicular to parallel to the ideal line. L only requires evaluation should k_{turn} change, and may be calculated and uploaded by the ground station along with user changes to k_{turn} to free autopilot resources.

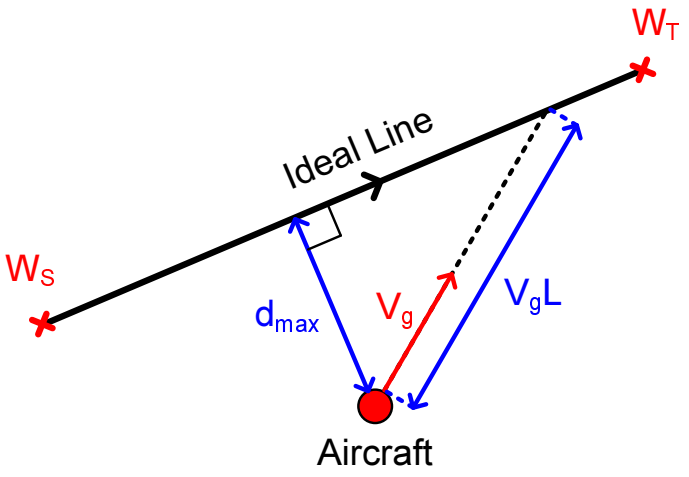


FIGURE 5.10: Calculation of d_{max}

Should the V_gL line not intercept the ideal line between source and target waypoints, an alternative algorithm is used that is not sensitive to d. This algorithm sets $k = k_{turn}$ and produces a new ε according to Equation 5.10.

$$\varepsilon = \begin{cases} \varepsilon - \sin^{-1}\left(\dfrac{d}{V_gL}\right) & |d| < V_gL \\ \varepsilon - \text{sign}\,(d)\,\dfrac{\pi}{2} & \text{otherwise} \end{cases} \tag{5.10}$$

The purpose of Equation 5.10 is to command a ground heading perpendicular to the ideal line when d is large, representing the shortest path to convergence. As d becomes smaller ($V_gL < d < d_{max}$) a gradual transition is made from a perpendicular to parallel alignment prior to the switch to Equation 5.6.

Since the autopilot directly controls heading rate (rather than heading acceleration) the commanded heading acceleration k is converted to ideal heading rate $\dot{\psi}_{ideal}$ with reference to ε as shown by Equation 5.11. The value of k and ε is assigned by either Equation 5.6 or Equation 5.10.

$$\dot{\psi}_{ideal} = \text{sign}\,(\varepsilon)\,\sqrt{2\varepsilon k} \tag{5.11}$$

Environmental disturbance, state estimation inaccuracies and controller behaviour will cause inevitable discrepancies between current and commanded heading rate. To maintain the limit on heading acceleration central to ACT, the previously commanded heading rate $\dot{\psi}_{c-}$ is compared to the new commanded rate $\dot{\psi}_{ideal}$. Should the difference exceed the maximum acceleration achievable by the aircraft (k_{max}) the commanded rate is saturated at this acceleration. k_{max} is always set greater than k_{turn}. For a given update time period Δt the commanded heading

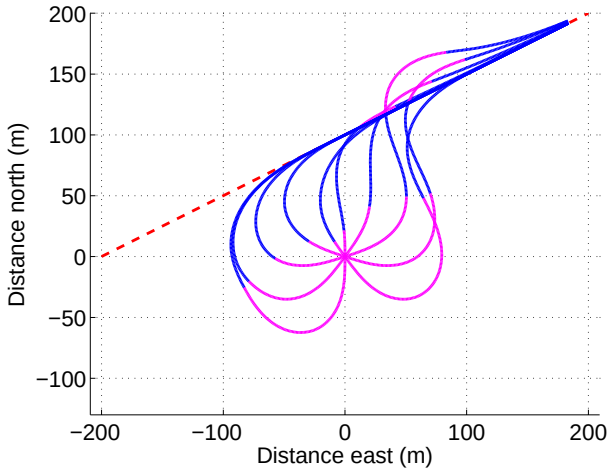


FIGURE 5.11: Simulation of aircraft precisely following multiple ACT paths under ideal conditions

rate $\dot{\psi}_c$ is shown by Equation 5.12.

$$\dot{\psi}_c = \begin{cases} \dot{\psi}_{ideal} & \left|\dot{\psi}_{ideal} - \dot{\psi}_{c-}\right| < k_{max}\Delta t \\ \dot{\psi}_{c-} + k_{max}\Delta t & \dot{\psi}_{ideal} > \dot{\psi}_{c-} \\ \dot{\psi}_{c-} - k_{max}\Delta t & \text{otherwise} \end{cases} \tag{5.12}$$

Should the tracking algorithm switch to the d-sensitive equation before the ground heading has reached ε, some residual heading rotation will be present. Since the V_gL line assumes an initial heading rate of zero, the aircraft may overshoot the ideal line due to the extra time required to counter this extra rotation. The "buffer" provided by the difference between k_{max} and k_{turn} helps to counter this and reduce the chance of overshoot.

5.3.4 Simulation Results

Ideal ACT tracks are presented in Figure 5.11. The aircraft is assumed to perfectly follow the commanded heading rate and no external disturbance is present. Ten paths are superimposed with the aircraft at different start headings. The ideal line is shown between source waypoint at (-200, 0) and target waypoint at (200, 200). Tracking while using the d-sensitive algorithm is highlighted in blue, while tracking using the ε-only algorithm is highlighted magenta. Some overshoot is seen in conditions where k exceeds k_{turn}, though the track quickly converges from the other side of the line.

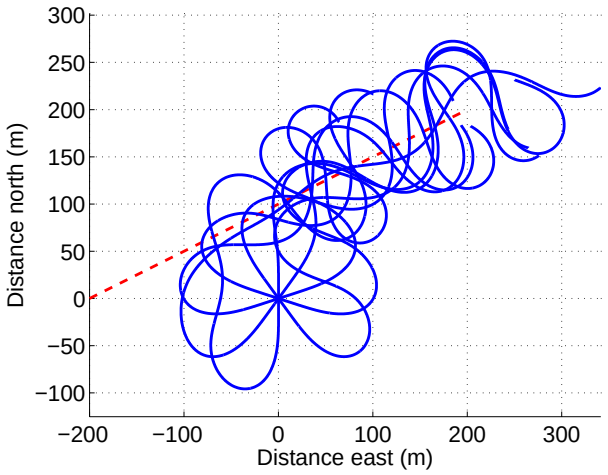


FIGURE 5.12: Simulated Trainer 60 following multiple ACT paths under ideal conditions

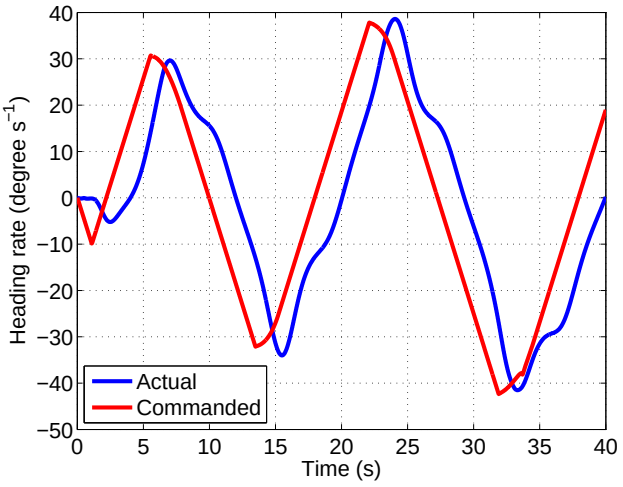


FIGURE 5.13: Comparison between actual and commanded heading rates for the simulated Trainer 60 following ACT

The same procedure was performed using the simulated Trainer 60. The resulting tracks are shown in Figure 5.12. An unacceptably high degree of heading oscillation is observed, leading to significant deviation from the expected tracks of Figure 5.11. The oscillation amplitude and frequency is observed to worsen with time up to a saturation limit, even when the aircraft is initially well aligned on the ideal line. Comparing the actual and commanded heading rates during ACT highlights a potential cause (Figure 5.13, taken from the ACT path with a 0 degree initial heading).

Though the range and gradient of the commanded heading rate is within the aircraft's ability (following the design specification for ACT) the actual heading rate

lags behind this command by approximately 2 seconds. Such a delay is expected from the controller response analysis (Section 3.3.4). Since the ACT algorithm does not compensate for this, the commanded heading acceleration continues to be accumulated on the commanded rate during this lag period, leading to the instability and oscillations observed.

5.3.5 Lag Compensation

A position and heading prediction stage was added before the main ACT algorithm in an attempt to remove this feedback lag, allowing changes in commanded heading rate to be reflected immediately in the ACT input. The assumption is made that the current heading rate will reach the commanded heading rate after a lag t_L seconds. Predicted ground heading $\hat{\psi}_g$ is then shown by Equation 5.13.

$$\hat{\psi}_g = \psi_g + \frac{1}{2}\left(\dot{\psi}_c + \dot{\psi}\right) t_L \tag{5.13}$$

To calculate the predicted position in a similar manner requires two Fresnel integrals. Since the autopilot must perform the prediction in real-time, a faster calculation is preferred. Instead, the aircraft is assumed to hold a constant heading rate ω, taken as the mean of current and commanded heading rates. It is also assumed ground speed V_g remains constant. The position offsets δx and δy are calculated by Equation 5.14.

$$\begin{aligned} \omega &= \frac{1}{2}\left(\dot{\psi}_c + \dot{\psi}\right) \\ \delta x &= \begin{cases} 0 & \omega = 0 \\ \dfrac{V_g}{\omega}\left(1 - \cos\left(\omega t_L\right)\right) & \text{otherwise} \end{cases} \\ \delta y &= \begin{cases} V_g t_L & \omega = 0 \\ \dfrac{V_g}{\omega}\sin\left(\omega t_L\right) & \text{otherwise} \end{cases} \end{aligned} \tag{5.14}$$

These offsets are rotated by the current ground heading and added to the current position to produce predicted positions $\hat{P}_x$ and $\hat{P}_y$ (Equation 5.15).

$$\begin{aligned} \hat{P}_x &= P_x + \delta x \cos\psi_g + \delta y \sin\psi_g \\ \hat{P}_y &= P_y + \delta y \cos\psi_g - \delta x \sin\psi_g \end{aligned} \tag{5.15}$$

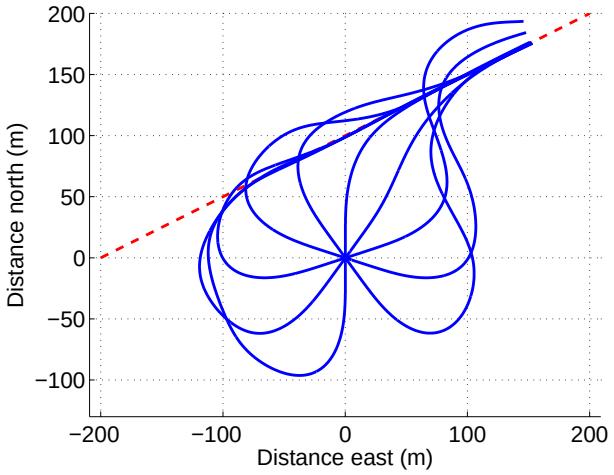


FIGURE 5.14: Simulated Trainer 60 following multiple ACT paths using predicted heading and position under ideal conditions

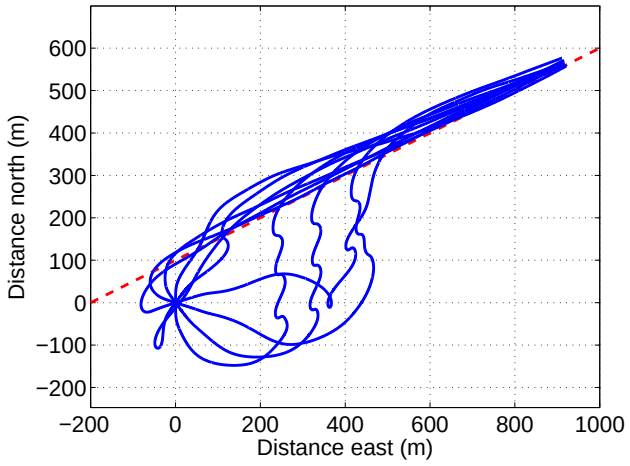


FIGURE 5.15: Simulated Trainer 60 following multiple ACT paths using predicted heading and position and estimated states with 15 ms^{-1} wind

The Trainer 60 ACT simulation was ran again using these predicted values and assumed lag of 2 seconds. The resulting tracks are shown in Figure 5.14. A considerable improvement over the unaided ACT (Figure 5.12) is observed, with no oscillation and the convergence of all tracks.

However, the algorithm remains sensitive to errors in heading and heading rate EKF state estimates and to changes in ground speed. Figure 5.15 shows the tracks of the simulated Trainer 60 with a 15 ms^{-1} westerly wind and using estimated heading and heading rate from the EKF. The target waypoint has been moved back along the ideal path to provide the extra space necessary to illustrate convergence. Though the aircraft does eventually converge to the line, evidence of loops and oscillation in the track suggests tracking instability.

5.3.6 Flight Test Results

ACT was attempted during practical flight tests with the real Trainer 60, though no successful convergence on the requested path was recorded. While the small flying space available made tracking conditions difficult, a tracking algorithm suitable for practical use is still expected to have adequately guided the aircraft. Though realistic paths are generated under ideal conditions, it is believed the ACT and prediction algorithms are too sensitive to measurement errors and environmental disturbance present in field operation. The prediction algorithm relies on states remaining constant that rarely are in practise. Estimated values for k_{max}, k_{track} are difficult to verify in practise, particularly as the path had not been successfully tracked. Measurement noise in heading rate also reduced confidence in t_L estimation.

As a result the aircraft is unlikely to follow the expected path generated by ACT, and simulated as well as practical results suggest the algorithm is insufficiently robust for reliable path tracking.

5.4 Closing Speed Tracking

5.4.1 Introduction

A new path tracking algorithm was sought to counter the deficiencies observed in the previous direct to waypoint and ACT methods. The new algorithm must efficiently align the aircraft along a line joining two waypoints, even in moderate winds. It must be simple enough for rapid computation by the autopilot's processor, and require only a small number of parameters. These parameters must be easily measured or estimated. The algorithm must be sufficiently robust to the level of measurement noise and external disturbance expected during real flight.

The inspiration for the following algorithm comes from identifying the technique a human pilot may use to manoeuvre the aircraft along a desired line. It is proposed this may be achieved by estimating the closing speed on the line, and adjusting the aircraft's trajectory to maintain a desired closing speed until convergence is achieved. The closing speed should decrease as the aircraft approaches the line.

To achieve this autonomously a closed-loop controller could maintain a commanded closing speed by adjusting bank angle. Previous experience and algorithm

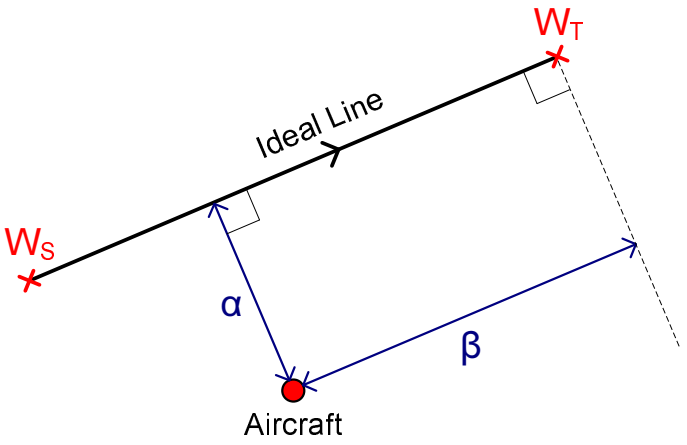


FIGURE 5.16: Cross and along–track distances for line tracking

simplicity suggest a PDF controller would be suitable. To maintain simplicity, the commanded closing speed will be directly proportional to the shortest distance to the ideal line. Knowledge of ground heading or heading rate is not directly required, so will be insensitive to discrepancies between true and estimated headings by the EKF.

Such a technique should be simple and robust, the latter achieved given the previously observed stable response of a well-tuned PDF controller to disturbances in external feedback. The ease of calculating closing speed to both linear and orbital paths would make this algorithm suitable for both.

5.4.2 Line Tracking

The path segment to be tracked is assumed to be a straight line between source W_S and target W_T waypoints. The proposed algorithm calculates the shortest distance α between the aircraft and this ideal line. α is always represented by a line running perpendicular to the ideal line, so is termed the cross-track distance. The distance β between the point on the ideal line closest to the aircraft and W_T is also calculated. This line runs parallel to the ideal line and is termed the along-track distance. Both distances are illustrated in Figure 5.16. α and β are calculated in two stages. The initialisation stage only requires calculation once

per waypoint pair, producing line constants l_{0-3} in Equation 5.16.

$$\begin{aligned} l_0 &= \frac{W_{Sy} - W_{Ty}}{\sqrt{(W_{Sx} - W_{Tx})^2 + (W_{Sy} - W_{Ty})^2}} \\ l_1 &= \frac{W_{Tx} - W_{Sx}}{\sqrt{(W_{Sx} - W_{Tx})^2 + (W_{Sy} - W_{Ty})^2}} \\ l_2 &= -l_0 W_{Sx} - l_1 W_{Sy} \\ l_3 &= l_1 W_{Tx} - l_0 W_{Ty} \end{aligned} \tag{5.16}$$

This permits a computationally efficient real-time update stage, with α and β calculated by Equation 5.17 given current aircraft position ($P_x,\ P_y$).

$$\begin{aligned} \alpha &= l_0 P_x + l_1 P_y + l_2 \\ \beta &= l_0 P_y - l_1 P_x + l_3 \end{aligned} \tag{5.17}$$

Cross-track and along-track speeds ($\dot{\alpha}$, $\dot{\beta}$) at time t are calculated from successive distance measurements given update period Δt (Equation 5.18).

$$\begin{aligned} \dot{\alpha}_t &= \frac{\alpha_t - \alpha_{t-1}}{\Delta t} \\ \dot{\beta}_t &= \frac{\beta_t - \beta_{t-1}}{\Delta t} \end{aligned} \tag{5.18}$$

5.4.2.1 Cross-track Speed

Cross-track speed is an important consideration for robust path tracking. This speed should be high when the aircraft is far from the track line to allow rapid convergence. Once the aircraft is exactly on track, this speed should be zero. The algorithm sets this speed as directly proportional to the cross-track distance, as shown in Equation 5.19 (where the subscript c denotes the commanded value).

$$\dot{\alpha}_c = \frac{\alpha}{\kappa} \qquad \kappa > 0 \tag{5.19}$$

While higher values of the constant κ will lead to a slower path convergence, the commanded path will generally be smoother and more achievable by the aircraft. κ should therefore be set as a compromise between expected aircraft performance and path convergence rate. Manual tuning during simulated aircraft runs is considered sufficient to determine a flight-ready value of κ.

The algorithm must next translate $\dot{\alpha}_c$ to a physical deflection of the aircraft's control surfaces to maintain $\dot{\alpha} = \dot{\alpha}_c$. Given it is generally preferable to maintain a steady airspeed, it is proposed this is achieved via the commanded bank angle ϕ_c. $\dot{\alpha}$ is reduced by turning away from the cross–track line and increased by turning towards it. The autopilot already employs a PDF controller to maintain $\phi = \phi_c$ by aileron deflection. An outer controller stage is added that commands bank angle to maintain $\dot{\alpha} = \dot{\alpha}_c$. The PDF algorithm is shown in Equation 5.20, where K_d and K_i represent the controller gains.

$$\phi_c = K_i \int (\dot{\alpha}_c - \dot{\alpha}) - K_d \dot{\alpha} \tag{5.20}$$

As with all the autopilot's controllers, optimal gain values are influenced by the aircraft's unique aerodynamics and require manual tuning by simulation and practical flight tests.

While the use of bank angle to maintain cross-track speed permits a simple control algorithm, it displays undesirable characteristics when the commanded cross-track speed is unattainably high. In this event the controller continues to increase the commanded bank angle to its maximum limit, resulting in only a tight circle being flown. The highest obtainable cross-track speed will be the ground speed while the aircraft's ground heading points normal to (and towards) the track line, at a heading ψ_n. This represents optimal convergence on the path. While an unobtainable $\dot{\alpha}_c$ may be prevented by limiting it to the current ground speed, any bank angle overshoot past ψ_n will still result in an unnecessary circle being turned. Such overshoot would be more frequent in high winds, particularly given the increased rate of change of ground speed reference. The circling occurs due to the inversion of the control law assumed in Equation 5.20 as the aircraft rotates past ψ_n; subsequent rotation *decreases* $\dot{\alpha}$.

The effect of limiting $\dot{\alpha}_c$ to fractions of ground speed is shown in Figure 5.17. For this simulation, an airspeed of 18 ms^{-1} was commanded and $\kappa = 11$. K_d and K_i have been manually tuned, and the ideal line is shown dashed. When $\dot{\alpha}_c$ is limited to ground speed a single circle can be seen due to bank angle overshoot. Repeated circling is observed when $\dot{\alpha}_c$ is limited to (unobtainable) values higher than ground speed. While $\dot{\alpha}_c$ can be limited to a fraction $\ll 1$ of the current ground speed to make rotating past ψ_n unlikely, Figure 5.17 shows the convergence on the ideal line to be considerably slower.

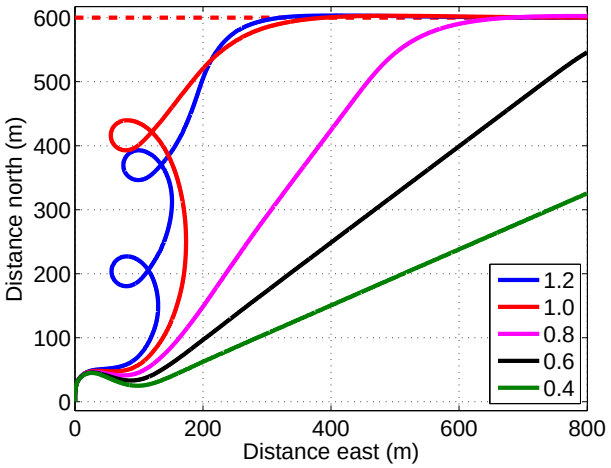


FIGURE 5.17: Simulated Trainer 60 tracks with $\dot{\alpha}_c$ limited to fractions of ground speed

5.4.2.2 Along-track Speed

The use of along-track distance β offers a more robust solution, while maintaining the simplicity of the algorithm. Holding β at a constant value will always command a perpendicular (optimum) approach to the ideal line. Thus setting $\dot{\beta}_c = 0$ and substituting α for β in Equation 5.20 offers a minimum convergence time without risk of tripping the circular track behaviour. β control is switched to α control when $\dot{\alpha}_c$ (calculated regardless of control mode) becomes less than the ground speed (V_g). Since α will already be decreasing by this point (given the aircraft is travelling towards the path) it is unlikely $\dot{\alpha}_c$ will exceed ground speed again in a sufficiently short time to require hysteresis on this switch condition. The combined tracking algorithm is shown in Equation 5.21, with the introduction of a generic closing speed value χ.

$$\begin{aligned} \chi &= \begin{cases} \dot{\alpha} & \dot{\alpha}_c \leq V_g \\ \dot{\beta} & \text{otherwise} \end{cases} \\ \chi_c &= \begin{cases} \dot{\alpha}_c & \dot{\alpha}_c \leq V_g \\ 0 & \text{otherwise} \end{cases} \\ \phi_c &= K_i \int (\chi_c - \chi) - K_d \chi \end{aligned} \tag{5.21}$$

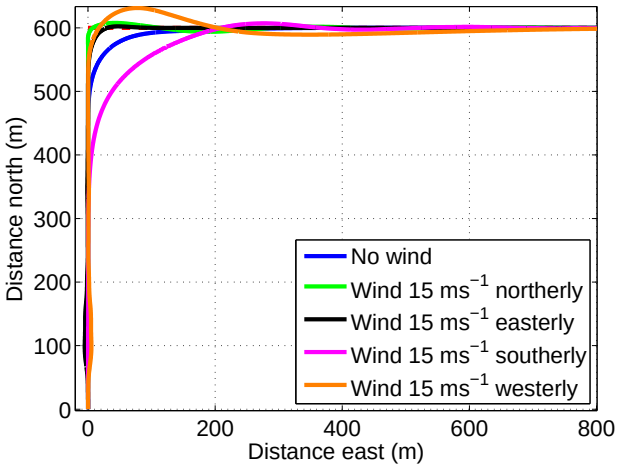


FIGURE 5.18: Simulated Trainer 60 tracks using closing speed tracking ($\kappa = 11$) in variable wind conditions

Simulations showed smoother tracking performance if the controller integral is reset (Equation 5.22) following a switch of control types, preventing integral accumulation from the previous control attempt affecting the new manoeuvre.

$$\int (\chi_c - \chi) = \frac{\phi + K_d \chi}{K_i} \tag{5.22}$$

Figure 5.18 shows tracks of the simulated Trainer 60 using closing speed tracking. The simulation settings are as before, only includes 15 ms^{-1} (34 mph moderate gale force) winds from various directions. The aircraft is seen to hold a steady perpendicular approach and a sufficiently rapid convergence on the ideal line. The distance from the path segment before the aircraft turns onto α control is seen to vary with ground speed. The aircraft begins the turn earlier under high ground speed conditions (southerly wind) than low ground speed (northerly wind), reducing the chance of overshooting the line. Some overshoot (<32 m) is observed in the westerly wind condition due to the increase in ground speed as the aircraft turns onto the line.

The identification of a stable line-tracking method permits the effect of varying κ to be better explored, shown in Figure 5.19. No wind is present, to clarify the results. Low values to κ can be seen to result in line overshoot and oscillation, due to the PDF control algorithm's (Equation 5.21) delay in rotating the aircraft to maintain the increased rate of change of α_c. While this may be due in part to non-optimal choice of controller gains (K_d, K_i), the aircraft performance itself limits the minimum achievable value of κ. Higher values of κ result in an increasingly

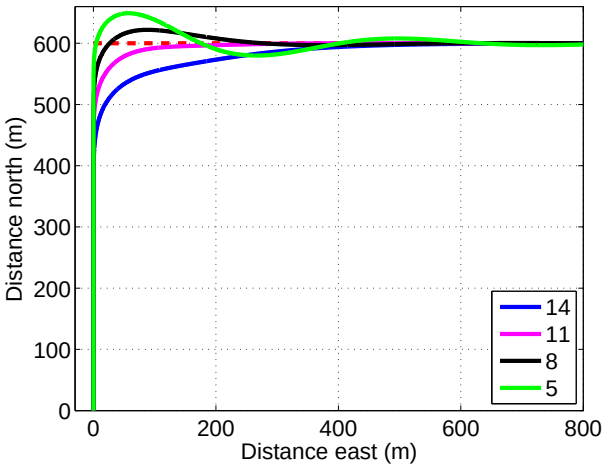


FIGURE 5.19: Simulated Trainer 60 tracks as κ is varied from 14 to 5

slow convergence to the ideal line. The setting of $\kappa = 11$ appears to be a suitable compromise for the Trainer 60.

5.4.3 Orbit Tracking

Closing speed tracking can also be used for orbiting a waypoint W_T with radius W_{Tr}. Cross-track distance α is calculated as the distance between the aircraft and orbit radius (Equation 5.23). The sign of α is switched to set clockwise or anti-clockwise orbiting.

$$\alpha = \sqrt{(P_x - W_{Tx})^2 + (P_y - W_{Ty})^2} - W_{Tr} \tag{5.23}$$

Along-track distance β is calculated by evaluating a target line from the previous aircraft position $\left(P_{x(t-1)},\ P_{y(t-1)}\right)$ to a point on the waypoint radius. Ideally this would be the tangent point where a line from $\left(P_{x(t-1)},\ P_{y(t-1)}\right)$ makes a single intercept with the orbit. To save computational resources however, a suitable approximation is made to a point ± 90 degrees on either side of the bearing of W_T from the previous aircraft position. The point I chosen is dependant on the desired clockwise or anti-clockwise orbiting, as shown in Figure 5.20 and Equation 5.24.

$$\begin{aligned} \theta &= \begin{cases} \texttt{atan2}\left(P_{x(t-1)} - W_{Tx},\ P_{y(t-1)} - W_{Ty}\right) + \dfrac{\pi}{2} & \text{clockwise} \\ \texttt{atan2}\left(P_{x(t-1)} - W_{Tx},\ P_{y(t-1)} - W_{Ty}\right) - \dfrac{\pi}{2} & \text{anti-clockwise} \end{cases} \\ I_x &= W_{Tx} + W_{Tr}\sin\theta \\ I_y &= W_{Ty} + W_{Tr}\cos\theta \end{aligned} \tag{5.24}$$

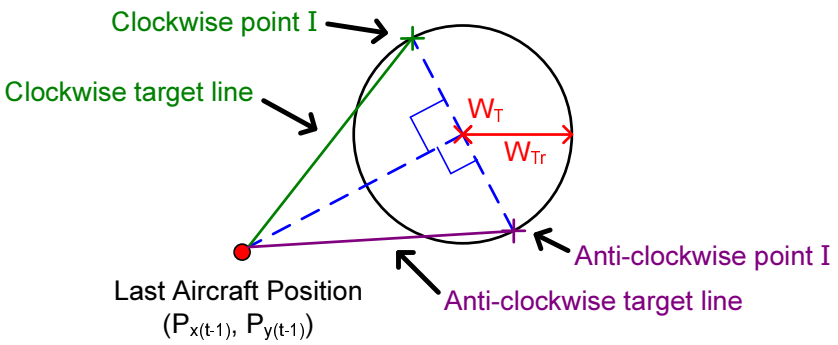


FIGURE 5.20: Target lines for clockwise and anti-clockwise orbiting

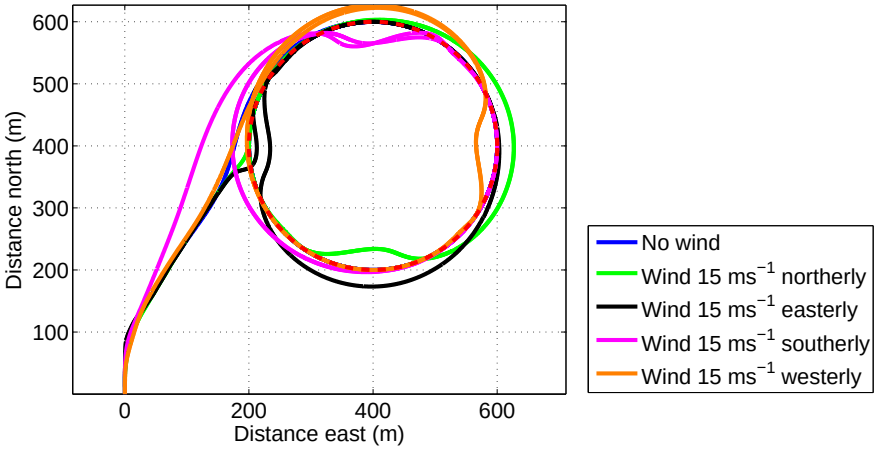


FIGURE 5.21: Simulated Trainer 60 tracks using closing speed tracking ($\kappa = 11$) in variable wind conditions

The *cross-distance* between the target line and current aircraft position is used to calculate β. Since I changes with aircraft position, pre-computation of the line coefficients is not possible for orbit tracking. l_{0-2} are calculated as for Equation 5.16 substituting W_S for P_{t-1} and W_T for I. β is then calculated as for α in Equation 5.17. When the aircraft is inside the waypoint orbit ($\alpha < 0$), I is set to W_T and the sign of β is inverted. This produces a target line radiating out from the centre of the waypoint and the aircraft is directed outwards along it. The remaining tracking equations (Equations 5.18, 5.19 and 5.21) are identical to line-tracking.

Figure 5.21 shows tracks of the simulated Trainer 60 under different wind conditions. The path being tracked is an orbit with 200 m radius centred 400 m north and 400 m east of the origin. As before, an airspeed of 18 ms^{-1} is commanded. The orbit track is seen to vary by <35 m from the zero wind condition to 15 ms^{-1} wind speeds and again remains stable throughout each simulation. The orbital deviations observed in the presence of high wind are due to rapid changes in ground

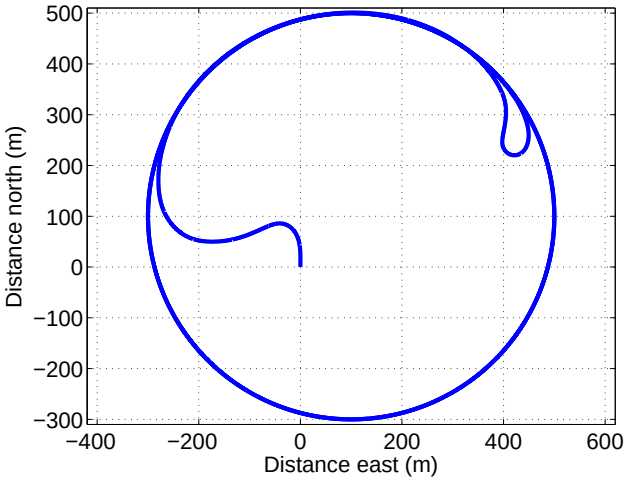


FIGURE 5.22: Simulated Trainer 60 track (orbit, $\kappa = 11$) demonstrating switch in orbit direction

speed as the aircraft turns. The delay before the closing speed PDF controller can compensate leads to this deviation.

Figure 5.22 demonstrates both an orbit from an internal start-point and a switch from clockwise to anti-clockwise commanded orbit direction. In both conditions the tracking algorithm is shown to provide a smooth transition to the required orbit. The commanded orbit centre is 100 m north and 100 m east of the origin, with a 400 m radius wide enough to demonstrate β tracking inside the orbit.

5.4.4 Complete Path Tracking

A path consisting of linear segments and orbit radii is expected to fulfil all expected mission requirements. The transition from one linear section to the next may be triggered once $\beta < 0$. Since β is already available, this solution is computationally efficient as well as permitting a direct fly-over of the target waypoint prior to switching. Transition always occurs even if the aircraft is far off track, and outperforms triggering on radial distance from the target waypoint. A small trigger radius allows the aircraft to fly close to the waypoint, though risks the aircraft missing it and initiating a costly turn-around which potentially leads to an infinite orbit pattern. A larger trigger radius is more robust to track deviations, though always triggers further from the target waypoint which may not be desirable for some mission objectives.

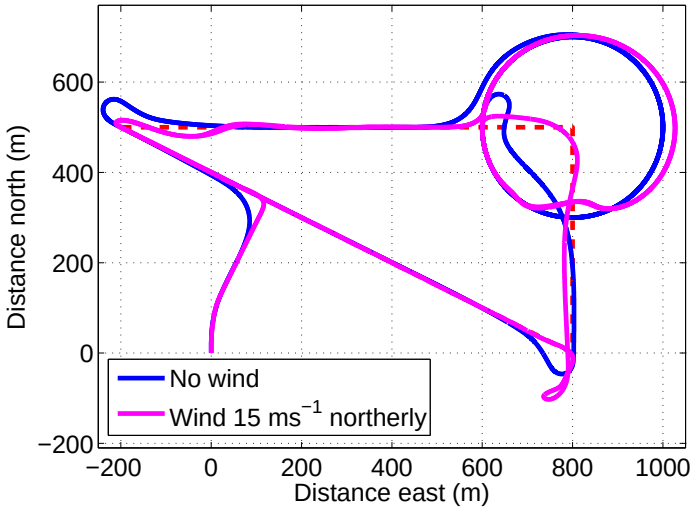


FIGURE 5.23: Simulated Trainer 60 track ($\kappa = 11$) of a complete mission path with zero and 15 ms^{-1} northerly wind

The switch condition for orbit-tracking can be either orbit duration (particularly time spent in α control) or the number of orbits performed. In the latter the bearing of W_T from the aircraft following the initial switch to α control is a suitable start reference. Thereafter, the crossing of successive quadrant boundaries ($\frac{\pi}{2}$ radians apart) is detected by observation of the bearing to W_T, and a complete orbit assumed once all four have been passed in the correct order. This technique will be resilient to premature triggering of a complete orbit should a gust of wind (for example) temporarily blow the aircraft back behind the start bearing.

The switch from a line to an orbit segment should be made once the orbit's α_c is less than or equal to ground speed. This requires α and α_c to be calculated for both line and orbit tracking when following a line where W_T has an orbit radius. The higher computational load of orbit β calculation is not required. The first time the orbit condition is triggered, the line segment is considered terminated.

Figure 5.23 shows two tracks of a complete mission path, under zero and 15 ms^{-1} northerly wind conditions. As for previous simulations the aircraft starts at the origin and is instructed to hold a constant airspeed of 18 ms^{-1}. No manual intervention was made during the simulated flight, with all waypoint switches performed automatically. The three mission waypoints form a triangle with the waypoint at the north-east corner having an associated orbit. The orbit had a 200 m radius and the aircraft was requested to perform three complete clockwise orbits before continuing. The complete path is followed as expected under both wind conditions. The aircraft actually tracks closer to the path at the first and westerly turn

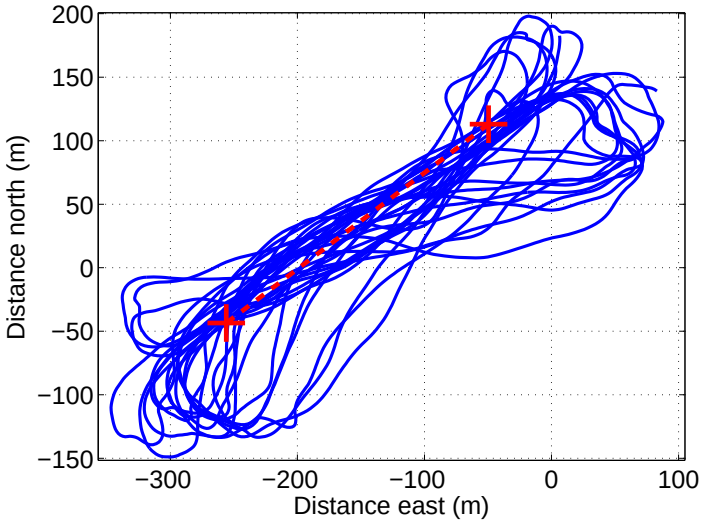


FIGURE 5.24: Trainer 60 tracks during flight test 19 using closing speed tracking to a line

points due to the northerly wind reducing its closing speed on the ideal lines and permitting a closer turn. The north wind blows the aircraft further south on the southerly waypoint crossing, leading to a switch to β control prior to successful recovery.

5.4.5 Flight Test Results

Figure 5.24 shows the successful tracking of a line during flight test 19 using closing speed tracking. The data collected represents over 14 minutes of uninterrupted autonomous path tracking. The measured tracks closely represent the simulated tracks of the Trainer 60 when given the same waypoints (Figure 5.25). Estimated states and zero wind have been used.

Similar successful results are observed during the flight test for orbit tracking (Figure 5.26). Again, simulated results under the same conditions confirm the practical tracks.

The waypoint positions used for the flight test 19 line tracking were selected using a hand-held GPS unit held at the position required. High visibility objects (a life ring over a high visibility jacket) were placed at both waypoint positions prior to take-off. A digital camera was fixed under the Trainer 60's fuselage with an electronic trigger attached to the autopilot. The autopilot was instructed to trigger the camera when switching waypoints. Rather than attaching a stand-alone

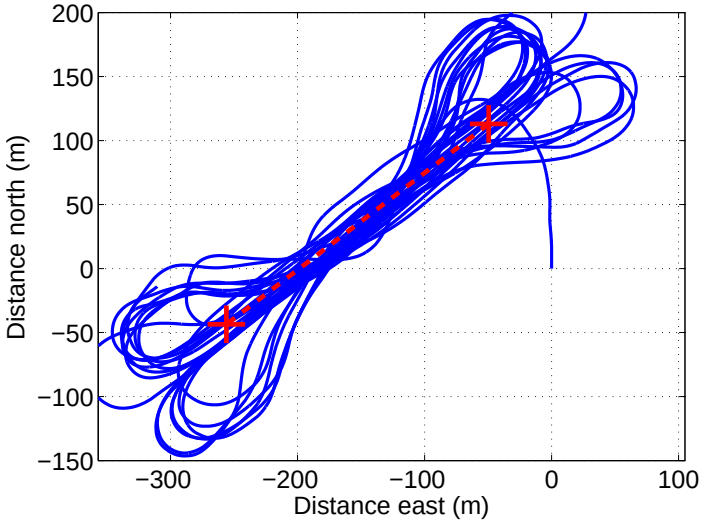


FIGURE 5.25: Simulated Trainer 60 tracking using closing speed tracking to a line and flight test 19 waypoints

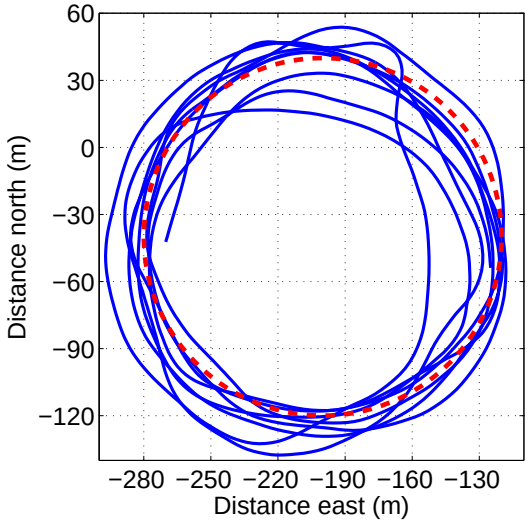


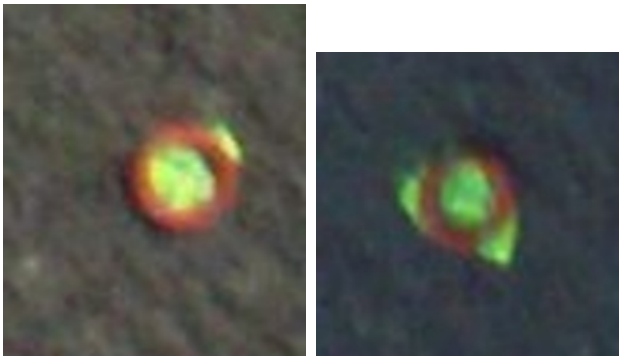
FIGURE 5.26: Trainer 60 tracks during flight test 19 using closing speed tracking to an orbit

video camera, the interaction between autopilot hardware, payload and current mission phase is considered a necessary step towards useful scientific missions and is a research objective. Images of both the easterly and westerly waypoints were taken during autonomous flight. Figure 5.27 shows the complete images, with the waypoint markers highlighted by green squares in the bottom-right. Zoomed images are shown in Figure 5.28.

The full images show the camera was not pointing directly down at the target markers when the picture was taken. While some position error may be attributed

(a) East waypoint (b) West waypoint

FIGURE 5.27: On-board digital camera images taken during line tracking for flight test 19



(a) East waypoint (b) West waypoint

FIGURE 5.28: Zoomed images of waypoints taken by the on-board camera during line tracking for flight test 19

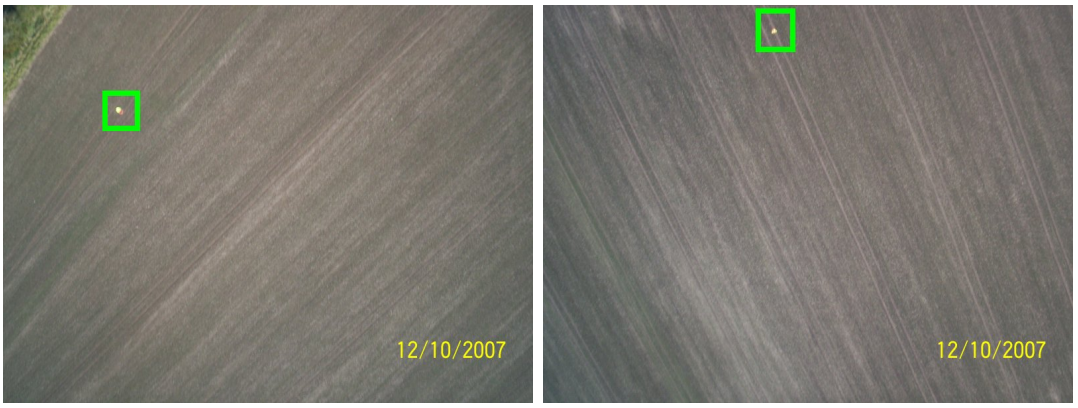
to PRS inaccuracies, the ∼2 second delay between the autopilot signalling the digital camera and the camera actually taking the picture is the most likely cause of the offset observed. By this time the aircraft has already began to turn towards the new waypoint, further offsetting the image since the camera is not gimballed.

The proposed solution for line tracking estimates the time t_w till switching to a new waypoint using along-track distance and speed (β and $\dot{\beta}$) as shown by Equation 5.25.

$$t_w = -\frac{\beta}{\dot{\beta}} \tag{5.25}$$

Each waypoint has an optional trigger time t_a causing the autopilot to signal the payload at the first occasion that $t_w \leq t_a$.

Flight test 20 flew a similar path using closing speed tracking. Both waypoints had a 3 second trigger time (t_a). Complete images are shown in Figure 5.29. All



(a) North waypoint (b) South waypoint

FIGURE 5.29: On-board digital camera images taken during line tracking for flight test 20

images taken show the waypoint markers in the top half of the frame, and are generally closer to the centre line than those from the previous flight test. Both these observations suggest the camera trigger delay compensation was successful, though a reduction of t_a to 2 seconds may further improve the marker position.

5.4.6 Conclusion

A path tracking algorithm designed for robust performance during oceanographic missions has been proposed. The combined control of cross-track and along-track closing speeds successfully navigated a simulated UAV along straight line and circular tracks in the presence of strong winds of various directions. Techniques for determining the reliable termination of a linear and circular path component have been demonstrated. The algorithm remains sufficiently simple for high-frequency updates on the relatively low-power processor envisaged for the UAV. Simulation results have been verified by practical flight demonstrations. Tracking parameters have estimated the arrival time at the target waypoint and been used to compensate for a digital camera's trigger delay to successfully image waypoint markers on the ground.

Chapter 6

Conclusions

A complete UAV system has been developed and flown on 22 flight tests. This research has focused on the UAV autopilot and ground station software, and met the following research objectives:

- Demonstrate a robust autonomous flight control system.
- Provide a reliable and accurate path tracking ability.
- Investigate real-time autonomous tuning of controllers.
- Develop reliable ground station software tailored to the oceanographic UAV and expected operator experience.
- Demonstrate autopilot interaction with payload.

Major novel contributions of this research were:

- Pseudo-derivative feedback controllers used for UAV flight control.
- Customisation of extended Kalman and complementary filters for aircraft state estimation using the available sensors and processor.
- Use of a finite impulse response filter to reduce aliasing of autopilot states transmitted to the ground station.
- Real-time autonomous tuning of controllers.
- The development, characterisation and optimisation of two novel path tracking algorithms.

The literature review in Chapter 2 identified potential oceanographic UAV applications and associated scientific instrumentation required to be carried. Though the need for such a system was confirmed, no suitable commercial UAV was found. Similarly no commercial autopilot was found, and the subject of this research focused on the provision of an in-house product. COTS model aircraft were used for autopilot testing due to their typically low cost, readily available spare parts and ease of construction. The Trainer 60 model was selected because of its adequate flight duration, stability and payload capacity. The construction of a custom airframe designed for oceanographic missions was handled by a separate research group.

No suitable method of direct attitude estimation was present in the literature. Instead, Kalman and complementary filters were identified as common sensor fusion algorithms used in UAV's for attitude estimation from inertial measurements. The Kalman filter has the advantage of being able to estimate states not directly measured, such as gyroscope bias, though required greater processor resources. Both PID and PDF controllers are commonly used control algorithms, with PID being a popular choice for UAV's but PDF having no previous UAV application. PDF control was chosen for this research due to the literature's suggestion that PDF control offered performance gains over PID in similar applications.

Chapter 3 presented the system design based upon the hardware and algorithms selected during the literature review. A custom daughter board was developed to extend the functionality of the commercial sensors and processor board purchased for the autopilot. A simulation environment was created that modelled the aerodynamics of the test aircraft under realistic conditions. This aided development of the flight control algorithms by providing a repeatable and predictable environment with a wide range of data logging and visualisation options. The unlimited duration and risk-free nature of the simulation also allowed a greater degree of robustness testing of the ground station operator interface and the manual tuning of the controller gains. The use of software-in-the-loop allowed the C code run by the autopilot hardware to be similarly tested without modification.

A robust binary communications protocol was developed with minimal header size suitable for use with a wireless modem. The low data rate excluded real-time transmission of all desired autopilot states during flight. Instead, a subset of these states was selected for transmission depending on the category of data required. Since the autopilot updated most states at a higher rate than the modem

bandwidth permitted, the data was passed through a FIR decimation filter prior to transmission to reduce aliasing.

An extended Kalman filter was used to provide estimates of the autopilot's Earth-referenced attitude and heading. The Joseph stabilised form of the error covariance update was found to improve numeric stability. A custom application was written to generate optimised matrix calculations allowing the autopilot to iterate the filter at a sufficiently high rate on the low power processor. The mean absolute error of the estimated attitude was less than half a degree when compared with the simulation's true states. The different frame of reference between the GPS and gyroscopes was considered the primary reason for the estimated heading's higher mean absolute error of 5 degrees, though was still considered acceptable. The use of a magnetic compass for heading measurement was recommended for future hardware revisions. Simulations showed the complementary filter provided acceptable position estimates by combining low-frequency GPS updates with high-frequency dead-reckoning.

PDF controllers were found to adequately control the attitude, airspeed, altitude and heading rate of the simulated aircraft. Little overshoot was observed following a step change in commanded value, though the airspeed and altitude controllers were tuned for a slow response to reduce the effect of Phugoid oscillation on the aircraft.

Practical flight tests demonstrated a similar level of state estimation and control performance to that suggested by simulation. Airspeed control was identified as a particular area where the controller gains required further tuning due to discrepancies between simulated and real aircraft behaviour. The successful in-field operation of the UAV system was confirmed with a 70 second fully-autonomous flight, meeting the flight control and ground station research objectives.

The PDF controller gain values had been selected using a manual trial and error technique with feedback from the simulated aircraft. Though practical flight tests at the end of Chapter 3 showed these values provided a stable flight performance, they were not considered optimal. Chapter 4 describes a real-time method of autonomous controller tuning. An ideal controller response was modelled as a linear section transitioning to a normal cumulative distribution function. The inverse mean squared-error between ideal and measured performance provided a quantitative estimate of controller optimality.

Optimality maps produced from simulated flights highlighted the effect of PDF controller gain values on response optimality. Observation of these maps showed a single region of high optimality existed for all controllers tested. A maximisation function was found to locate the gain pair near the region of highest optimality within a practical number of iterations. The algorithm successfully tuned a simulated bank angle controller with both standard and extra response delay. The practical application of the algorithm was demonstrated by a bank angle controller optimisation run during a flight test, though further flight tests would be required to validate the optimal gain values.

Chapter 5 extended the autonomous flight control system developed by the previous chapters to include a high-level path tracking ability. Simply pointing the aircraft's ground heading at successive waypoints was shown to lead to an inefficient and unstable track when the distance between waypoints was small or wind was present. Heading was proved to be one of the least reliable state estimates in Chapter 3. Two novel path tracking algorithms were presented that do not directly rely on heading and aimed to guide the aircraft more accurately along the mission path.

Acceleration controlled tracking directed the aircraft by commanded heading rate. Acceleration refers to the derivative of heading rate and was set by the algorithm to ensure changes in commanded rate were kept within achievable limits. Simulations under ideal conditions showed the algorithm could dynamically generate realistic aircraft trajectories with a smooth convergence on the path. Following realistic simulation and practical flight tests the algorithm proved insufficiently robust due to its high sensitivity to response delay and external disturbances.

The second novel algorithm adjusted bank angle to control the aircraft's closing speed on the ideal path. When the aircraft was close to the ideal path, closing speed refers to the rate of change of the distance between the aircraft and closest point on the path. When far from the ideal path, closing speed refers to the rate of change of the distance travelled parallel to the path. Simulations suggested the method provided robust tracking under high winds for both linear and circular path sections. Successful practical flight tests confirmed the algorithm's suitability and was deemed to meet the path tracking research objective. The final research objective of autopilot interaction with a payload was met by the successful triggering of a digital camera over predetermined waypoints. Estimation of the waypoint arrival time allowed the autopilot to compensate for the delay between camera trigger and image acquisition.

This research has provided an autopilot and ground station considered applicable for future oceanographic application by the National Oceanography Centre, Southampton. This thesis will be used as a resource for future projects at the Centre as it is the first to bring together all the components of a complete UAV system. The system has been demonstrated using a commercial model aircraft as a test platform. Future work will apply this technology to the full size custom airframe. The autopilot controllers will be tuned to this custom airframe using the aerodynamic model generation and simulation software presented in Chapter 3. Once complete, the autonomous flight testing of a low-cost, mission-ready oceanographic UAV is anticipated.

Appendix A

AVL Trainer 60 Model

The aerodynamic coefficients for the Trainer 60 test aircraft generated by AVL are summarised in Table A.1. Since AVL coefficients for control surface deflections are specified in degrees, the listed values have been multiplied by $\frac{180}{\pi}$ to convert to the radian reference required by the Aerosim model definition. In addition, all coefficients relating to flaps were zero (since they are not present on the Trainer 60), compressibility (Mach-related coefficients) is zero (due to slow speed) and all alpha rate coefficients assumed to be zero for unsteady flow. For $CLmind$, X-Foil analysis of the Trainer 60 aerofoil suggested minimum drag was achieved at zero alpha, enabling $CLmind = CL0$.

The aircraft moment of inertia (with a full fuel tank) calculated by AVL are shown in Table A.2.

The complete listing of the AVL aircraft input files for the Trainer 60 aircraft are provided in listings A.1 and A.2. The first listing is for the .avl file, the second for the .mass file.

TABLE A.1: Aerodynamic coefficients generated by AVL for the Trainer 60 aircraft

Coefficient	Description	Value
$CL0$	Lift at zero alpha	0.50261
CLa	Lift derivative w.r.t alpha	4.810457
$CLdf$	Lift due to flap deflection	0
$CLde$	Lift due to elevator deflection	0.500250
$CLalphadot$	Lift due to alpha rate	0
CLq	Lift due to pitch rate	8.436431
CLM	Lift due to compressibility	0
$CLmind$	Lift at minimum drag	0.50261
$CDmin$	Minimum drag	0.03467
$CDdf$	Drag due to flap deflection	0
$CDde$	Drag due to elevator deflection	0.027846
$CDda$	Drag due to aileron deflection	0
$CDdr$	Drag due to rudder deflection	0
CDM	Drag due to compressibility	0
osw	Oswald's coefficient (e in AVL)	0.9940
$CYbeta$	Side force due to beta	-0.167419
$CYda$	Side force due to aileron deflection	-0.076662
$CYdr$	Side force due to rudder deflection	0.107830
CYp	Side force due to roll rate	0.003789
CYr	Side force due to yaw rate	0.158863
$Cm0$	Pitching moment at zero alpha	-0.12382
Cma	Pitching moment derivative w.r.t alpha	-1.036001
$Cmdf$	Pitching moment due to flap deflection	0
$Cmde$	Pitching moment due to elevator deflection	-1.2117
$Cmalphadot$	Pitching moment due to alpha rate	0
Cmq	Pitching moment due to pitch rate	-10.711699
CmM	Pitching moment due to compressibility	0
$Clbeta$	Rolling moment due to sideslip	-0.100305
$Clda$	Rolling moment due to aileron deflection	-0.32618
$Cldr$	Rolling moment due to rudder deflection	0.0037242
Clp	Rolling moment due to roll rate	-0.442534
Clr	Rolling moment due to yaw rate	0.138892
$Cnbeta$	Yawing moment due to sideslip	0.055546
$Cnda$	Yawing moment due to aileron deflection	0.021429
$Cndr$	Yawing moment due to rudder deflection	-0.049504
Cnp	Yawing moment due to roll rate	-0.029518
Cnr	Yawing moment due to yaw rate	-0.063150

TABLE A.2: Moments of inertia calculated by AVL for the Trainer 60 aircraft

Moment of inertia	Value ($kg\,m^2$)
I_{xx}	0.3330
I_{yy}	0.4920
I_{zz}	0.7772
I_{xz}	0.0052

LISTING A.1: AVL input for Trainer 60 .avl file

```
#****************************************
# Trainer 60 Test Aircraft
# Written by Matthew Bennett, October 2006
#****************************************

# Units are metric (see mass file)
# Geometric origin is the centre of the propeller shaft level with where the
    nose cone touches the motor
# This is 8cm above the base of the fuselage

# Name
T60-1
# Mach
# Travelling about 17 m/s
0.05
# IYsym IZsym Zsym
# Assume no aircraft Y or Z symmetry
0 0 0.0
# Sref (surface area of wing, m^2) Cref (average chord, [wing surface
    area]/[wing span], m) Bref (wing span, m)
0.6164 0.335 1.84
# Xref Yref Zref (centre of gravity)
0.387 0.005 0.0155 ! full fuel tank
# 0.397 0.005 0.0165 ! empty fuel tank
# CDoref
# Default profile drag coefficient added to geometry. 0.020 is a common default.
0.02

SURFACE
Wing
# Nchord Cspace Nspan Sspace
# Use 10 chord vortexes with cosine spacing, and 35 span vortexes with -sine
    spacing
# Adjust numbers to ensure smooth spacing with no sudden breaks
# Spacing types used concentrate vortexes at areas where geometry more rapidly
    changes
10 1.0 35 -2.0
# Reflect wing about Y=0 plane (i.e. XZ plane)
YDUPLICATE
0.0
# Set angle of incidence of wing (saves having to change all defining surface
    Ainc values)
# This is the angle between the wing chord line and longitude of the fuselage
```

```
ANGLE
2.0
SECTION
# Wing section from centre of fuselage to start of aileron
# Xle Yle Zle (location of aerofoil's leading edge) Chord Ainc [Nspan, Sspace
    already defined above]
0.298 0.0 0.093 0.335 0.0
AFILE
clark-y-mod.dat
# CLaf is the ratio between a thin-aerofoil dCl/dalpha (alpha in radians) of
    2*pi and the actual value
# Could use XFoil to get true value and work out CLaf as true/(2*pi), but data
    seems a bit odd, so using
# estimate of CLaf = 1 + 0.77 t/c where t/c is the aerofoil's thickness/chord
    ratio
CLAF
1.08
SECTION
# Wing section containing aileron
# Increase Zle to correspond to the wing's 3.7 degree dihedral
# Xle Yle Zle Chord Ainc
0.298 0.092 0.09894 0.335 0.0
AFILE
clark-y-mod.dat
CONTROL
aileron 1.0 0.8687 0.0 0.0 0.0 -1
CLAF
1.08
SECTION
# Wingtip
# Xle Yle Zle Chord Ainc
0.298 0.882 0.14990 0.335 0.0
AFILE
clark-y-mod.dat
CONTROL
aileron 1.0 0.8687 0.0 0.0 0.0 -1
CLAF
1.08
SECTION
# End of wing
# Xle Yle Zle Chord Ainc
0.298 0.92 0.15135 0.335 0.0
AFILE
clark-y-mod.dat
CLAF
1.08

SURFACE
Horizontal Stabaliser
# Nchord Cspace Nspan Sspace
8 1.0 20 -1.25
YDUPLICATE
0.0
# surface has a 4 degree angle measured
ANGLE
4.0
SECTION
# Xle Yle Zle Chord Ainc
```

```
1.122 0.0 -0.019 0.240 0.0
CONTROL
elevator 1.0 0.7917 0.0 0.0 0.0 1
SECTION
1.192 0.335 -0.019 0.170 0.0
CONTROL
elevator 1.0 0.7059 0.0 0.0 0.0 1
SECTION
# tip of h.stabaliser (no elevator control)
1.196 0.359 -0.019 0.166 0.0

SURFACE
Vertical Stabaliser
# Nchord Cspace Nspan Sspace
8 1.0 20 -1.25
SECTION
# Xle Yle Zle Chord Ainc
0.928 0.0 0.005 0.39 0.0
CONTROL
rudder -1.0 0.8468 0.0 0.0 0.0 1
SECTION
1.118 0.0 0.052 0.200 0.0
CONTROL
rudder -1.0 0.650 0.0 0.0 0.0 1
SECTION
1.194 0.0 0.245 0.124 0.0
CONTROL
rudder -1.0 0.43548 0.0 0.0 0.0 1
```

Listing A.2: AVL input for Trainer 60 .mass file

```
#****************************************
# Trainer 60 Test Aircraft
# Written by Matthew Bennett  October 2006
#****************************************

# Mass & Inertia Breakdown

Lunit = 1.0 m
Munit = 1.0 kg
Tunit = 1.0 s

g = 9.81
rho = 1.225

# mass  x       y       z       Ixx     Iyy     Izz
0.3595 0.231 0 -0.01 0.001640219 0.007214536 0.007214536 ! Fuselage (front)
0.3595 0.884 0 -0.03 0.000813369 0.021696544 0.021696544 ! Fuselage (back)
0.069 1.2 0 0.11 0.000374262 0.000549988 0.000176462 ! Fin
0.185 1.24 0 -0.02 0.006880042 0.000617422 0.007495953 ! Horiz. Stabaliser
0.08 0.105 0 -0.14 0.000228667 0.000228667 0.000001 ! Undercarrage (front)
0.027 0.465 0.05 -0.08 0 0 0 ! Undercarrage (starboard)
0.027 0.465 0.145 -0.15 0 0 0 ! Undercarrage (starboard)
0.027 0.465 0.24 -0.22 0 0 0 ! Undercarrage (starboard)
0.027 0.465 -0.05 -0.08 0 0 0 ! Undercarrage (port)
0.027 0.465 -0.145 -0.15 0 0 0 ! Undercarrage (port)
0.027 0.465 -0.24 -0.22 0 0 0 ! Undercarrage (port)
0.573 0.055 0 0.015 0.000621944 0.000908444 0.000575388 ! Engine
```

```
0.185 0.105 0.085 0.045 5.78125e-05 0.000448625 0.000448625 ! Exhaust
0.086 0 0 0 0.0036163 0.001808867 0.001808867 ! Propeller and spinner
0.465 0.19 0 0 0.00020925 0.0007595 0.0007595 ! Fuel tank (full)
# 0.091 0.19 0 0 6.93875e-05 0.000162852 0.000162852 ! Fuel tank (empty)
0.5825 0.451 0 0.093 0.029592214 0.005507052 0.034980339 ! Wings (centre)
0.291 0.451 0.655 0.093 0.006841531 0.002751163 0.009533281 ! Wings (startboard
    end)
0.291 0.451 -0.655 0.093 0.006841531 0.002751163 0.009533281 ! Wings (port end)
0.053 0.78 0 0.033 1.13818e-05 1.13818e-05 0.000020776 ! GPS Aerial
0.055 0.56 0 0.015 3.19733e-05 9.97333e-06 3.66667e-05 ! RX Box
0.12 0.53 0.005 -0.045 0.00009425 0.00011125 0.000145 ! Servo Cluster
0.412 0.411 0 0.028 0.000516408 0.000978672 0.000752381 ! Autopilot
0.153 0.38 0 -0.048 8.54888e-05 0.000149456 0.000196159 ! Comms Box
0.232 0.277 0 0.005 0.000185465 0.000159945 4.59747e-05 ! Battery (Servos)
0.323 0.195 0 -0.06 6.52729e-05 0.0004845 0.000518765 ! Battery (Autopilot)
```

Appendix B

Autopilot Hardware Timings

Table B.1 and Table B.2 summarise the hardware execution times of all key software components of the autopilot. From the timing information presented, in theory the worst-case execution time will occur when the following conditions are simultaneously met:

- Streaming flight data
- Travelling the wrong way while outside of an orbit being tracked by closing-speed tracking
- GPS update occurs
- External ADC's trigger both altitude and airspeed updates during the main flight control update

The resulting total execution time should be 13.217 ms, representing 92.19 % of the main flight control update period. When the software was modified to force such conditions to always occur, the actual execution time recorded was 13.600 ms or 94.87% of the main flight control update period. This discrepancy is believed to occur due to additional code executed by the autopilot that was not included in the component analysis. Such code would include stack read and write operations during interrupt handling, function call overheads and small amounts of miscellaneous code executing between timed procedures. The relative component timings under worst-case conditions are illustrated in Figure B.1.

In both cases however the worse-case code execution time remains within the main flight control update period and as such the autopilot is verified as capable

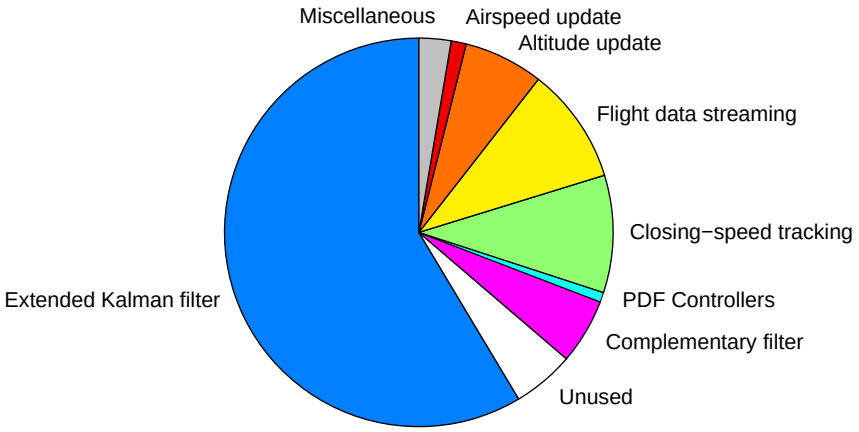


FIGURE B.1: Software component execution times under worst-case conditions

of meeting its control deadlines. The small amount (∼5% worst-case) of "spare" processing time highlights the need to include only essential and speed-optimised C code. During operation, this spare processing time is in fact be used to processes user commands received from the ground station. Since this occurs as a separate, low-priority process it does not interfere with the timing of the main flight control update. The majority of operations required following user commands require few processor instructions, and as such no response delay is perceived by the user even under worst-case main flight control execution conditions.

TABLE B.1: Hardware execution times of key autopilot software components

Function	Time (ms)	Process duty (%)
Main flight control update period	14.336	100.00
Extended Kalman filter (AHRS) iteration		
No GPS update	7.450	51.97
GPS update	8.400	58.59
Complementary filter (PRS) iteration		
No GPS update	0.540	3.77
GPS update	0.780	5.44
Single PDF controller iteration (1 of 3 in main flight control update)	0.039	0.27
Closing speed tracking iteration		
Line track		
Along-speed mode	0.105	0.73
Cross-speed mode	0.096	0.67
Wrong-way recovery mode	0.130	0.91
Orbit track (inside orbit)		
Along-speed mode	0.260	1.81
Cross-speed mode	0.250	1.74
Wrong-way recovery mode	0.620	4.32
Orbit track (outside orbit)		
Along-speed mode	1.320	9.21
Cross-speed mode	0.980	6.84
Wrong-way recovery mode	1.400	9.77
Data streaming (Worst-case, some SCI TX interruption)		
Sensors	0.480	3.35
Controller	0.320	2.23
Optimising	0.110	0.77
EKF	0.400	2.79
Flight	1.390	9.70
Navigation		
Normal	1.280	8.93
CST	0.400	2.79

TABLE B.2: Hardware execution times of software components linked to 20 Hz external ADC update

Function	Time (ms)	Process duty (%) (of main flight control update)
Altitude iteration		
Normal	0.950	6.63
With altitude controller streaming	0.980	6.84
Airspeed iteration		
Normal	0.180	1.26
With altitude controller streaming	0.200	1.40

Appendix C

PDF Controller Listing

Listing C.1: PDF controller implementation pseudo-code

```
loopD ← Kd × input
loopI ← loopI + Ki × (command − input) × update period

if loopI − loopD < minimum output then
begin
    loopI ← minimum output + loopD
    raise own minimum saturated flag
end

if loopI − loopD > maximum output then
begin
    loopI ← maximum output + loopD
    raise own maximum saturated flag
end

if slave is minimum saturated and loopI − loopD < last output
   then loopI ← last output + loopD
if slave is maximum saturated and loopI − loopD > last output
   then loopI ← last output + loopD

output ← loopI − loopD
last output ← output
```

Additionally, when the autopilot starts up or when the integral gain Ki is modified, the controller integral should be reset by loopI ←output + Kd ×input before

the controller code is executed again. In this case output may also be forced to a particular value, such as when setting the control surfaces to neutral positions.

Appendix D

True Airspeed Estimation

As mentioned in Section 3.3.4, the current autopilot hardware does not measure outside air temperature. While this functionality will be introduced in future hardware, Figure D.1 shows the autopilot's TAS estimate is still more accurate than CAS even in true air temperature variations of ± 30 K from the assumed T_0. The static pressure used for this graph was set at 79.5 kPa, equivalent to the highest expected altitude of 2 km at ISA sea level pressure. The TAS estimate will always be a more accurate than CAS when $T \geq T_0$. However, as altitude decreases TAS becomes closer to CAS, and so the amount by which the true air temperature can drop below T_0 reduces if TAS is to remain more accurate. Figure D.2 plots the minimum altitude that the TAS estimate remains more accurate for a given temperature drop. This maximum static pressure P_{smax} in kPa for a given

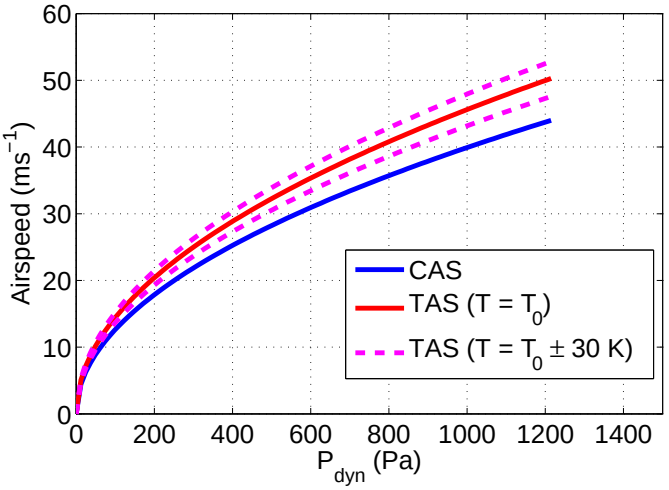


FIGURE D.1: Comparison of TAS with CAS over the expected dynamic pressure range (P_{static} = 79.5 kPa)

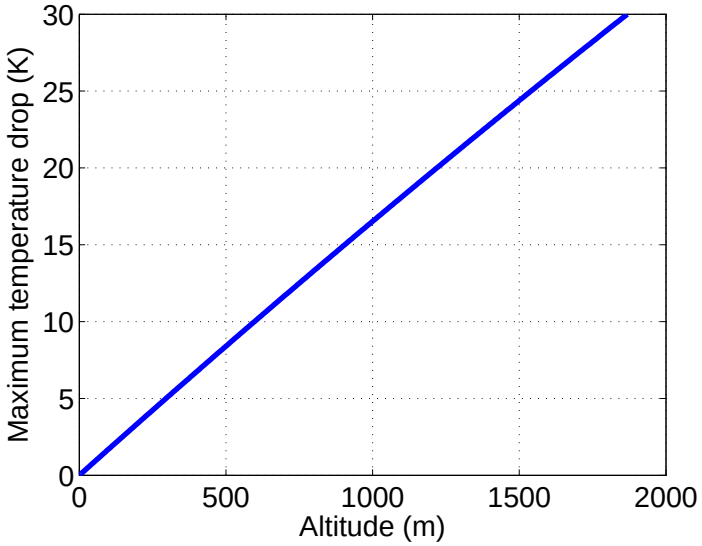


FIGURE D.2: Maximum temperature drop against minimum altitude that the fixed-temperature TAS estimate remains more accurate than CAS

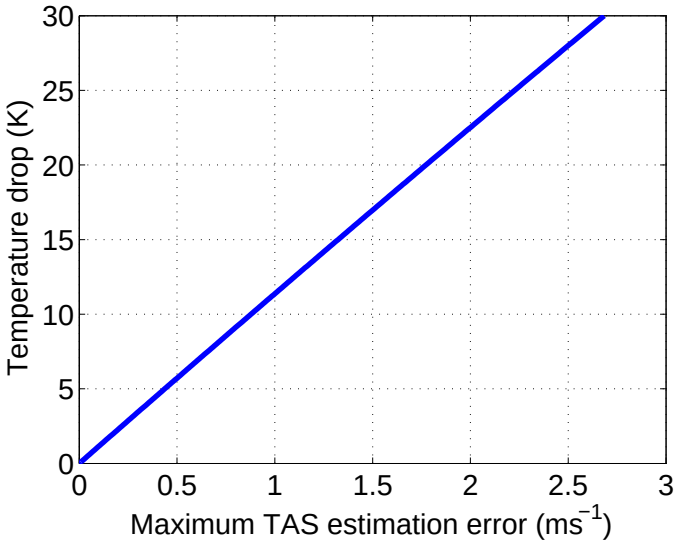


FIGURE D.3: Worst-case fixed-temperature TAS estimation error due to difference in true air temperature from T_0

temperature drop χ in K is expressed by Equation D.1. Note this is independent of P_{dyn}. P_{smax} was converted to altitude to produce Figure D.2.

$$P_{smax} = \frac{D_0}{1000}\left(2\sqrt{R(T_0-\chi)} - \sqrt{RT_0}\right)^2 \tag{D.1}$$

Figure D.3 shows the expected TAS estimation error due to difference in true air temperature from T_0 under worst-case conditions (lowest expected P_{static} and highest expected P_{dyn}). It is apparent the worst-case TAS error is less than 1.5 knots

for typical daily temperature variations $T_0 \pm 10$ K regardless of its comparison with CAS. As such, the fixed-temperature TAS estimate is considered acceptable for test flight conditions.

Appendix E

Phugoid Analysis

A marked longitudinal oscillation (and resulting airspeed and altitude oscillation) was observed while trying to set the airspeed or altitude controller to the relatively high response rates previously achieved (in simulation) when controlling altitude by elevation and airspeed by throttle. The ∼15 second period oscillations typically observed were initially believed to be solely due to the aircraft's Phugoid motion, though subsequent simulations suggested such behaviour is already adequately compensated for by the elevation controller. Phugoid motion was initiated at 120 seconds by a 2 second step change in elevator position from the cruise condition (set as straight and level at 18 ms^{-1}), before being returned to normal. All other control surfaces and throttle were set to fixed cruise positions, with the autopilot effectively disabled. The results of Figure E.1 confirm the interchange of kinetic and potential energy typical of such behaviour. Angle of attack remains relatively constant throughout. Enabling the elevation angle controller, commanding the equivalent cruise elevation, quickly dampens the Phugoid motion (Figure E.2). A similar response is observed after enabling the airspeed controller (commanding cruise speed). The latter response shows a slower recovery with some underlying oscillation, due to the less responsive airspeed control gains and increased control path propagation delay, though still performs satisfactorily.

Further investigation reveals the altitude controller is the source of the oscillations, though due to its tendency to reinforce the underlying Phugoid motion. Figure E.3 demonstrates this behaviour, in contrast to the dampening effect of the elevation and airspeed controllers. Increasing the trimmed airspeed from 18 to 22 ms^{-1} shows the increased kinetic energy increases the amplitude of the Phugoid motion, though in the uncontrolled state this oscillation reduces to the steady state condition in a similar time. However, with the introduction of altitude control

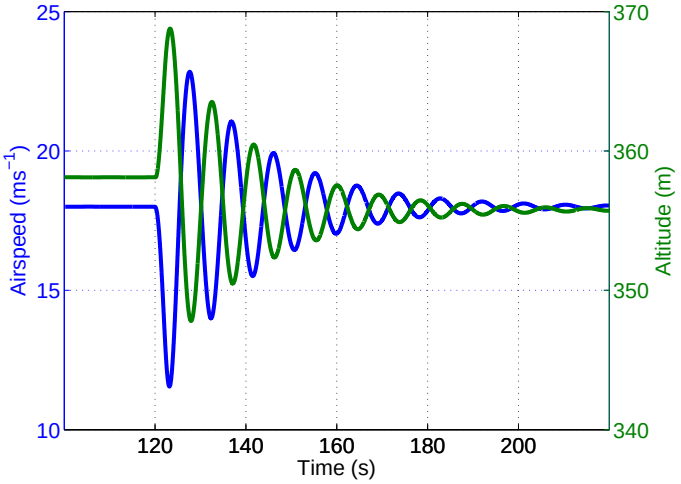


FIGURE E.1: Phugoid motion of simulated Trainer 60

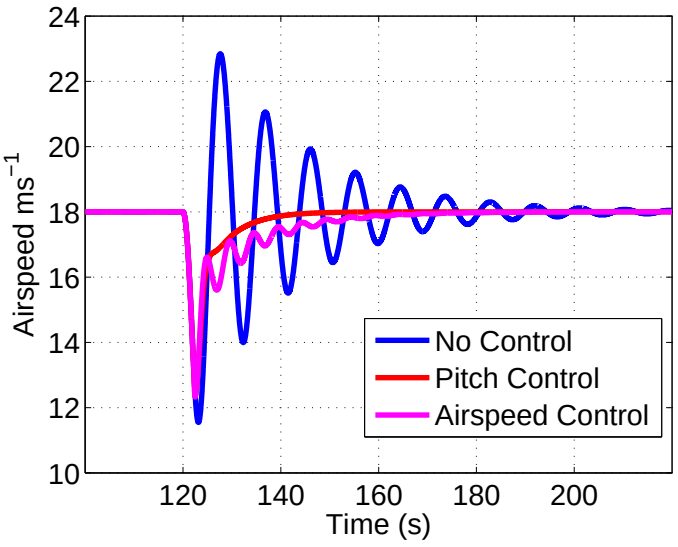


FIGURE E.2: Comparison between controlled and uncontrolled elevator response to Phugoid motion of simulated Trainer 60

the increased airspeed is shown to result in poorer performance, with sustained oscillation observed (Figure E.4). Increasing the responsiveness of the altitude controller is seen to cause a similar decrease in stability.

The airspeed controller is however capable of countering this behaviour and restoring stability in altitude control (Figure E.5). However, while the airspeed controller alone (with fixed throttle) is capable of a faster response to that chosen, increasing the response speed appears to reduce the dampening ability observed - particularly at high airspeeds (Figure E.6). Therefore while the altitude controller's enhancement of Phugoid oscillation is the true limiting factor, both altitude and airspeed controller response must be relatively slow. The former to reduce this enhancing

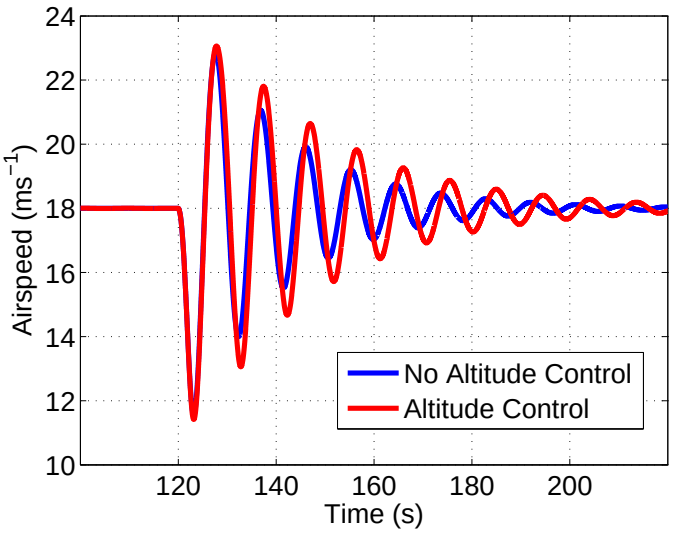


FIGURE E.3: Comparison between controlled and uncontrolled throttle response to Phugoid motion of simulated Trainer 60 (initial airspeed 18 ms^{-1})

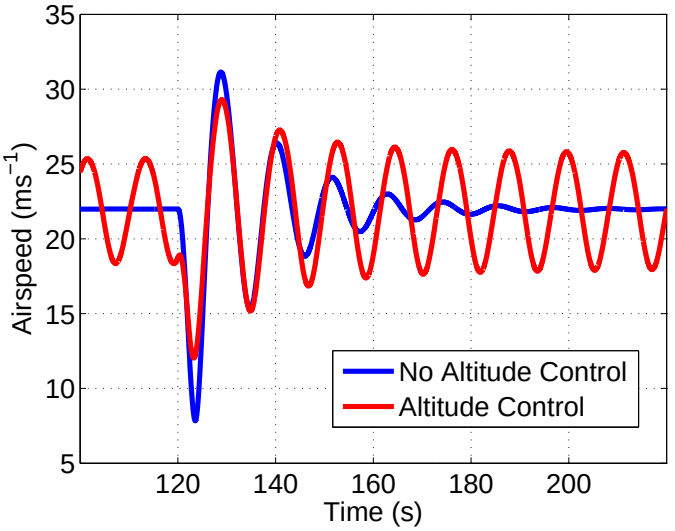


FIGURE E.4: Comparison between controlled and uncontrolled throttle response to Phugoid motion of simulated Trainer 60 (initial airspeed 22 ms^{-1})

effect in the beginning, the latter to better compensate for the remaining oscillation. While a fast airspeed response is still possible, this would be at the expense of an impractically slow altitude response. A compromise has therefore been reached in the airspeed and altitude gains selected. The slower airspeed response is considered acceptable due to gust resistance still being provided by the fast elevation angle controller, and a typical oceanographic mission being unlikely to require a rapid change in commanded airspeed.

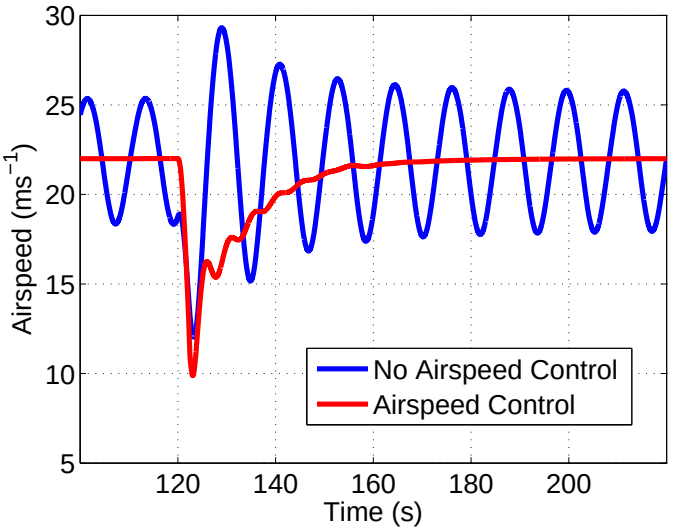


FIGURE E.5: Stabilisation of altitude controller response by airspeed control following Phugoid motion of simulated Trainer 60

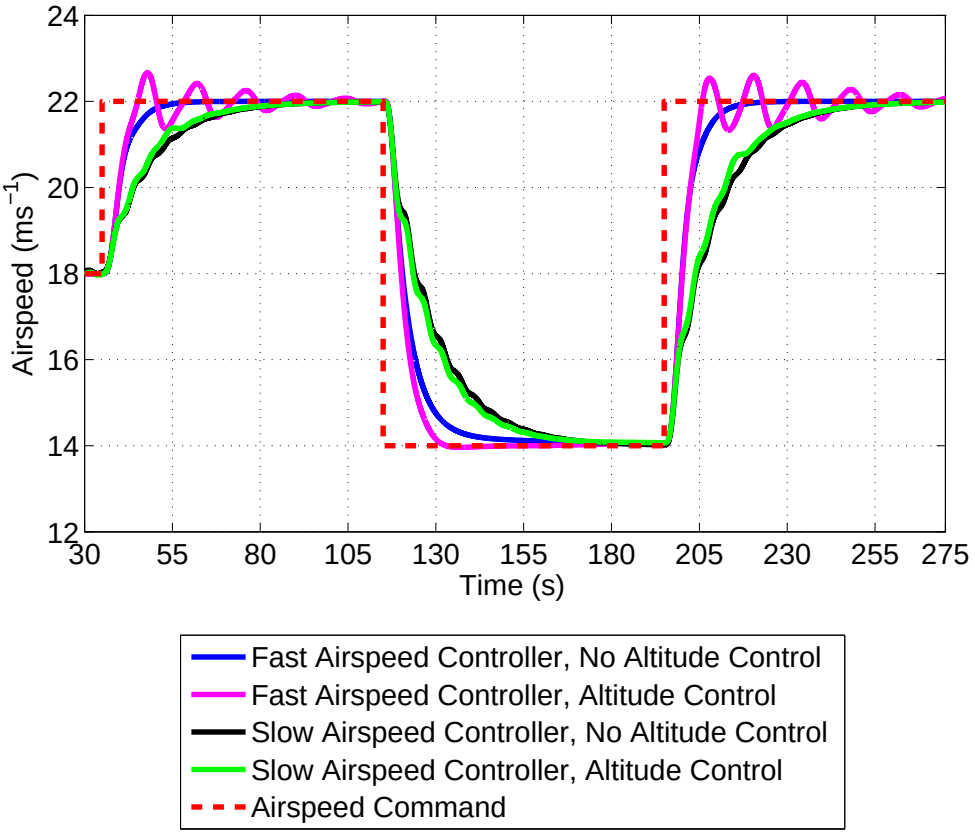


FIGURE E.6: Comparison of slow and fast airspeed controller responses to controlled and uncontrolled altitude for the simulated Trainer 60

References

[1] G. Welch and G. Bishop, "An introduction to the kalman filter," Tech. Rep. 95-041, University of North Carolina at Chapel Hill, July 2006.

[2] D. Kingston, R. Beard, and T. McLain, "Autonomous vehicle technologies for small fixed wing uavs," in *Proceedings of 2nd AIAA "Unmanned Unlimited" Conference*, San Diego, USA, 15-18 September 2003.

[3] B. Vaglienti, R. Hoag, and M. Niculescu, *Piccolo System User's Guide*, Cloud Cap Technology, Available from `http://www.cloudcaptech.com/resources_autopilots.shtm`, 1.3.2 edition, April 2008.

[4] M. Niculescu, "Lateral track control law for aerosonde uav," in *Proceedings of the 39th AIAA Aerospace Sciences Meeting and Exhibit*, Reno (NV), USA, 8-11 January 2001.

[5] G.J. Holland, P. J. Webster, J. A. Curry, G. Tyrell, D. Gauntlett, G. Brett, J. Becker, R. Hoag, and W. Vaglienti, "The aerosonde robotic aircraft: A new paradigm for environmental observations," *Bulletin of the American Meteorological Society*, vol. 82 part 5, pp. 889–902, 2001.

[6] E. Waugh, G. Pluck, R. Gilbertson, H. Lim, S. Roberts, M. McKinley, and D. Hatts, "An unmanned aerial vehicle for oceanographic applications," M.S. thesis, University of Southampton, Southampton, UK, 2003.

[7] A.S. Lomax, W. Corso, and J.F. Etro, "Employing unmanned aerial vehicles (uavs) as an element of the integrated ocean observing system," in *Proceedings of MTS/IEEE Oceans 2005*, Washington, DC, USA, 17-23 September 2005.

[8] R.D. Williams, "Studies of mixed-phase cloud microphysics using an in-situ unmanned aerial vehicle (uav) platform," M.S. thesis, Georgia Institute of Technology, Georgia, USA, August 2005.

[9] A.C. van den Kroonenberg, T. Spie, M. Buschmann, T. Martin, P.S. Anderson, F. Beyrich, and J. Bange, "Boundary layer measurements with the autonomous mini-uav m^2av," in *Proceedings of DACH2007*, Hamburg, Germany, 10–14 September 2007.

[10] R.T. Miles, J.A. Melhado, E.W. Hughes, and D. Osiecki, "Air-launched expendable micro-sized wave buoy," in *Proceedings of MTS/IEEE Oceans 2001*, Honolulu, USA, 5-8 November 2001, vol. 3, pp. 1867–1871.

[11] D.P. Horner and A.J. Healey, "Use of artificial potential fields for uav guidance and optimization of wlan communications," in *Proceedings of IEEE/OES Autonomous Underwater Vehicles*, Maine, USA, 17-18 June 2004, pp. 88–95.

[12] A. Finn, K. Brown, and A. Lindsay, "Miniature uavs & future electronic warfare," in *Proceedings of Land Warfare Conference 2002*, Brisbane, Australia, 22-24 October 2002.

[13] National Oceanography Centre National Marine Facilities, *Io Unmanned Aerial Vehicle*, `http://www.noc.soton.ac.uk/nmf/usl_index.php?page=iop`, 2006.

[14] B. Johnson, R. Joseph, M. Nischan, A. Newbury, J. Kerekes, H. Barclay, B. Willard, and J.J. Zayhowski, "A compact, active hyperspectral imaging system for the detection of concealed targets," in *Proceedings of SPIE Conference on Detection and Remediation Technologies for Mines and Minelike Targets IV*, Orlando, FL, USA, April 1999, vol. 3710, pp. 144–153.

[15] National Environmental Research Council, *Unmanned aerial vehicles mark robotic first for British Antarctic Survey*, `http://www.nerc.ac.uk/press/releases/2008/14-uav.asp`, 2008.

[16] J. Batterbee, M. Bennett, D Hart, S Hosking, P Mills, and K Roskilly, "Uninhabited aeronautical vehicle for oceanographic applications the next step," M.S. thesis, University of Southampton, Southampton, UK, 2004.

[17] Aerosonde Pty Ltd, *Aerosonde*, `http://www.aerosonde.com/`, 2006.

[18] Moire Incorporated, "Cost and business model analysis for civilian uav missions," Final report, National Aeronautics and Space Administration, June 2004.

[19] T. McGeer and J. Vagners, "Wide-scale use of long-range miniature aerosondes over the worlds oceans," in *Proceedings of AUVSI 26th Annual Symposium, Association for Unmanned Vehicle Systems International*, Baltimore, MD, USA, 12–16 July 1999.

[20] The Insitu Group Inc., *ScanEagle A-15 Product Sheet*, `http://www.insitu.com/documents/ScanEagleA-15_000.pdf`, 2004.

[21] The Insitu Group Inc., *SeaScan Product Sheet*, `http://www.insitu.com/documents/SeaScan.pdf.pdf`, 2004.

[22] J.W. Ramsey, *UAVs: Out of Uniform*, Avionics Magazine, `http://www.aviationtoday.com/av/categories/military/779.html`, 2004.

[23] P. Ashworth, "Unmanned aerial vehicles and the future navy," *Royal Australian Navy Working Paper 6*, May 2001.

[24] B.J.Carlson, *Past UAV Program Failures and Implications for Current UAV Programs*, Air Command and Staff College, Maxwell Air Force Base, AL USA, April 2001.

[25] J. Ott and D. Biezad, "Design of a tube-launched uav," in *Proceedings of AIAA 3rd "Unmanned Unlimited" Technical Conference*, Chicago, USA, 20-23 September 2004.

[26] G. Pisanich and S. Morris, "Fielding an amphibious uav: development, results, and lessons learned," in *Proceedings of the 21st Digital Avionics System Conference 2002*, Irvine (CA), USA, 27-31 October 2002, vol. 2, pp. 8C4–1 – 8C4–9.

[27] G. Avanzini and G. de Matteis, "Design and analysis of a shipboard recovery system for a shrouded-fan uav," in *Proceedings of the 23rd ICAS Congress*, Toronto, Canada, 8-13 September 2002.

[28] M. Wills, A. Burmeister, T. Nelson, T. Denewiler, and K. Mullens, "The development of a ugv-mounted automated refueling system for vtol uavs," in *Proceedings of SPIE Unmanned Systems Technology Conference VIII*, Orlando, USA, 12 May 2006, vol. 6230.

[29] G. Avanzini, S. D'Angelo, and G. de Matteis, "Design and development of a vertical take-off and landing uninhabited aerial vehicle," *Proceedings of the Institution of Mechanical Engineers Part G Journal of Aerospace Engineering*, vol. 217, no. 4, pp. 169–178, August 2003.

[30] M. Quigley, B. Barber, S. Griffiths, and M.A. Goodrich, "Towards real-world searching with fixed-wing mini-uavs," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems 2005*, Edmonton, Canada, 2-6 August 2005, pp. 3028–3033.

[31] D.R. Nelson, D.B. Blake, T.W. McLain, and R.W. Beard, "Vector field path following for small unmanned air vehicles," in *Proceedings of American Control Conference*, Minneapolis, USA, 14-16 June 2006.

[32] R.S. Christiansen, "Design of an autopilot for small unmanned aerial vehicles," M.S. thesis, Brigham Young University, Utah, USA, 2004.

[33] E.T. King, "Distributed coordination and control experiments on a multi-uav testbed," M.S. thesis, The State University of Baffalo, Buffalo (NY), USA, 2002.

[34] E.N. Johnson, S.G. Fontaine, and A.D. Kahn, "Minimum complexity uninhabited air vehicle guidance and flight control system," in *Proceedings of The 20th Digital Avionics Systems 2001*, Daytona Beach (FL), USA, 14-18 October 2001.

[35] J. How, E. King, and Y. Kuwata, "Flight demonstrations of cooperative control for uav teams," in *Proceedings of the AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, Chicago, USA, 20-23 September 2004.

[36] Tower Hobbies, *Tower Hobbies' Easy RC - Airplanes*, `http://www.easyrc.com/airplanes/index.html`, 2006.

[37] T.W. McLain and R.W. Beard, "Unmanned air vehicle testbed for cooperative control experiments," in *Proceedings of the American Control Conference 2004*, Boston (MA), USA, 30 June-2 July 2004, vol. 6, pp. 5327–5331.

[38] J.A. Miller, P.D. Minear, A.F. Niessner Jr., A.M. DeLullo, B.R. Geiger, L.N. Long, and J.F. Horn, "Intelligent unmanned air vehicle flight systems," in *Proceedings of Infotech@Aerospace 2005*, Arlington (VA), USA, 26-29 September 2005.

[39] S. Rodgers and S. Yokum, "Reconfigurable adaptive autopilot system for man portable fixed wing uninhabited aerial vehicles," in *Proceedings of Infotech@Aerospace 2005*, Arlington (VA), USA, 26-29 September 2005.

[40] R. Shivkumar, V.V. Gupta, H. Arya, P.S. Gandhi, and K. Sudhakar, "System testing & integration," in *Proceedings of the All India Seminar on Mini and Micro Aerial Vehicles*, Pune, India, February 2004, India Institude of Technology.

[41] D. R. Maroney, R. H. Bolling, M. E. Heffron, and G. W. Flathers, "Experiemntal platforms for evaluating sensor technology for uas collision avoidance," in *Proceedings of IEEE/AIAA 26th Digital Avionics Systems Conference*, Dallas (TX), USA, 21-25 October 2007.

[42] E.W. Frew, X. Xiao, S. Spry, T. McGee, Z.W. Kim, J. Tisdale, R. Sengupta, and J.K. Hedrick, "Flight demonstrations of self-directed collaborative navigation of small unmanned aircraft," in *Proceedings of AIAA 3rd "Unmanned Unlimited" Technical Conference*, Chicago, USA, 20-23 September 2004.

[43] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta, "Vision-based road-following using a small autonomous aircraft," in *Proceedings of IEEE Aerospace Conference 2004*, Big Sky (MT), USA, 6-13 March 2004.

[44] K. Schulze, A. Abramson, and B. Rogan, "An economical approach to autopilot development and integration," in *Proceedings of AIAA 3rd "Unmanned Unlimited" Technical Conference*, Chicago, USA, 20-23 September 2004.

[45] S. Bayraktar, G. Fainekos, and G.J. Pappas, "Hybrid modeling and experimental cooperative control of multiple unammned aerial vehicles," Tech. Rep., Department of Computer and Information Science, University of Pennsylvania, 2004.

[46] M.J. Matczynski, "A distributed embedded software architecture for multiple unmanned aerial vehicles," M.S. thesis, Massachusetts Institute of Technology, Cambridge (MA), USA, 2006.

[47] R. Becker, C. Borowski, O. Petrovic, C. Kitts, and N. Quinn Jr., "Scu aerial robotics team - experimentation with an autonomous uav observation platform," in *Proceedings of the 2004 AUVSI Unmanned Systems Conference*, Anaheim (CA), USA, August 2004.

[48] J. Evans, G. Inalhan, J.S. Jang, R. Teo, and C.J. Tomlin, "Dragonfly: A versatile uav platform for the advancement of aircraft navigation and

control," in *Proceedings of The 20th Digital Avionics Systems 2001*, Daytona Beach (FL), USA, 14-18 October 2001.

[49] J.S. Jang and C.J.Tomlin, "Autopilot design for the stanford dragonfly uav: Validation through hardware-in-the-loop simulation," in *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Canada, 6-9 August 2001.

[50] R. Teo, J.S. Jang, and C.J. Tomlin, "Automated multiple uav flight - the stanford dragonfly uav program," in *Proceedings of the 43rd IEEE Conference on Decision and Control 2004*, Atlantis, Bahamas, 14-17 December 2004.

[51] J. Evans, S. Houck, G. McNutt, and B. Parkinson, "Integration of a 40 channel gps receiver for automatic control into an unmanned airplane," in *Proceedings of the 11th International Technical Meeting of the Satellite Division of the Institute of Navigation*, Nashville (TN), USA, 15-18 September 1998, pp. 1173–1180.

[52] L.J. Horng, "Unmanned air vehicle flight control," M.S. thesis, National University of Singapore, Singapore, 2005.

[53] C.E. Hall Jr., "A real-time linux system for autonomous navigation and flight attitude control of an uninhabited aerial vehicle," in *Proceedings of The 20th Digital Avionics Systems 2001*, Daytona Beach (FL), USA, 14-18 October 2001.

[54] S. Angermuller and C.E. Hall Jr., "Flight test validation of autonomous uav with autoreturn function," in *Proceedings of 2nd AIAA "Unmanned Unlimited" Conference*, San Diego, USA, 15-18 September 2003.

[55] D. Caltabiano, G. Muscato, A. Orlando, C. Federico, G. Giudice, and S. Guerrieri, "Architecture of a uav for volcanic gas sampling," in *Proceedings of the 10th IEEE Conference on Emerging Technologies and Factory Automation 2005*, Catania, Italy, 19-22 September 2005.

[56] MicroPilot, *MicroPilot - World Leader in Small UAV Autopilots*, `http://www.micropilot.com/`, 2008.

[57] MicroPilot, *MP2028g Installation and Operation*, Available: `http://www.micropilot.com/Manual-MP2028.pdf`, 2005.

[58] G. Wigley and M. Jasiunas, "A low cost, high performance reconfigurable computing based unmanned aerial vehicle," in *Proceedings of IEEE Aerospace Conference 2006*, Big Sky (MT), USA, 4-11 March 2006.

[59] G. Platanitis and S. Shkarayev, "Integration of an autopilot for a micro air vehicle," in *Proceedings of Infotech@Aerospace 2005*, Arlington (VA), USA, 26-29 September 2005.

[60] H. Grankvist, "Autopilot design and path planning for a uav," Tech. Rep. FOI-R-2224-SE, Swedish Defense Research Agency (FOI), December 2006.

[61] Cloud Cap Technology Inc., *Cloudcap Technology - Avionics for Autonomous UAV Autopilots*, `http://www.cloudcaptech.com/`, 2006.

[62] Cloud Cap Technology Inc., *Cloud Cap Capabilities Brochure*, `http://www.cloudcaptech.com/misc/CCT_Capabilities.pdf`, 2006.

[63] J. Elston, B. Argrow, and E. Frew, "A distributed avionics package for small uavs," in *Proceedings of Infotech@Aerospace 2005*, Arlington (VA), USA, 26-29 September 2005.

[64] E.W. Frew, C. Dixon, B. Argrow, and T. Brown, "Radio source localization by a cooperating uav team," in *Proceedings of Infotech@Aerospace 2005*, Arlington (VA), USA, 26-29 September 2005.

[65] S. Bayraktar, G.E. Fainekos, and G.J. Pappas, "Experimental cooperative control of fixed-wing unmanned aerial vehicles," in *Proceedings of the 43rd IEEE Conference on Decision and Control 2004*, Atlantis, Bahamas, 14-17 December 2004.

[66] J. Lee, R. Huang, A. Vaughn, X. Xiao, K. Hedrick, M. Zennaro, and R. Sengupta, "Strategies of path-planning for a uav to track a ground vehicle," in *Proceedings of the 2nd Annual Symposium on Autonomous Intelligent Networks and Systems*, Menlo Park (CA), USA, 30 June - 1 July 2003.

[67] UAV Flight Systems Inc., *UAV Flight Systems - Unmanned Aerial Vehicles*, `http://www.uavflight.com/`, 2006.

[68] Unav LLC., *UNAV3400 Miniature UAV Autopilot*, `http://www.u-nav.com/3400BRCH.pdf`, 2006.

[69] T. Hudson, *Autopilot: Do it yourself UAV*, SourceForge, `http://autopilot.sourceforge.net/`, 2005.

[70] R. De Nardi, O. Holland, J. Woods, and A. Clark, "Swarmav: A swarm of miniature aerial vehicles," in *Proceedings of the 21st Bristol International UAV Systems Conference*, Bristol, UK, 3-5 April 2006.

[71] S. Doncieux, J. B. Mouret, L. Muratet, and J.A. Meyer, "The robur project: towards an autonomous flapping-wing animat," *Proceedings of the Journées MicroDrones*, 2004.

[72] Freescale Semiconductor, *MPC555*, `http://www.freescale.com/files/microcontrollers/doc/fact_sheet/MPC555FACT.pdf`, 2005.

[73] Advanced Micro Devices Inc., *Am186TMES/ESLV and Am188TM ES-/ESLV*, `20002.pdf` from `www.amd.com/epd/processors/2.16bitcont/2.am186exfa/6.am186es/a20002/`, 2002.

[74] D. Borys and R. Colgren, "Advances in intelligent autopilot systems for unmanned aerial vehicles," in *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco (CA), USA, 15-18 August 2005.

[75] K. van der Molen, *Feature tracking using vision on an autonomous airplane*, Autonomous Systems Lab, Swiss Federal Institute of Technology, `rapportVanderMolen.pdf` from `http://asl.epfl.ch/research/projects/VtolIndoorFlying/rapports/`, 2004.

[76] UAV Flight Systems Inc., *AP50 AutoPilot Flight Control System Manual*, `http://www.uavflight.com/docs/AP50%20System%20Manual.pdf`, 2006.

[77] Crossbow Technology, *MNAV100CA*, `www.xbow.com/Products/Product_pdf_files/Inertial_pdf/uNAV_Datasheet.pdf`, datasheet 6020-0083-02 rev b edition, 2006.

[78] Atmel Corporation, *ATmega128 / ATmega128L Summary*, `http://www.atmel.com/dyn/resources/prod_documents/2467S.pdf`, 2467ns-avr-03/06 edition, 2006.

[79] Crossbow Technology, *Stargate*, `www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Stargate_Datasheet.pdf`, datasheet 6020-0049-03 rev a edition, 2006.

[80] Crossbow Technology Inc., *SourceForge.net: MNAV Autopilot*, `http://sourceforge.net/projects/micronav`, 2006.

[81] Gumstix Incorporated, *Gumstix - dream, design, deliver*, `http://www.gumstix.com/`, 2008.

[82] R. Ellen, P. Roberts, and D. Greer, "An investigation into the next generation avionics architecture for the qut uav project," in *Proceedings of Smart Systems 2005 Postgraduate Research Conference*, Brisbane, Australia, 2005.

[83] O-Navi LLC., *OEM Inertial Sensors: Phoenix AX*, `http://www.o-navi.com/phoenix_ax.htm`, 2004.

[84] Freescale Semiconductor Inc., *MMC2114 MCORE Microcontroller Product Brief*, Available from `http://www.freescale.com/files/32bit/doc/prod_brief/MMC2114PB.pdf`, mmc2114pb/d rev. 0 edition, 2002.

[85] J. Langelaan and S. Rock, "Towards autonomous uav flight in forests," in *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco (CA), USA, 15-18 August 2005.

[86] Rabbit Semiconductor, *RCM3100 RabbitCore Control Module*, `http://www.rabbitsemiconductor.com/products/rcm3100/rcm3100.pdf`, 2005.

[87] Procerus Technologies, *Kestrel Autopilot v2.22*, `http://www.procerusuav.com/Documents/Kestrel_2.22.pdf`, 2006.

[88] Inc. Freescale Semiconductor, *MC68336/376 User's Manual: Introduction*, `http://www.freescale.com/files/microcontrollers/doc/user_guide/MC68336376UM01.pdf`, rev.15 edition, 2000.

[89] A.R. Girard, A.S. Howell, and J.K. Hedrick, "Border patrol and surveillance missions using multiple unmanned air vehicles," in *Proceedings of the 43rd IEEE Conference on Decision and Control 2004*, Atlantis, Bahamas, 14-17 December 2004, vol. 1, pp. 620–625.

[90] MicroStrain, *3DM-G & 3DM-GX1 Gyro Enhanced Orientation Sensor FAQ's*, `http://www.microstrain.com/manuals/3DM-GFAQ.pdf`, 2004.

[91] Y. Q. Chen H. Chao, Y. Cao, "Autopilots for small fixed-wing unmanned air vehicles: A survey," in *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation*, Harbin, China, 5-8 August 2007.

[92] C. Arnold, N. Desch, E. Holk, C. Humbert, J. Krall, K. Wijesekera, and J. Yoder, "Rose-hulman institute of technology autonomous helicopter for the 2005 international aerial robotics competition," Association for

Unmanned Vehicle Systems International (AUVSI), International Aerial Robotics Competition 2005, 2005.

[93] B. Tweddle, N. Mahendran, A. Phillip, J. Gillham, S. Peleato, M. Black, and D. Wang, "A network-based implementation of an aerial robotic system," Association for Unmanned Vehicle Systems International (AUVSI), International Aerial Robotics Competition 2004, 2004.

[94] N. Holifield, J. Lallinger, and G. Underwood, "International aerial robotics competition 2004," Association for Unmanned Vehicle Systems International (AUVSI), International Aerial Robotics Competition 2004, 2004.

[95] Xsens Technologies B.C., *MTi - Miniature Heading and Attitude Reference System*, `http://www.xsens.com/download/MTi_leaflet.pdf`, 2006.

[96] Crossbow Tech., *AHRS400*, `http://www.xbow.com/Products/Product_pdf_files/Inertial_pdf/AHRS400CD_Datasheet.pdf`, 6020-0025-12 rev b edition, 2006.

[97] J.S. Jang and C.J. Tomlin, "Design and implementation of a low cost, hierarchical and modular avionics architecture for the dragonfly," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey (CA), USA, 5-8 August 2002.

[98] R.H. Stone, "Control architecture for a tail-sitter unmanned air vehicle," in *Proceedings of the 5th Asian Control Conference*, Melbourne, Australia, 20-23 July 2004, vol. 2, pp. 736–744.

[99] C. Hide and T. Moore, "Gps and low cost ins integration for positioning in the urban environment," in *Proceedings of the Institute of Navigation GNSS 2005*, Long Beach (CA), USA, 13-16 September 2005.

[100] J. Skaloud, "Reliability of direct georeferencing phase 0," *Euro SDR Commission 1: Sensors, Primary Data Acquisition and Georeferencing*, 9 May 2006.

[101] D.B. Kingston and R.W. Beard, "Real-time attitude and position estimation for small uavs using low-cost sensors," in *Proceedings of AIAA 3rd "Unmanned Unlimited" Technical Conference*, Chicago, USA, 20-23 September 2004.

[102] D. Gebre-Egziabher, R.C. Hayward, and J.D. Powell, "Design of multi-sensor attitude determination systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 2, pp. 627–649, April 2004.

[103] D. Simon, "Kalman filtering," *Embedded Systems Programming*, pp. 72–29, June 2001.

[104] M. Niculescu, "Sensor fusion algorithms for unmanned air vehicles," in *Information, Decision and Control (IDC) Conference*, Adelaide, Australia, 11-13 February 2002, pp. 65–70.

[105] J.A. Rios and E. White, "Fusion filter algorithm enhancements for a mems gps/imu," in *Proceedings of the 14th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS 2001)*, Salt Lake City (UT), USA, 11-14 September 2001, pp. 1382–1393.

[106] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practise using MATLAB*, John Wiley and Sons Inc., second edition, 2001.

[107] J.L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*, CRC Press, 2004.

[108] P. W. McBurney, "A robust approach to reliable real-time kalman filtering," in *Proceedings of IEEE PLANS'90 Position Location and Navigation Symposium*, Las Vegas (NV), USA, 20-23 March 1990, pp. 549–556.

[109] G. Gopalratnam and C. Zorn, "Flight path reconstruction for sensor failure detection and health monitoring," Tech. Rep. PD FC 0418, National Aerospace Laboratories (India), December 2004.

[110] G. Pedersen, *Estimation and Sensor Information Fusion, Lecture 5*, Available at: `http://www.cs.aaue.dk/contribution/courses/spring2007/IRS8/ESIF/lecture5.article.pdf`, 2007.

[111] R. S. Bucy and P. D. Joseph, *Filtering for Stochastic Processes with Applications to Guidance*, Interscience Publishers, New York, NY, USA, 1968.

[112] C. G. Prevost, A. Desbiens, and E. Gagnon, "Extended kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle," in *Proceedings of American Control Conference ACC'07*, New York (NY), USA, 9-13 July 2007, pp. 1805–1810.

[113] K. Lievens, "Single gps antenna attitude determination of a fixed wing aircraft aided with aircraft aerodynamics," M.S. thesis, Faculty of Aerospace Engineering, Delft University of Technology, 2004.

[114] M. St-Pierre and D. Gingras, "Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system," in *IEEE 2004 Intelligent Vehicles Symposium*, Parma, Italy, 14-17 June 2004, pp. 831–835.

[115] T.J. Yeh, C.Y. Su, and W.J. Wang, "Modelling and control of a hydraulically actuated two-degree-of-freedom inertial platform," *Proceedings of the Institution of Mechanical Engineers I, Journal of Systems and Control Engineering*, vol. 219, no. 6, pp. 405–417, September 2005.

[116] S. Merhav and M. Velger, "Compensating sampling errors in stabilizing helmet-mounted displays using auxiliary acceleration measurements," *Journal of Guidance, Control, and Dynamics*, vol. 14, pp. 1067–1069, September-October 1991.

[117] J.M. Roberts, P.I. Corke, and G. Buskey, "Low-cost flight control system for a small autonomous helicopter," in *Proceedings of the 2002 Australasian Conference on Robotics and Automation*, Auckland, Australia, 27-29 November 2002.

[118] A. Baerveldt and R. Klang, "A low-cost and low-weight attitude estimation system for an autonomous helicopter," in *Proceedings of IEEE International Conference on Intelligent Engineering Systems*, Budapest, Hungary, 15-17 September 1997, pp. 391–395.

[119] S. Saripalli, J.M. Roberts, P.I. Corke, and G. Buskey, "A tale of two helicopters," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, USA, 27-31 October 2003, pp. 805–810.

[120] D. Kingston and R. Beard, "Real-time attitude and position estimation for small uavs using low-cost sensors," in *AIAA 3rd Unmanned Unlimited Technical Conference Workshop and Exhibit*, Chicago (IL), USA, 20-23 September 2004.

[121] A. M. Sabatini, "Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 7, pp. 1346–1356, July 2006.

[122] J. L. Marins, X. Yun, E. R. Bachmann, R. B. McGhee, and M. J. Zyda, "An extended kalman filter for quaternion-based orientation estimation using marg sensors," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii, USA, 29 October – 3 November 2001.

[123] E. R. Bachmann, *Inertial and Magnetic Angle Tracking of Limb Segments for inserting Humans into Synthetic Environments*, Ph.D. thesis, Naval Postgraduate School, Monterey (CA), USA, 2000.

[124] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*, Princeton University Press, 2002.

[125] R. Azuma and G. Bishop, "Improving static and dynamic registration in an optical see-through hmd," in *Proceedings of SIGGRAPH '94*, Orlando, FL, USA, 24–29 July 1994.

[126] The MathWorks, *MATLAB and Simulink*, `http://www.mathworks.com/`, 2007.

[127] M. J. Baker, *Maths - Conversion Quaternion to Euler*, `http://www.euclideanspace.com/maths/geometry/rotations/conversions/`, 2008.

[128] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," Tech. Rep., Stanford University, October 2006.

[129] S. B. Choe, *Statistical Analysis of Orientation Trajectories via Quaternions with Applications to Human Motion*, Ph.D. thesis, University of Michigan, Ann Arbor, MI, USA, 2006.

[130] J.M. Ham, "Remote sensing using remote controlled airplanes: Monitoring vegetation near eddy-covariance towers," in *Poster Presentation at American Society of Agronomy Annual Meetings*, Denver (CO), USA, 1-6 November 2003.

[131] A. Drouin and P. Brisset, *Paparazzi: Quick and Dirty UAV*, `http://paparazzi.enac.fr/wiki/index.php/Main_Page`, 2006.

[132] B. Taylor, C. Bil, and S. Watkins, "Horizon sensing attitude stabilisation: A vmc autopilot," in *18th International UAV Systems Conference*, Bristol, UK, 31 March - 2 April 2003.

[133] E. Favey, M. Cerniar, M. Cocard, and A. Geiger, "Sensor attitude determination using gps antenna array and ins," in *ISPRS WG III/1 Workshop: Direct Versus Indirect Methods of Sensor Orientation*, Barcelona, Spain, 25-26 November 1999.

[134] B.R. Woodley, H.L. Jones, E.A. LeMaster, E.W. Frew, and S.M. Rock, "Carrier phase gps and computer vision for control of an autonomous helicopter," in *Proceedings of the 9th International Technical Meeting of the Satellite Division of the Institute of Navigation*, Kansas City (MO), USA, 17-20 September 1996, pp. 461–465.

[135] J. Evans, W. Hodge, J. Liebman, C.J. Tomlin, and B.W. Parkinson, "Flight tests of an unmanned air vehicle with integrated multi-antenna gps receiver and imu: Towards a testbed for distributed control and formation flight," in *Proceedings of the 12th International Technical Meeting of the Satellite Division of the Institute of Navigation*, Nashville (TN), USA, 14-17 September 1999, pp. 1799–1808.

[136] A. Simsky, L.V. Kuylen, and F. Boon, "Single-board attitude determination system based on the polarx2@ gps receiver," in *Proceedings of the 18th International Technical Meeting of the Satellite Division of the Institute of Navigation*, Long Beach (CA), USA, 13-16 September 2005.

[137] M. Zhuang and D. P. Atherton, "Automatic tuning of optimum pid controllers," *IEE Proceedings D Control Theory and Applications*, vol. 140 issue 3, pp. 216–224, May 1993.

[138] Z. Y. Zhao, M. Tomizuka, and S. Isaka, "Fuzzy gain scheduling of pid controllers," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23 issue 5, pp. 1392–1398, Sept 1993.

[139] P. Eng, L. Mejias, R. A. Walker, and D. L. Fitzgerald, "Simulation of a fixed-wing uav forced landing with dynamic path planning," in *Proceedings of the 2007 Australasian Conference on Robotics and Automation*, Brisbane, Australia, 10–12 December 2007.

[140] B. Vaglienti, R. Hoag, and T. Miller, *Piccolo Aircraft Integration Guidelines*, Cloud Cap Technology Inc., Available from `http://www.cloudcaptech.com/resources_autopilots.shtm`, August 2006.

[141] R.M. Phelan, *Automatic Control Systems*, Cornell Univ. Press, 1977.

[142] I. Ž. Nikolić and I. Milivojević, "Application of pseudo-derivative feedback in industrial robots controllers," *Facta Universitatis: Mechanics, Automatic Control and Robotics*, vol. 2, no. 8, pp. 741–756, 1998.

[143] B. J. Huang and Y. C. Chen, "System dynamics and control of a linear compressor for stroke and frequency adjustment," *Journal of Dynamic Systems, Measurement, and Control*, vol. 124, pp. 176–182, March 2002.

[144] Z. Nagy and A. Bradshaw, "Comparison of pi and pdf controls of a manipulator arm," in *Proceedings of UKACC International Conference of Control '98*, Swansea, UK, 1-4 September 1998, vol. 1, pp. 739–744.

[145] A. Vahedipour and J. P. Bobis, "Smart autopilots," in *Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation*, San Diego (CA), USA, 9-13 November 1992, vol. 3, pp. 1437–1442.

[146] A. Vahedipour and J. P. Bobis, "Cascade pseudo-derivative feedback control algorithm and its application to design of autopilots," in *Proceedings of the IECON '93, International Conference on Industrial Electronics, Control and Instrumentation*, Maui, Hawaii, USA, 15-19 November 1993, vol. 3, pp. 2368–2372.

[147] P. N. Paraskevopoulos, G. D. Pasgianos, and K. G. Arvanitis, "New tuning and identification methods for unstable first order plus dead-time processes based on pseudoderivative feedback control," *IEEE Transactions on Control Systems Technology*, vol. 12, pp. 455–464, May 2004.

[148] L. Pernebo and B. Hansson, "Plug and play in control loop design," in *Preprints of Reglermote 2002*, Linkoping, Sweden, 29-39 May 2002.

[149] B. Vaglienti, R. Hoag, and M. Niculescu, *Piccolo System User's Guide*, Cloud Cap Technology, Available from `http://www.cloudcaptech.com/resources_autopilots.shtm`, 2.0.4 edition, March 2008.

[150] B. Vaglienti, *Piccolo Software Roadmap*, Cloud Cap Technology, Available from `http://www.cloudcaptech.com/resources_autopilots.shtm`, 2.1.0 edition, March 2008.

[151] M. Patcher, N. Ceccarelli, and P. R. Chandler, "Estimating mav's heading and the wind speed and direction using gps, inertial, and air speed measurements," in *Proceedings of AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, 18–21 August 2008.

[152] A. Bhatia, M. Graziano, S. Karaman, R. Naldi, and E. Frazzoli, "Dubins trajectory tracking using commercial off-the-shelf autopilots," in *Proceedings of AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, 18–21 August 2008.

[153] R. L. McNeely and R. V. Iyer, "Tour planning for an unmanned air vehicle under wind conditions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1299–1306, Sept 2007.

[154] L. E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1954.

[155] E.P. Anderson, "Extremal control and unmanned air vehicle trajectory generation," M.S. thesis, Department of Electrical and Computer Engineering, Brigham Young University, Utah, USA, April 2002.

[156] M. Shanmugavel, A. Tsourdos, B. A. White, and R. Zbikowski, "Differential geometric path planning of multiple uavs," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 620–632, Sept 2007.

[157] Cloud Cap Tech., *Steps to Autonomous Flight*, Available from `http://www.cloudcaptech.com/resources_autopilots.shtm`, 1.1 edition, August 2003.

[158] B. Vaglienti, M. Niculescu, and J. Becker, *HIL/SIL simulator for the Piccolo avionics*, Available from `http://www.cloudcaptech.com/resources_autopilots.shtm`, 2.0.4 edition, August 2007.

[159] M. Derela and H. Youngren, *Athena Vortex Lattice*, `http://web.mit.edu/drela/Public/web/avl/`, 2007.

[160] J. Becker, *Creating Vortex Lattice Aircraft Models for the Piccolo Simulator with AVL*, Available from `http://www.cloudcaptech.com/download/Piccolo/AircraftModelingTools/`, March 2008.

[161] E. G. Garcia and J. Becker, "Uav stability derivatives estimation for hardware-in-the-loop simulation of piccolo autopilot by qualitative flight testing," in *1st Latin American UAV Conference*, Panama City, Panama, 14-17 August 2007.

[162] A. Ficola, M. L. Fravolini, V. Brunori, and M. La Cava, "A simple control scheme for mini unmanned aerial vehicles," in *Proceedings of 14th Mediterranean Conference on Control and Automation*, Ancona, Italy, 28-30 June 2006.

[163] M. Abdulrahim and R. Lind, "Control and simulation of a multi-role morphing micro air vehicle," in *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, USA, 15-18 August 2005.

[164] S. Park, *Avionics and control system development for mid-air rendezvous of two unmanned aerial vehicles*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge (MA), USA, 2004.

[165] M. Mana, A. Desbiens, and E. Gagnon, "Identification of a uav and design of a hardware-in-the-loop system for nonlinear control purposes," in *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco (CA), USA, 15–18 August 2005.

[166] Unmanned Dynamics LLC, *AeroSim Blockset*, `http://www.u-dynamics.com/aerosim/`, 2005.

[167] I.A. McManus, D.G. Greer, and R.A. Walker, "Uav avionics "hardware in the loop" simulator," in *Proceedings 10th Australian International Aerospace Congress*, Brisbane, Australia, July 29 – August 1 2003.

[168] Laminar Research, *X-Plane, by Austin Meyer*, `http://www.x-plane.com/`, 2006.

[169] J. Osborne and R. Rysdyk, "Waypoint guidance for small uavs in wind," in *Proceedings of Infotech@Aerospace 2005*, Arlington (VA), USA, 26-29 September 2005.

[170] L.R. Cork, R. Walker, and S. Dunn, "Fault detection, identification and accommodation techniques for unmanned airborne vehicle," in *Proceedings of the 11th Australian International Aerospace Congress*, Melbourne, Australia, 13–17 March 2005.

[171] H. Abbot, B. Bradbury, M. Connor, C. Hawthorne, S. Henson, A. Mitchell, and B. Taylor, "Development of an autonomous aerial reconnaissance system," Association for Unmanned Vehicle Systems International (AUVSI), International Aerial Robotics Competition 2005, 2005.

[172] O. Antoshko, D. Au, S. Dam, J. Desai, V. Kakkar, A. Miliauskaite, and E. Niyonkuru, "Development of an unmanned aerial vehicle," Association for Unmanned Vehicle Systems International (AUVSI), International Aerial Robotics Competition 2005, 2005.

[173] G.D. Chandler, D.K. Jackson, A.W. Groves, O.A. Rawashdeh, N.A. Rawashdeh, W.T. Smith, J.D. Jacob, and J.E. Lumpp, "A low-cost control system for a high-altitude uav," in *IEEE Aerospace Conference 2005*, Big Sky (MT), USA, 5-12 March 2005.

[174] J.S. Berndt, *JSBSim: Open Source Flight Dynamics Model in C++*, `http://jsbsim.sourceforge.net/`, 2005.

[175] E.F. Sorton and S. Hammaker, "Simulated flight testing of an autonomous unmanned aerial vehicle using flightgear," in *Proceedings of Infotech@Aerospace 2005*, Arlington (VA), USA, 26-29 September 2005.

[176] C.L. Olsen, *FlightGear Flight Simulator*, `http://jsbsim.sourceforge.net/`, 2006.

[177] V. R. Puttige and S. G. Anavatti, "Real-time neural network based online identification technique for a uav platform," in *International Conference on Computational Intelligence for Modelling Control and Automation CIMCA2006*, Sydney, Australia, 29 November – 1 December 2006, p. 92.

[178] J.S. Berndt, *JSBSim Aeromatic*, `http://jsbsim.sourceforge.net/aeromatic2.html`, 2008.

[179] M. Hepperle, *JavaProp Java Applet*, `http://www.mh-aerotools.de/airfoils/javaprop.htm`, 2008.

[180] N. Bowditch, *American Practical Navigator*, vol. II, Topographic Centre, United States Defence Mapping Agency, 1975.

[181] Freescale Semiconductor Inc., *MPX4115A Datasheet*, revision 4 edition, 2001.

[182] Freescale Semiconductor Inc., *MPXV5004G Datasheet*, revision 5 edition, 2002.

[183] Portland State Aerospace Society, *A Quick Derivation relating altitude to air pressure*, `http://psas.pdx.edu/RocketScience/PressureAltitude_Derived.pdf`, version 1.03 edition, December 2004.

[184] Trimble, Sunnyvale, CA, USA, *Lassen(tm) SQ GPS Receiver: System Designer Reference Manual*, part number 47830-00, revision a edition, June 2002.

[185] Mono, *Mono Project*, `http://www.mono-project.com/`, 2008.

[186] Civil Aviation Authority, *Civil Aviation Authority Home Page*, `http://www.caa.co.uk/`, 2008.

[187] M. de la Maza and D. Yuret, "Dynamic hill climbing," *AI Expert*, vol. 9, no. 2, pp. 26, 1994.

[188] D. Yuret, "From genetic algorithms to efficient optimization," A. I. Technical Report 1569, Massachusetts Institute of Technology, May 1994.

[189] W.H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, second edition, 1993.

[190] P.J. Acklam, *An algorithm for computing the inverse normal cumulative distribution function*, `http://home.online.no/~pjacklam/notes/invnorm/`, 2008.