MODULAR AUTOPILOT DESIGN AND DEVELOPMENT FEATURING BAYESIAN NON-PARAMETRIC ADAPTIVE CONTROL

By

Jacob Stockton

B.S. Mechanical Engineering
B.S. Aerospace Engineering
Oklahoma State University
Stillwater, OK
2011

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
Master of Science
December, 2014

MODULAR AUTOPILOT DESIGN AND DEVELOPMENT
FEATURING BAYESIAN NON-PARAMETRIC
ADAPTIVE CONTROL

Thesis Approved:

Dr. Girish Chowdhary

Thesis Advisor

Dr. Jamey Jacob

Committee Member

Dr. Andy Arena

Committee Member

ACKNOWLEDGMENTS [1]

First and foremost, I would like to thank my wife Christine Stockton. Without her support, I could not have spent the countless hours required to complete my degree. Christine, I am incredibly grateful for your love and support during this busy season of life.

I would also like to thank my adviser, Dr. Girish Chowdhary, and my committee members, Dr. Jamey Jacob and Dr. Andy Arena. Without your guidance, patience and time, I would not be where I am today. Each of their abilities complement each other, and the result has been some amazing counseling, both technical and practical. I admire you all, and I hope that you continue to put your full efforts into teaching and research.

Lastly, I would also like to thank all of the members of the Distributed Autonomous Systems Laboratory, and specifically Ben Reish. Most of what I have learned has been due to the interactions and discussions with lab members. I want to thank all of the electrical engineering undergraduates that I have worked with. Without them, I could not have constructed the majority of the autopilot.

[1]Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: Jacob Stockton

Date of Degree: December, 2014

Title of Study: MODULAR AUTOPILOT DESIGN AND DEVELOPMENT FEATURING BAYESIAN NON-PARAMETRIC ADAPTIVE CONTROL

Major Field: Aerospace Engineering

Abstract:

Over the last few decades, Unmanned Aircraft Systems, or UAS, have become a critical part of the defense of our nation and the growth of the aerospace sector. UAS have a great potential for the agricultural industry, first response, and ecological monitoring. However, the wide range of applications require many mission-specific vehicle platforms. These platforms must operate reliably in a range of environments, and in presence of significant uncertainties. The accepted practice for enabling autonomously flying UAS today relies on extensive manual tuning of the UAS autopilot parameters, or time consuming approximate modeling of the dynamics of the UAS. These methods may lead to overly conservative controllers or excessive development times. A comprehensive approach to the development of an adaptive, airframe-independent controller is presented. The control algorithm leverages a non-parametric, Bayesian approach to adaptation, and is used as a cornerstone for the development of a new modular autopilot. Promising simulation results are presented for the adaptive controller, as well as, flight test results for the modular autopilot.

Contents

Chapter Page

List of Tables

List of Figures

Figure Page

Chapter 1

Introduction

1.1 Motivation

Over the last few decades, Unmanned Aircraft Systems, or UAS, have become a critical part of the defense of our nation and the growth of the aerospace sector. UAS have already demonstrated a positive impact in many industries such as agriculture, first response, and ecological monitoring. Recently, there has been an increasing push industry-wide for UAS platforms to perform novel tasks such as Short Take-Off and Landing, or STOL, deep stall landings, or other acrobatic maneuvers. Of course, these novel tasks cannot be completed solely with innovative vehicle design, rather a more holistic approach is required. The ability to develop novel control systems that can perform such tasks is highly limited by the computational abilities of the autopilot system on board the UAS. In general, commercial-off-the-shelf (COTS) autopilots are split between between two categories: open-source autopilots and closed-source autopilots. The latter feature low-quality hardware and unreliable software, but a low price point; whereas, the former are extremely reliable, but highly proprietary, relatively expensive, and limited in their capability to perform novel tasks. These limitations clearly restrict the ability for researchers to push the boundaries of higher functionality for UAS.

The wide range of applications of UAS mentioned above has resulted in countless mission specific Unmanned Aerospace Vehicle, or UAV, platforms. These platforms must operate reliably in a range of environments, and in presence of significant uncertainties. The accepted practice for enabling autonomously flying UAVs today relies on extensive manual tuning of the UAV autopilot parameters or time consuming approximate modeling of the dynamics of the UAV. In practice, these methods usually lead to overly conservative controllers or excessive development times. Furthermore, controllers cannot be simply transferred from one UAV to another, rather each platform must be tuned independently of the others in order to acheive the desired performance criteria. This process can be extremely costly and time consuming for companies.

To solve these problems, this thesis posits the use of adaptive control to provide an airframe-independent control algorithm. The problem is framed using past works in adaptive control and Rapid Controller Transfer (RTC). However, RCT has not been realized on fixed wing UAV platforms in the outdoor environment. The primary goal of RCT is to transfer autopilot hardware with negligible effects on the controller performance from a source system, whose dynamics are well-known, to a transfer system, whose dynamics are poorly understood. A practical example of this could be transferring an autopilot from an Aerosonde airframe, the well-known source system, to a Zaggi airframe, the unknown transfer system, as demonstrated in Chapter 2.

The primary advantage of RCT is the significant reduction in time and cost of developing a model for the poorly understood system. The algorithm overcomes the need of laboriously tuning traditional PID controller parameters. The proposed method is an alternative, adaptive control method derived from a new class of data driven adaptive control algorithms. This control algorithm leverages a nonparametric, Bayesian approach to adaptation, and is used as a cornerstone for the development of a airframe-independent autopilot. The limitations of existing autopilot platforms listed above was one of the primary concerns of the author. Thus, this thesis also presents the design and evaluation of a new, open-source autopilot named, Stabilis, to address the standing limitations of current COTS autopilots.

1.2 Literature Review

This research focuses on a platform independent autopilot by means of adaptive control. The following section has been provided in order to present a thorough understanding and a complete history of both adaptive control as it relates to this thesis, as well as, an understanding of existing autopilot technology. The problem of using adaptive control techniques for control transfer has not been extensively studied. However, adaptive control has proved to be a notable solution to both modeling error and system uncertainty. Adaptive control has loosely been classified into two categories direct and indirect adaptive controllers. The former uses the tracking error to directly modify controller parameters or gains; whereas, the latter approximates the difference between the assumed and actual system dynamics, then uses the approximation to control the plant.

Model reference adaptive control, or MRAC, has become a well established direct adaptive control method among the academic community [2–4]. Classical direct MRAC formulations have proven to be applicable to many different systems [5–7], and most relevant to this thesis, flight control systems [8–10]. Classical MRAC methods force the plant output to follow a reference model using weight update laws to reduce tracking error over time. These weight update laws generally rely on Lyapunov stability criteria to guarantee convergence of the system plant to the reference model. Although direct MRAC has proven to be effective on many systems, there are some undesirable effects for certain types of systems. In order to converge to the ideal weight values, there is a need for persistent excitation, or PE, of the system. In addition, such algorithms have been proven to be susceptible to sensor noise, and possess a lack of robustness (see [6, 11] and the references therein).

Since the inception of classical MRAC, there have been many MRAC formulations that have sought to solve some of the issues that are associated with such methods. L1 adaptive control is a well known MRAC formulation that has been widely used in aerospace guidance and control applications [12,13], as well as others [14,15]. Authors claim that the benefits of L1 adaptive control are threefold: fast and robust adaptation, analytically computable performance bounds and excellent performance with minimal flight control design cost [16]. The formulation differs from classical MRAC methods through the use of high adaptive gains with an input filter. The high adaptive gains help ensure the adaptive controller is responsive enough to track the uncertainty point wise in time. Another direct MRAC formulation known as

Intelligent Excitation, seeks to mitigate the need to inject PE in the reference input while guaranteeing parameter convergence [17, 18]. This is done by injecting excitation only when the tracking error exceeds a desirable limit. Although this MRAC formulation reduced the need for excitation, PE is still used, thus control effort is ultimately wasted. A more recent direct MRAC formulation called Derivative Free MRAC, or DF-MRAC, was presented by Yucelen et al. [19]. DF-MRAC relaxes the assumption of constant ideal weights that classical MRAC methods , and thus, features a time varying set of weight parameters. This feature of the algorithm allows for a time varying system to be modeled in the face of uncertainty. In [20], the DF-MRAC formulation is shown to be uniformly ultimately bounded, and the error is shown to be ultimately bounded exponentially. Authors present evidence for dramatic improvement in robustness and superior performance over conventional adaptive laws.

All direct MRAC methods employ a reactive approach, in that, each favors instantaneous adaption through point wise uncertainty suppression to learn the underlying modeling error. Indirect methods, on the other hand, try to learn the underlying dynamical uncertainty using regression techniques. This provides a particular advantage that direct methods do not. Since direct adaptive control methods are not focused on estimating the uncertainty itself, these types of controller can suffer from "Short Term Learning". Essentially, tracking performance does not necessarily improve over time when identical commands are repeatedly tracked [21].

The first, and most widely used, technique for estimating the system uncertainty in the context of indirect MRAC methods is undoubtedly the neural network. From the early 1990s to present neural networks have fascinated and spurred on research within the control community [22–26]. Neural networks used in conjunction with adaptive control techniques are used extensively in flight control and guidance. This formulation guarantees the existence of a set of ideal weights that guarantees optimal approximation of uncertainty, which is implied by the universal approximation property of neural networks.

There are primarily two types of neural networks that are used in adaptive control: single layer hidden (SHL) neural networks and radial basis function (RBF) neural networks. The idea of RCT is first presented using a neural network based MRAC formulation [27], but RCT was not explicitly studied. Later, Chowdhary et al. extended neural networks into a formulation of MRAC that uses both recorded and instantaneous data to "concurrently learn", and was thus called Concurrent Learning MRAC, or CL-MRAC. The most notable feature of CL-MRAC is its ability to leverage the advantages of both direct and indirect adaptive control to mitigate the need for PE [28]. Recently, CL-MRAC was used for RCT on indoor quadcopters with promising results [29].

However, both SHL and RBF neural networks have disadvantages. One of the more notable disadvantages of RBF neural network based approaches, is that the number of centers and hyperparameters must be expertly allocated a-priori over the operating domain. Thus controllers operating outside of the intended domain experience degraded performance [30, 31]. On the other hand, SHL neural networks performance can suffer from getting stuck in local minimum, provided deficient momentum [32].

Unlike RBF Neural Networks, Gaussian Processes, or GPs, can cover the entire operating domain by dynamically allocating kernel locations based on a fixed budget of kernels. Furthermore, since GPs are Bayesian in nature, the model itself provides a quantified confidence metric in its predictions via the predictive variance. Previously, the prospect of using online GPs to model uncertainty was computationally intractable due to large data sets. However, largely due to the derivation of sparse, online Gaussian Processes by Casato et al. [33], GPs were recently proposed as a nonparametric approach to modeling dynamical uncertainty in an adaptive controller [31]. Furthermore, Grande et al. proved that the hyperparameters associated with the kernels can be optimized online as well [34]. The flight test results presented in this research show GP-MRAC outperforms modern MRAC methods that use NN by a factor of two to three times. However, the promising results of these works have yet to be implemented in the outdoor environment, nor on fixed wing platforms.

1.3 Outline of Contributions

The main contributions of this thesis are twofold:

- A control architecture is defined that extends GP-MRAC to fixed wing flight in order to perform Rapid Controller Transfer (RCT).
- An open architecture autopilot is designed, constructed and evaluated. This autopilot design differs from past designs in its modularity and superior computational performance.

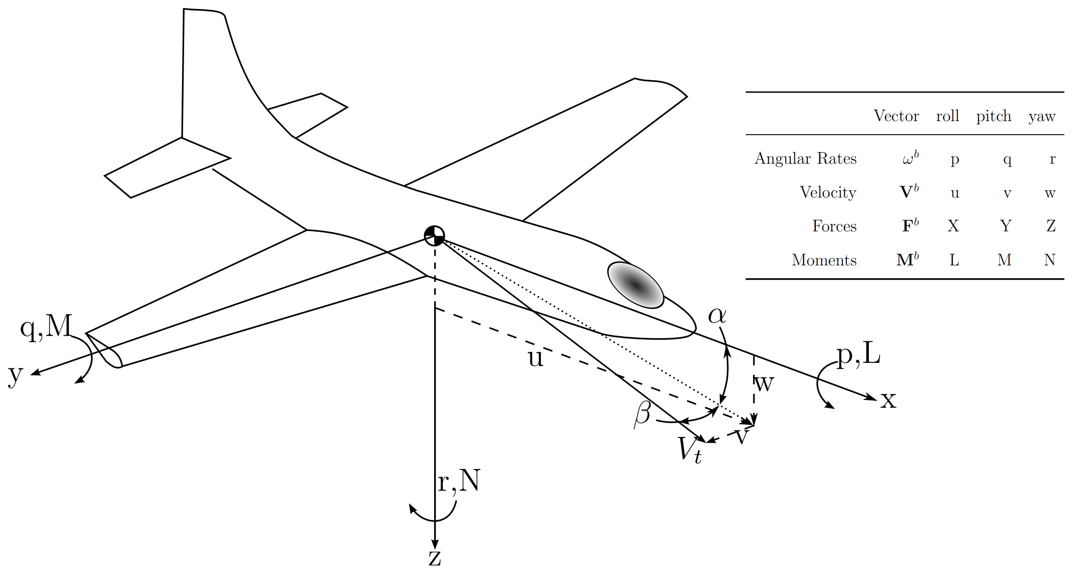
The thesis outline is as follows. The contributions of this thesis are primarily distributed between Chapters 2 and 3. The problem of RCT as it pertains to fixed wing aircraft dynamics is formulated in Chapter 2. A brief overview of nonlinear flight dynamics is followed by an exposition of industry practice pertaining to UAS automatic control systems. A full nonlinear control scheme for longitudinal motion is implemented using the GP-MRAC formulation discussed above. Benchmarking, design and construction of a modular, open-source autopilot, which we term Stabilis, is presented in Chapter 3. Due to the potential risk of autonomous UAV operations, thorough testing before flight is paramount. To this end, Stabilis is thoroughly tested using hardware in the loop simulations and ground testing procedures. Finally, flight test results are presented to validate the predicted performance of Stabilis. Future work and concluding remarks are in Chapter 4.

Chapter 2

Bayesian Nonparametric Adaptive Control for Fixed-Wing Flight

2.1 Introduction

In order to control a system, an in-depth understanding of the system dynamics and its operating conditions must be possessed. In the case of this thesis, Rapid Controller Transfer causes uncertainty within the context of fixed wing flight dynamics. Thus, the notation and equations used to describe nonlinear, rigid body flight are briefly covered first. Interested readers are referred to [35–38] for more information regarding coordinate frames, flight dynamics and control. Since this work departs from traditional adaptive control formulations in its use of Gaussian Processes, a brief overview of GPs is provided.



	Vector	roll	pitch	yaw
Angular Rates	ω^b	p	q	r
Velocity	$\mathbf{V}^b$	u	v	w
Forces	$\mathbf{F}^b$	X	Y	Z
Moments	$\mathbf{M}^b$	L	M	N

Figure 2.1: Body Fixed Frame with aerodynamic angles

2.1.1 Aircraft Kinematics and Dynamics

Consider an aircraft, as shown in Figure 2.1, with a mass moment of inertia, $\mathbf{I}^b$, and mass, m. The mass moment of inertia is aligned with the body fixed frame denoted with the superscript, $(\cdot)^b$. Note the x-axis of the body fixed frame points out the nose of the aircraft, the y-axis is directed out of the starboard wing of the aircraft, and the z-axis is oriented downward, normal to the x and y axes. The origin of the body fixed frame is the aircraft center of mass.

Let the position of the aircraft with respect to the origin of the body fixed frame be described using the navigational frame denoted with the superscript, $(\cdot)^n$. The attitude of the vehicle is described using Euler angles defined, $\mathbf{EA} = [\ \phi \quad \theta \quad \psi\ ]$. The translational kinematics of the aircraft in the navigational frame are related to the body fixed frame by Euler angles

$$\dot{\mathbf{p}}^n = \frac{d}{dt}\left(\mathbf{R}_b^n \mathbf{p}^b\right) \quad = \quad \mathbf{R}_b^n \mathbf{V}^b \quad ; \tag{2.1}$$

where, $\dot{\mathbf{p}}^n = [\ \dot{p}^n \quad \dot{p}^e \quad \dot{p}^d\ ]^T$ and,

$$\mathbf{R_b^n} = \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi + S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} . \tag{2.2}$$

Note that in equations 2.2 and 2.3, $S_\theta = \sin\theta, C_\phi = \cos\phi$, and so on. The relation of the body fixed angular rates to inertial frame angular rates is

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S_\phi \tan_\theta & C_\phi \tan\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi \sec\theta & C_\phi \sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.3}$$

The relationships for the translational and rotational kinematics above can be used to express Newton's second law in the navigational frame. The equations of translational and angular dynamics of a 6 degree-of-freedom rigid body are given by

$$\sum_i \mathbf{F}_i = m\ddot{\mathbf{p}}^n = \mathbf{g}^n + \mathbf{R}_b^n m\mathbf{a}^b \tag{2.4}$$

$$\dot{\omega}^b = \left(\mathbf{I}^b\right)^{-1}\left(\mathbf{M}^b - \omega^b \times \mathbf{I}^b\omega^b\right); \tag{2.5}$$

where, $\mathbf{g}^n = [\ 0 \quad 0 \quad g\ ]^T$ is the navigational frame gravity vector, and $\mathbf{a}^b = [\ \dot{u} \quad \dot{v} \quad \dot{w}\ ]^T$ is the body fixed accelerations. Equations (2.4) and (2.5) above can be expanded and written in terms of the body frame as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{pmatrix} -g\sin\theta \\ -g\sin\phi\cos\theta \\ -g\cos\phi\cos\theta \end{pmatrix} + \frac{1}{m}\left[\begin{pmatrix} F_T \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}\right] - \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \left(\mathbf{I^b}\right)^{-1} \left(\begin{bmatrix} L \\ M \\ N \end{bmatrix}^b - \begin{bmatrix} p \\ q \\ r \end{bmatrix}^b \times \mathbf{I^b} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \quad ; \tag{2.7}$$

where, F_b and M_b are given by the aerodynamic forces on the aircraft. The aerodynamic forces are primarily dependent on the angle of attack, α, and side slip, β, in steady states. However, the body fixed angular rates can significantly change the aerodynamc forces as shown in equations 2.8 and 2.9.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} C_X(\alpha,\beta) \\ C_Y(\beta) \\ C_Z(\alpha) \end{pmatrix} QS \tag{2.8}$$

$$\begin{pmatrix} L \\ M \\ N \end{pmatrix} = \begin{pmatrix} C_L(\delta_a,\beta,\tilde{p},\tilde{r})QSb \\ C_M(\delta_e,\alpha,\tilde{q})QS\bar{c} \\ C_N(\delta_r,\beta,\tilde{r}) \end{pmatrix} \quad ; \tag{2.9}$$

where, $\tilde{p} = \dfrac{bp}{2V_t}, \tilde{q} = \dfrac{\bar{c}q}{2V_t}, \tilde{r} = \dfrac{br}{2V_t}$. Since body fixed forces and moments are functions of multiple variables, they are the most complex part of the aircraft to be modeled. Usually, linear approximations are used for aerodynamics forces; readers are referred to the references in the introduction for full linear state models.

2.1.2 Gaussian Processes

A Gaussian process is a supervised learning technique, which is the problem of mapping an input to a corresponding output given a set of data. The applications for supervised learning are practically endless. All supervised learning techniques utilize a set of training data, $\mathcal{D}$, which is usually a set of observations or emprical data, defined, $\mathcal{D} = \{(x_i, y_i)|i = 1, \ldots, n\}$. Supervised learning techniques are inductive in nature, in that, the objective is to make predictions for an input, say x^*, that is not included in the training set. In order to make an accurate prediction about an input that is not included in our data set, there must be some kind of assumption about the underlying function. Supervised learning techniques generally correspond to inference utilizing either parametric or nonparametric models; in the first case, the structure of the model or predictor is assumed to be known (for instance, it is assumed to be linear, quadratic, etc in the input), and the parameters associated to the model are inferred from the data. In the second case, the structure of the predictor is inferred from the data itself, which makes the predictor more flexible, although more expensive to compute. GP regression is an example of the second class of techniques.

GPs are in the class of Bayesian nonparametric methods. In a GP, the prior is placed on a function space, specifically, on functions contained in a Reproducing Kernel Hilbert Space, or RKHS. Here the prior encodes the "prior belief" of what class of functions the predictor belongs to. The actual, unknown function is a point in the RKHS. Consider the case of RCT, where there exists significant uncertainty between aircraft dynamics. It is desirable to predict the dynamical uncertainty using a set of discretely sampled state measurements, $Z_t = z_1, \ldots, z_t$; where, t is the current measured state, and there exists an inherent extent of noise for all $z \in Z$. Let the uncertainty, which will be furthered defined in Section 2.3.1, be denoted as, Δ; where, $\Delta(\cdot) \in \mathbb{R}$ for ease of exposition. When modeled using a GP,

$$\Delta(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)); \tag{2.10}$$

where $m(\cdot)$ is the mean function, and $k(\cdot, \cdot)$ is a real-valued, positive definite covariance kernel function. The covariance kernel function operates on Z such that a covariance matrix is defined by the indexed sets, $K_{i,j} = k(z_i, z_j)$. The most popular choice of kernel matrix, is the Gaussian radial basis function,

$$k(z, z') = exp\left(\frac{\|z - z'\|^2}{2\mu^2}\right). \tag{2.11}$$

It is assumed that the GP prior has a zero mean, that is, $\Delta(z_i) = m(z_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \omega_n^2)$. The posterior is not restricted to zero mean. Given a new measured state value, z_{t+1}, the joint distribution of the data under the prior distribution is

$$\begin{bmatrix} y_t \\ y_{t+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(Z_t, Z_t) + \omega^2 I & k(Z_t, z_{t+1}) \\ (k(Z_t, z_{t+1}))^T & k(z_{t+1}, z_{t+1}) \end{bmatrix}\right). \tag{2.12}$$

The posterior distribution is obtained using Bayes law, and by conditioning the joint Gaussian prior distribution over the observation, z_{t+1}

$$p(y_{t+1}|Z_t, z_{t+1}) \sim \mathcal{N}(\hat{m}_{t+1}, \hat{\Sigma}_{t+1}); \tag{2.13}$$

where, the mean and the covariance are respectively estimated by

$$\hat{m}_{t+1} = \left[(K(Z_t, Z_t) + \omega^2 I)^{-1} y_t\right]^T K(z_{t+1}, Z_t) \tag{2.14}$$

$$\hat{\Sigma}_{t+1} = k(z_{t+1}, z_{t+1}) - K(z_{t+1}, Z_t)^T (K(Z_{tau}, Z_t) + \omega^2 I)^{-1} K(z_{t+1}, Z_t) \tag{2.15}$$

RCT requires the prediction of uncertainty be done online, as the controller has no foreknowledge of the uncertainty that exists between the aircraft. However, the measurement vector Z_t and observation set grow quickly over time. This makes the computation of equations 2.14 and 2.15 intractable over time. Clearly, a modification to traditional GP regression has to exist to make GP regressions possible for MRAC.

2.1.3 Online Learning using Gaussian Processes

As aforementioned, calculations associated with traditional GP regressions quickly become intractable for larger data sets, since the method scales as $O(n^3)$, where n is the number of data points. Csato developed a method that utilizes rank 1 updates for the weight vector α and covariance. Additionally, a budget can be implemented to retain a limited number of measurement values in Z_t. This subset of available measurements used to is called the *basis vector set*, $\mathcal{BV}$. The implementation of rank 1 updates while budgeting $\mathcal{BV}$ allows for real time uncertainty modeling. Not every measurement is useful for prediction. Thus, we work to restrict $\mathcal{BV}$ using a linear independence test, given by equation 2.16 to determine the novelty of incoming data.

$$\gamma_{t+1} = K(z_{t+1}, Z_t) - k(z_{t+1}, z_{t+1})\alpha_t \tag{2.16}$$

where, γ_{t+1} is a scalar. There are two schemes used to determine which point in

$\mathcal{BV}$ is retained, oldest point method, **OP**, and KL divergence method, **KL** [31]. For **OP**, provided that γ_{t+1} is greater than some user specified tolerance, the data point is retained in $\mathcal{BV}$ and the oldest point is discarded. Although this method is less computationally intensive, the retention of a new measurement in $\mathcal{BV}$ may come at the cost of discarding a more useful measurement for prediction. Thus, researchers have found that the **KL** method performs significantly better [31,34] in the context of flight controllers. The KL divergence method employs Csato's sparse online Gaussian process to efficiently approximate the KL divergence . For the specifics concerning this method, readers are referred to [33].

2.2 Aircraft Guidance

Although there has been extensive research in trajectory tracking [16, 39–41], most real world applications involve navigating between, or orbiting around, waypoints. Furthermore, time-parameterized trajectories are typically not robust due to environmental interactions and physical limitations of the transfer system. Thus, this work utilizes waypoint-based guidance methods. In practice, waypoints are provided to the aircraft as geodetic coordinates through the ground station user interface. Guidance methods in the following sections utilize the navigational frame; conversion from geodetic coordinates to the navigational frame can be found in [38]. Consider the problem of navigating through n waypoints in an environment without obstacles. Let the waypoints be given in the navigational frame as, $\mathbf{WP}_i^n$, where $i \in \mathbb{N}$ and i represents the current waypoint. Furthermore, let the aircraft position be $\mathbf{p}^n$, the desired path, $\mathbf{q}^n$, and the UAS location relative to the current waypoint, $\mathbf{r}^n$. Hence,

$$\mathbf{q}^n = \mathbf{WP}^n_{i+1} - \mathbf{WP}^n_i \tag{2.17}$$

$$\mathbf{u}^n = \mathbf{p}^n - \mathbf{WP}^n_i . \tag{2.18}$$

Then, the path tracking error can be found by taking the vector rejection of the actual and desired path vectors by

$$\mathbf{e}_p = \begin{pmatrix} e_{p_x} \\ e_{p_y} \\ e_{p_z} \end{pmatrix} = u^n - (\mathbf{u}^n \cdot \hat{\mathbf{r}})\,\hat{\mathbf{r}} \ ; \tag{2.19}$$

where, $\hat{\mathbf{r}} = \dfrac{r^n}{\|\mathbf{r^n}\|}$.

In order to follow the path, the UAS must minimize the error in the lateral direction known as crosstrack error, $e_{p_{xy}}$, as well as the longitudinal error dictated by the altitude of the aircraft, e_{p_z}. The sign of the crosstrack error is determined by the angle given by

$$\mathcal{A}(\angle_1, \angle_2) := \{\angle_1 - \angle_2 + 2\pi n \mid n \in \mathbb{I} \ni |\mathcal{A}(\angle_1, \angle_2)| \leq \pi\} \ ; \tag{2.20}$$

where, $\angle_1 = \angle \mathbf{r}_{xy}$ and $\angle_2 = \angle \mathbf{p}_{xy}$. The magnitude of the crosstrack error is determined by the north and east elements of the vector e_p. Thus,

$$e_{p_{xy}} = sign(\mathcal{A}(\angle \mathbf{r}_{xy}, \angle \mathbf{p}_{xy}))\,\|e_p(1,2)\| . \tag{2.21}$$

The guidance method must also encompass some type of switching mechanism to advance the directive of the aircraft to the next waypoint provided the current waypoint is reached. The simplest and most common is the distance method [35,38, 42], which states that the waypoint will be switched when the distance between the desired waypoint and the aircraft is less than the tolerance, ϵ. Thus, the waypoint is switched provided that

$$\left\|\mathbf{p}^n - WP_{i+1}^n\right\| \leq \epsilon \; ; \tag{2.22}$$

Alternatively, the waypoint can be switched when the UAV enters the half plane between the segments a and b; where, $a = WP_i^n - WP_{i-1}^n$ and $b = WP_{i+1}^n - WP_i^n$. Empirical tests between the methods dictated that the half-plane method yielded better waypoint tracking performance.

2.3 Control Design

The most widely used method in autopilot control design is the successive closure of control loops to achieve a desired inertial position and attitude [35,42–45]. Successive loop closure, in most cases, uses the assumption that the dynamics of the aircraft, both longitudinal and lateral are decoupled. This assumption is widely utilized, and allows for simplification of the autopilot control schemes. In Figure 2.2, the loop closure design is shown. In order to keep the dynamics of each loop sufficiently decoupled, the bandwidth of each loop must be sufficiently smaller as one moves from the outer loop design to the inner loop design. The differences in bandwidth will vary due to the application, but authors have had success with variance by a factor of 5 to 10 between each loop [35].

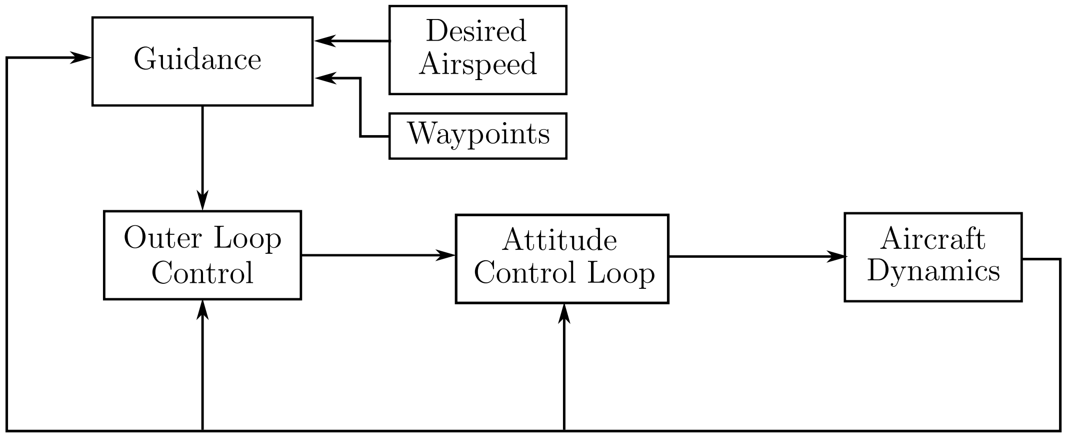


Figure 2.2: Control loop closure

Rather than taking a traditional successive loop closure design shown in Figure 2.2, this implementation takes a more "human based" approach to flight. Essentially, altitude is commanded using the available control input directly, rather than relying on simplifying assumptions required for successive loop closure. We are able to do this largely because the GP can sufficiently model the coupling between inner and outerloop dynamics. Unlike the indoor flight environment, precise position and attitude measurements are not available when using modern MEMS inertial sensors. Thus it is necessary to turn to the following assumption.

Assumption 2.1 *Sufficiently accurate estimates of ϕ, θ and ψ are available for control.*

This assumption can be satisfied by utilizing a good inertial navigation system that fuses together inertial and absolute reference (such as GPS) measurements to reliably estimate attitude [35, 38, 42]. Empirical results in flight test dictated that longitudinal motion proved to be more sensitive to controller parameters. Thus, this thesis demonstrates the viability of GP-MRAC in longitudinal motion. However, a parallel formulation is provided for fixed-wing lateral motion using the crosstrack error as the reference system. This formulation is given in Appendix A for reference.

2.3.1 GP-MRAC in Longitudinal Motion

Interested readers are referred to [31] for a rigorous exposition of proofs for the stability of GP-MRAC that directly relate to this work. The results of which, dictate that GP-MRAC is a exponentially mean square ultimately bounded controller in the context of AMI-MRAC. Furthermore, let $\delta_e(t) \in D_{\delta_e} \subset \mathbb{R}^n$ be bounded for all $t \in \mathbb{R}^+$. The following assumption must hold for both the source and the transfer system:

Assumption 2.2 *For all $\delta_e(t) \in D_{\delta_e}$, there exists a finite value $B > 0$ such that $|q(t)| < B$.*

Since UAS are required to be piloted remotely, the vast majority of fixed-wing UAS are designed to be dynamically and statically stable, and therefore, satisfy this assumption de facto. An inner loop controller must be used to provide baseline stability, provided that Assumption 2.2 does not hold. The dynamics of the source and transfer system must be defined . To this end, let $z = [\begin{array}{ccccc} \alpha & \theta & q & \dot{h}^T & \delta_e \end{array}]^T$. It is assumed that for both systems the outer loop states for the longitudinal direction can be modeled by the following differential equation,

$$\begin{aligned} \dot{h}_1(t) &= h_2(t) \\ \dot{h}_2(t) &= f(z(t)) + b(z(t))\delta_e \end{aligned} \tag{2.23}$$

The function f is assumed to be Lipschitz continuous in $z, \dot{z} \in D_z$, and the systems are assumed to be finite input controllable. This assumption is validated, in part, by rearranging Equation 2.4. The altitude dynamics can be written as

$$\ddot{h} = g - \sin\theta\ \dot{u} + \sin\phi\cos\theta\ \dot{v} + cos\phi\cos\theta\ \dot{w}\ , \tag{2.24}$$

where, $[\ \dot{u} \quad \dot{v} \quad \dot{w}\ ] = [\ X \quad Y \quad Z\ ]/m$, and the body fixed forces are given by 2.8. The following assumption characterizes the controller on the system [29]:

Assumption 2.3 *For the source and transfer system, there exists a control law* $g : D_z \to D_{\delta_e}$ *such that* $\delta_e(t) = g(z, \ddot{h}_{rm})$ *drives* $h \to h_{rm}$ *as* $t \to \infty$. *Furthermore, the control law is invertible w.r.t.* δ_e, *hence the relation* $\ddot{h}_{rm} = g^{-1}(z, \delta_e)$ *holds. [29]*

The primary goal of MRAC based methods is to design a control law such that h converges to h_{rm} satisfactorily. In the case of AMI-MRAC, feedback linearization of the system is achieved by identifying the pseudo-control, $\nu(t) \in \mathbb{R}$, that achieves the desired acceleration. Provided that the system dynamics were known and invertible, the control input could be easily found as $\delta = f^{-1}(\nu, b, t)$. However, in the case of RCT, the plant dynamics are extremely poorly understood. Thus, an approximate model must be used which, in practice, is usually the previous vehicle. The use of an approximate model, $\hat{f}(z) + \hat{b}(z)\delta$, leads to modeling error Δ; where delta is defined

$$\Delta(z) = \dot{h}_2(t) - \nu(z) \tag{2.25}$$

A designer reference model is used to characterize the desired system response. In the case of straight path tracking, the positional input translates to a ramp input.

Standard reference models include second order systems or the second time derivative of a time-parameterized trajectory polynomial. The former results in steady state error, whereas, the latter is not used in this work for reasons discussed above. Thus, a more appropriate reference model selection would be a PID reference model. Then the feedforward term of reference model is given by

$$\ddot{h}_{rm}(t) = K_p e_h + \int_a^b e_h(\tau)\,\mathrm{d}\tau + K_d \dot{e}_h \ ; \tag{2.26}$$

where, $e_h = h - h_{cmd}$ and $\dot{e}_h = \dot{h} - \dot{h}_{rm}$. Here, h_{cmd} is given by the path between waypoints, q, and $\dot{h}_{cmd} = \mathbf{q}_d / \sqrt{\mathbf{q}_n^2 + \mathbf{q}_e^2}\, cos(\chi - \chi_q)\ |V_g|$. The reference model states $\dot{h}_{rm}$ and h_{rm} are given by integrating 2.26 for some initial conditions $\dot{h}_{rm_0}$ and h_{rm_0}. The tracking error is defined as, $e_t = h_{rm} - h$, and the psuedocontrol, ν to be

$$\nu = \nu_{rm} + \nu_{pd} - \nu_q - \nu_{ad}. \tag{2.27}$$

where, $\nu_{rm} = \dot{x}_{2_{rm}}$, $\nu_{pd} = [K_1\ K_2]\, e_t$ and the robustifying term $\nu_q = q$. The adaptive term, ν_{ad} is tasked with canceling the uncertainty $\Delta(z)$. Thus, the existence and uniqueness of a fixedpoint solution to $\nu_{ad} = \Delta(z)$ is assumed [31,34]; results from [46] dictate that the sign of control effectiveness derivative must be the same for both systems, i.e. $sgn\dfrac{\partial g}{\partial \delta_s} = sgn\dfrac{\partial g}{\partial \delta_t}$. It was alluded to in Section 2.1.2 that the uncertainty of the dynamical system could be predicted using a draw from a GP. Thus, ν_{ad} is modeled

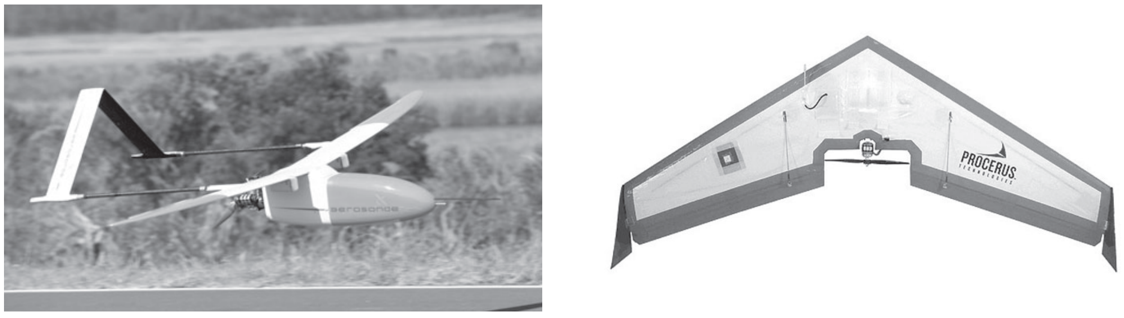
$$\nu_{ad}(z) \sim \mathcal{GP}(\hat{m}(z), k(z, z'))\ . \tag{2.28}$$

Rather than drawing the adaptive element strictly from the distribution in 2.28, the adaptive element is set equal to the GP mean output, i.e. $\nu_{ad} = \hat{m}(z)$, for reasons discussed in [31,34]. The simulation results below required the GP to model the dynamic uncertainty in the outer loop based upon the input vector z, which included both inner and outer loop states. Practically speaking, it can be costly and difficult to measure the aerodynamic angles included in the vector z. Accordingly, simulations were run that excluded the aerodynamic angles from the results. The tracking performance for both cases was remarkably similar, varying only by 1-3% depending on the aircraft. Although this is not entirely intuitive, the performance is largely attributed to the GP's ability to model the coupling between inner and outer loop dynamics.

2.4 Simulation Results

Two aircraft were chosen with largely different dynamics and different configurations: the boom-tailed Aerosonde and the flying wing Zaggi. Viability of the controller for fixed-wing aircraft is shown for the longitudinal direction only. The aerodynamic models for the aircraft were taken from [35]. Due to time constraints, the RCT was implemented solely using a custom MATLAB simulator. The same PD gains were used for both aircraft ($[K_p\ K_d] = [1.2\ .2]$). The GP was preallocated a budget of 25 active bases; where, the tolerance $\epsilon = .0001$ and the RBF kernel bandwidth $\mu = 0.5$.

Each aircraft was flown in the same altitude climb maneuver, tasked with following a given path used as the reference model. The simulations detailed in Figures



(a) (b)

Figure 2.3: (a) Aerosonde UAV. (b) Zaggi UAV

2.4 through 2.7 show each aircraft being flown with a simple PD, feedforward control scheme, and then subsequently, using the GP-MRAC algorithms. In order to perform Rapid Controller Transfer, the marginally tuned controller for the Aerosonde in Figure 2.4 is applied directly to the Zaggi in Figure 2.6. The controller performance on transfer aircraft is degraded significantly. Without modifying the controller parameters, the feedback linearization is applied using the GP to learn the uncertainty that arises from transfer system in Figure 2.7. In Figure 2.8, the performance of the GP to model the uncertainty is shown.

2.4.1 Computation Requirements for GP-MRAC

Practically speaking, all adaptive control algorithms come at the cost of some kind of computational power. The computational requirements of GP-MRAC are primarily influenced by the number of kernels used in GP estimates, which is preallocated for online GP methods. In order to provide an estimate of the computational requirements, the GP-MRAC architecture was translated to C++ and implemented on the on the Flight Control Computer in Section 3.1.1. A subroutine was written that utilized an online GP with **KL** divergence method to regress. Utilizing 25 kernel centers, the script ran at approximately 10Hz. Taking into consideration that the code was running in the context of an operating system and the code was not optimized for run-time, the number of instructions per iteration was approximately, $5 \cdot 10^6$.

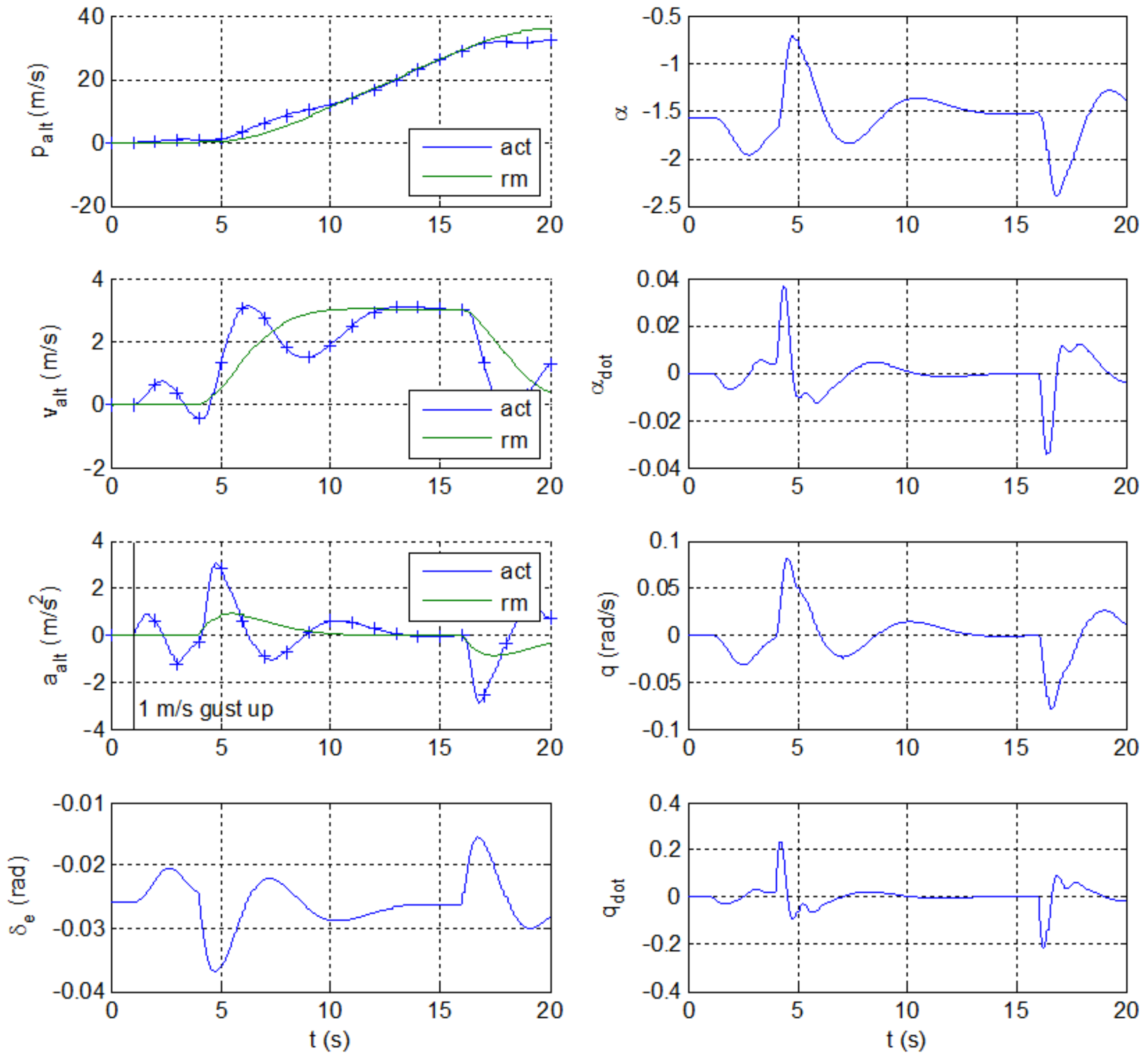


Figure 2.4: Outer loop longitudinal control of Aersonde aircraft without GP-MRAC. The controller performs fair with respect to tracking the desired path, but poorly in other respects. Wind gust disturbance is introduced at t = 1 second.

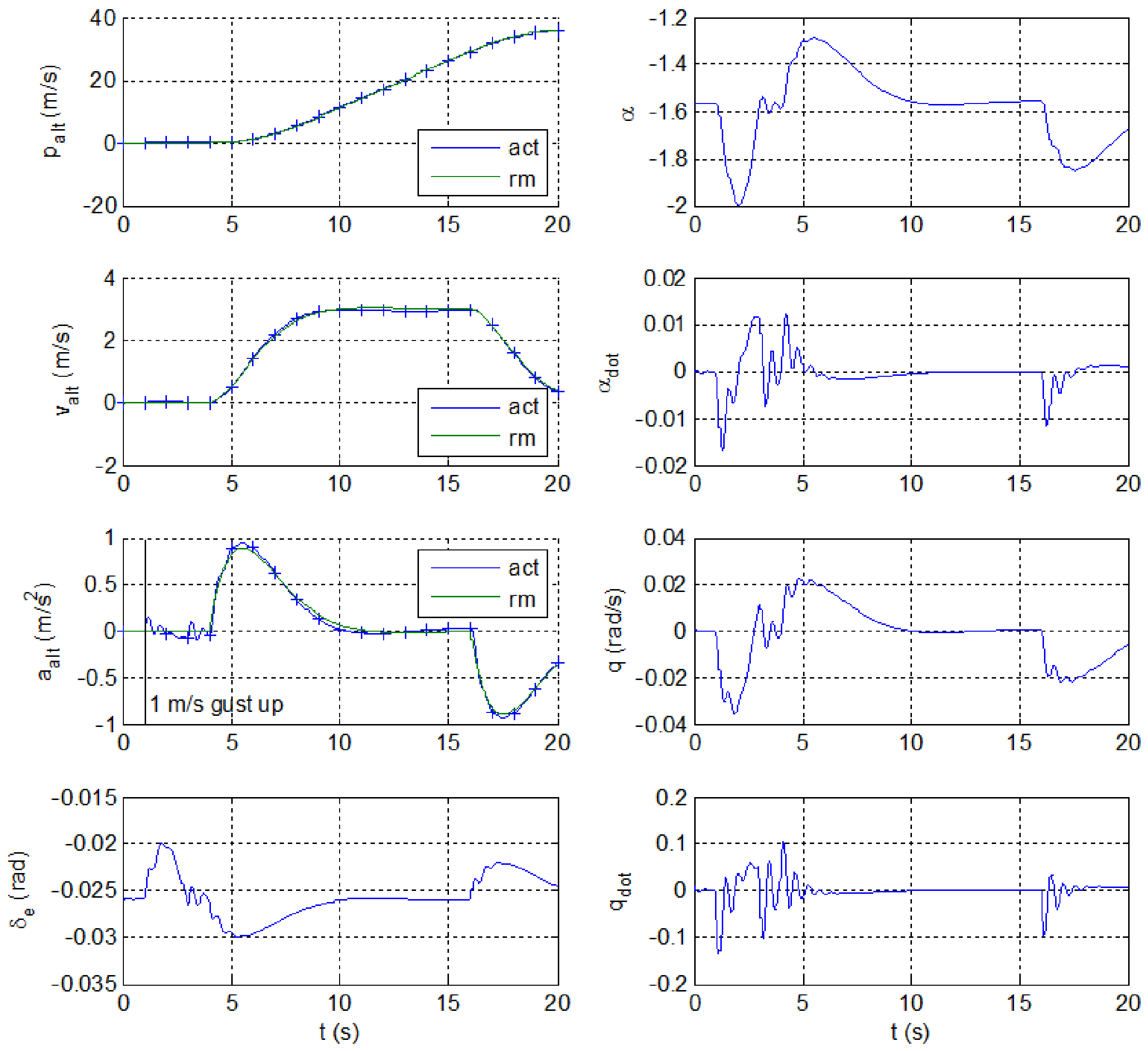


Figure 2.5: Outer loop longitudinal control of Aersonde aircraft with GP-MRAC. Clear, enhanced performance in tracking the path and reference model.Wind gust disturbance is introduced at t = 1 second.

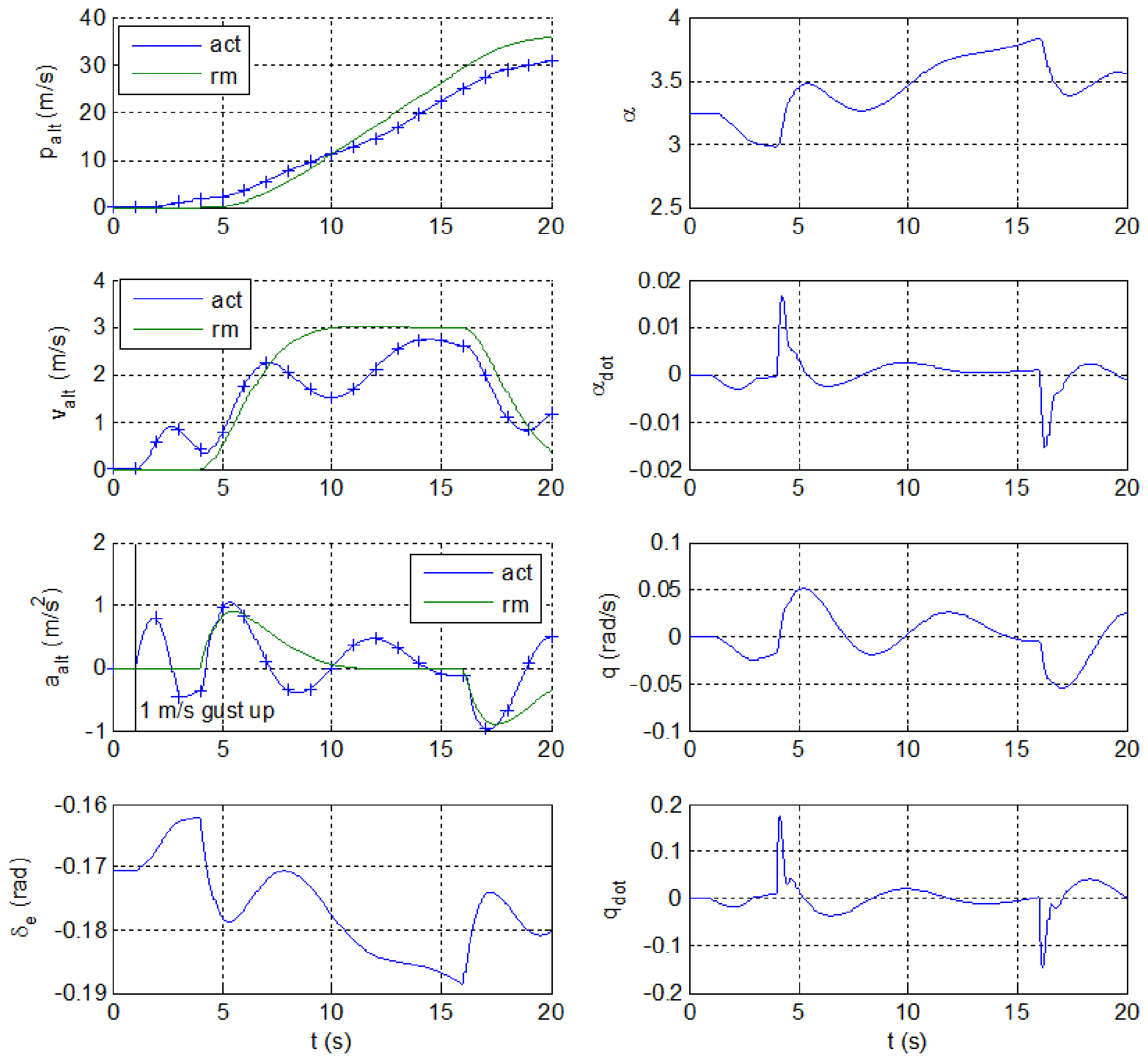


Figure 2.6: Outer loop longitudinal control of Zaggi aircraft using the same gains parameters as the Aerosonde aircraft. Controller performance is clearly degraded.Wind gust disturbance is introduced at t = 1 second.

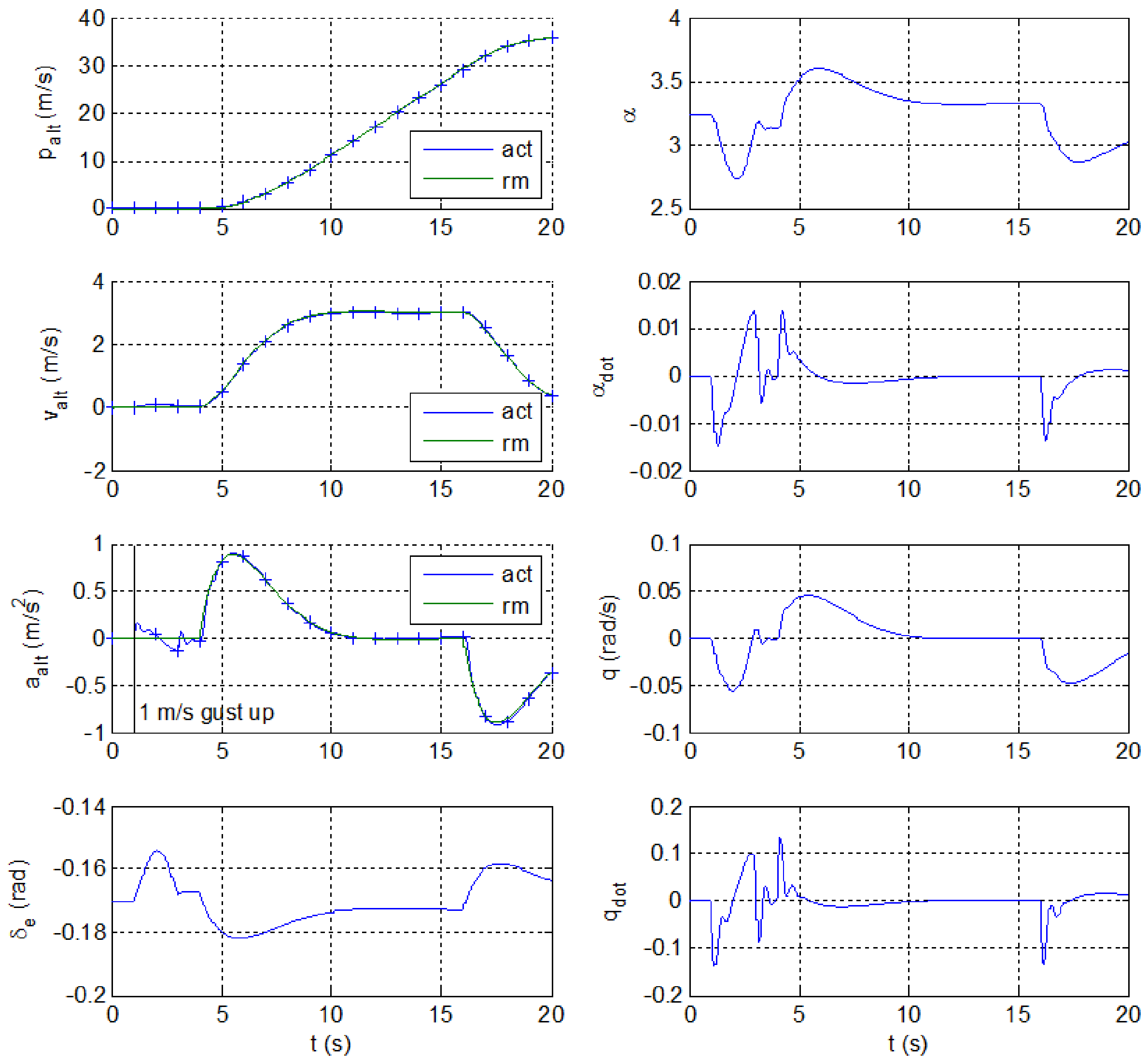


Figure 2.7: Outer loop longitudinal control of Zaggi aircraft with GP-MRAC. Uncertainty between the aircraft dynamics is learned quickly enhancing the controller performance and allowing for the proper path tracking. Wind gust disturbance is introduced at t = 1 second.

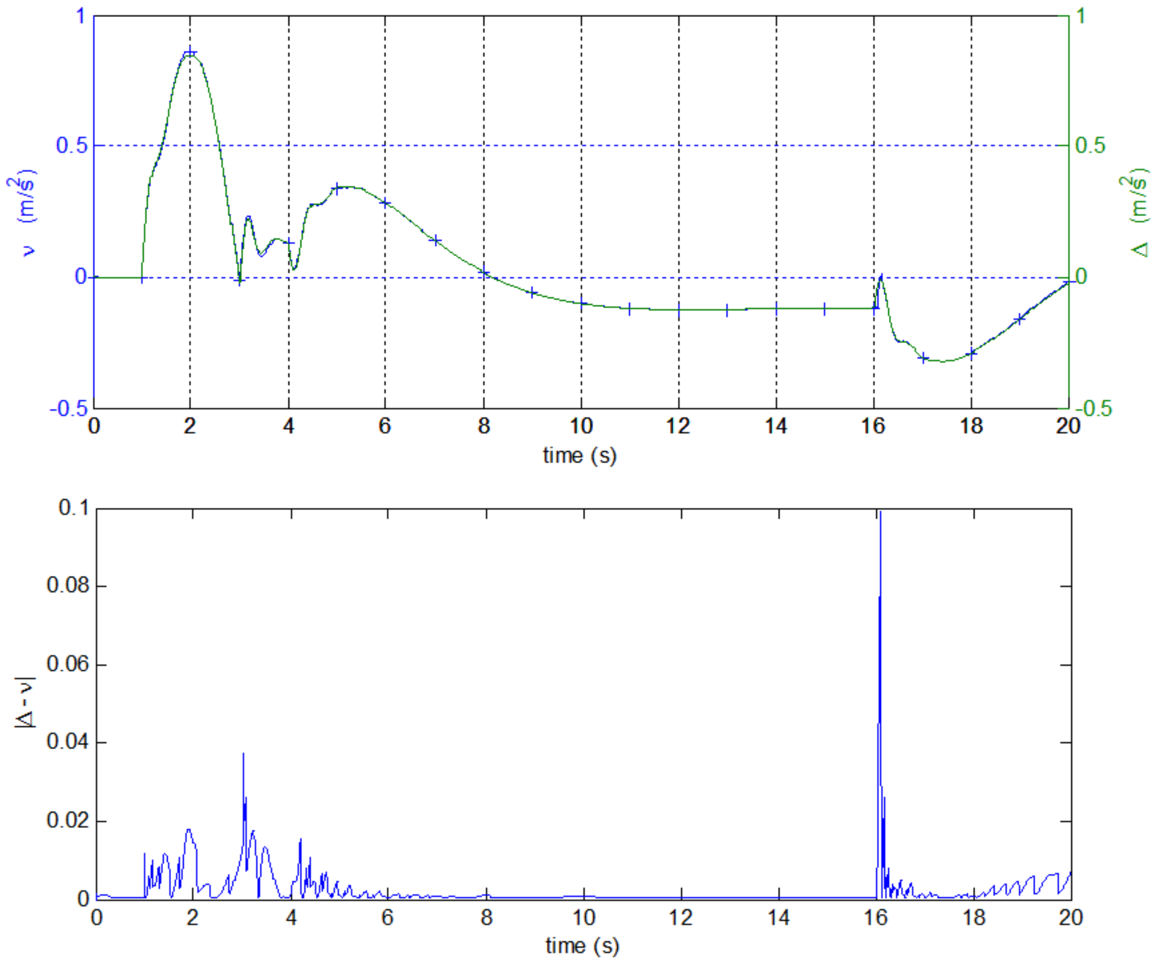


Figure 2.8: GP learning the uncertainty between the aircraft dynamics in order to perform feedback linearization associated with AMI-MRAC architecture.

2.5 Summary

The control laws presented in this chapter posit that a controller can be transferred between fixed-wing aircraft platforms with very little foreknowledge of the system dynamics. The primary benefit would be the potential time and cost savings in developing highly functional UAS. The control law presented fused AMI-MRAC, a well known adaptive control method, with an online supervised learning technique known as a Gaussian Process for adaptation. The formulation involved adaptation in the outer loop, rather than the inner loop. Simulation results presented in this chapter demonstrate the ability of this alternative approach to adapt to new airframe dynamics.

Chapter 3

Autopilot Design & Development

The implementation of control theory is only as robust and powerful as the hardware and software supporting it. This chapter presents the justification, design, construction and assembly of the autopilot hardware and software. There have been many autopilots designed both in the academic and industry settings. However, many of the autopilots developed in years past have become obsolete due to the computational requirements of modern control techniques, and, in part, by Moore's Law. The market survey conducted in [47] is no longer current. Table 3.1 includes a comprehensive SWAP analysis of existing COTS autopilots. Additional tables regarding the sensor and peripheral compatibility of each autopilot are provided in Appendix B. The autopilots featured in Table 3.1 are primarily limited in three ways: computational ability, software access and hardware flexibility. Since most vision-based, machine learning, adaptive control and cooperative control algorithms call for higher computational capabilities, these hardware limitations clearly restrict the ability for researchers to push the boundaries of higher functionality for UAS.

Table 3.1: COTS Autopilot SWAP analysis

	Size (in)	**Wt.(g)**	**Power**	**Cost $**	**CPU**	**Mem.**	**VDC In**
Kestrel 2.2	2.0x1.4x0.5	17	2.5W	5k	29MHz	512Kb	6-16.5
MP 2128g	3.9x1.6x0.6	28	0.9W	5.5k	150MIPS	-	4-26
Piccolo Nano	3x1.5x0.43	32	-	8k	-	-	6-30
Piccolo LT	5.1x2.3x0.7	110	4W	10k	40MHz	448Kb	4.5-28
Piccolo II	5.2x2.5x1.8	220	4W	10k	-	-	8-20
Unav3521	2.0x1.0x0.4	42	.6W	3-5k	40MIPS	256Kb	4-7
osflexPilot[1]	2.4x2.2x1.2	200	-	7.5k	1GHz	512Mb	-
osnanoPilot[1]	2.7x1.2x0.8	32	-	-	-	-	-
osflexQuad[1]	2.7x2.7x0.8	41	-	-	-	-	-
Slugs	2.0x3.0x1.0	-	-	-	70MIPSx2	-	6-12
PixHawk[2]	2.0x3.2x0.7	38	1W	300	252 MIPS	256Kb	5
Ardupilot[2]	2.6x1.6x0.4	31	1W	300	16MHz	256Kb	5
Swiftpilot	2.8x1.3x1.1	34.0	1.3W	2.5k	-	-	5
wePilot 1/3000	6.3x4.6x2.4	1020	6W	-	400MHz	64Mb	12
SkyCircuit-SC2	4.8x3.1x1.7	285	1W	-	-	-	4-15
SmartAP	2.4x1.6x.5	16	-	-	168MHz	-	-
Paparazzi	3.6x2.0x.7	30	-	-	72MHz	-	-
GNC1000	4.0x6.3x3.0	1360	6W	-	264MHz	256Kb	-

(1) Autopilots are currently in development and unavailable for purchase.

(2) Open source autopilots

Hardware aside, most companies are forced to protect the intellectual property and the software integrity by limiting user access. While this may be desirable for applications that only require the integration of payload hardware, it limits the ability of researchers to implement novel software. Stabilis, on the other hand, is presented as an open-architecture autopilot. One notable exception to the hardware computational limitations mentioned above is the “os” series by Airware. However, Airware products continue to be offered on a limited basis to the market as they are still largely in development. Moreover, even the Airware autopilots are fairly rigid with respect to autopilot hardware flexibility. In the next section, the concept of hardware design and flexibility is outlined.

3.1 Hardware Design

This thesis takes an alternative approach to conventional autopilot design by modularizing specific subsystems. In this way, the autopilot system can be prevented from becoming obsolete. Here modularity is defined as the ability to readily exchange system components. The benefits of modularity are two fold. Not only does modularity prevent obsolete,it allows components to be tailored to the application. For example, if one were flying an foam airplane a low cost inertial sensor such as the ArduIMU, rather than a higher cost inertial sensor such as the Epson M-G350, could be used. Furthermore, broken subsystems can be swapped out individually, minimizing the impact on system.

A system block diagram of key components was identified and is provided in

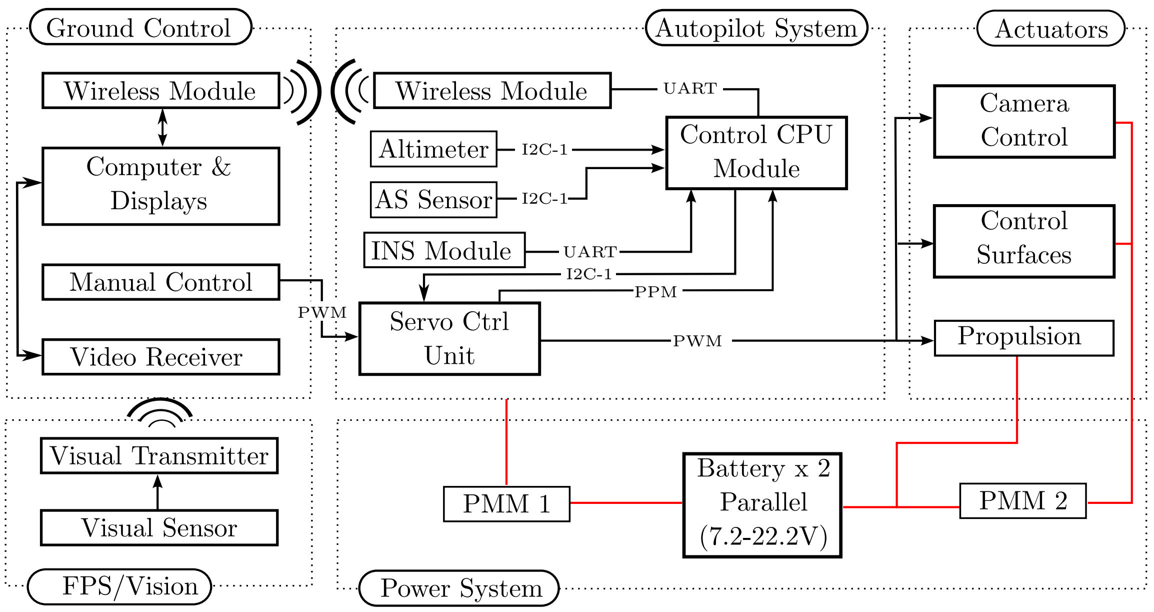


Figure 3.1: The autopilot block diagram shows the different components and their communication protocols. Note that the protocols may be changed depending on the modules used.

Figure 3.1. The benefits of a modular design are only justifiable in those components for which technology advances quickly, or component price differs largely. Thus, the following components were selected to feature modularity: flight control computer, inertial navigation system and wireless ground control communications module. Note these components are identified with the keyword "Module" in Figure 3.1.

One must be especially concerned with the form, weight and power consumption of all components in aerospace design. Benchmarking efforts dictated the following requirements for Stabilis.

1. Minimum 750MHz clock speed with 256Mb RAM

2. Minimum of 8 servo outputs/inputs, PPM compatibility, and SBus compatibility

3. Feature RS232, RS485, UART, I^2C, SPI, CAN, ethernet, and USB compatibility (with a minimum of 3 serial based connections not including necessary components, i.e. INS, airspeed, and wireless module)

4. Maximum of 80 grams, dimensions not to exceed 3.0 x 3.0 x 1.5 inches, and power consumption less than 1W including flight essential sensors.

Since COTS autopilots span a wide spectrum of capability, the requirements above were provided as generic guidelines for being competitive with the market as it stands. However, the ultimate goal is to design a system that can accommodate new hardware as it becomes available to prevent the system from ever becoming obsolete.

3.1.1 Modular Components

Flight Control Computer

The flight control computer (FCC) is responsible for interacting with almost every component on-board the aircraft, as well as the ground control station. Thus, it is the primary hardware limitation when it comes to overall system functionality. It must be able to analyze and filter sensor inputs, log data, communicate with the ground station and compute automatic flight control system outputs. Given all of these roles, it is paramount to ensure that a vast array of peripherals are supported in order to maximize the capabilities of the UAS.

A market survey was conducted in order to identify the most suitable computer for the application. The specifications of wich are summarized in Table C.1 in Appendix C. The Beaglebone Black, shown in Figure 3.2a, was the most suitable for the application given the design parameters above. There were a few notable factors that played into the selection. First, those embedded systems that feature a small form factor, such as the Gumstix DuoVero, Arduino Due and Arietta G-25, do not feature the connectors necessary to implement valuable peripherals, i.e. USB, HDMI, ethernet, etc. Rather, they feature simple headers for breakout boards that have the connectors. Thus, the form factor was actually higher than the RaspberryPi and Beaglebone products. Additionally, community and product support played a role in the final decision, as some of the products seemed to have little to no support.

The Beaglebone black features Sitara AM3359AZCZ100 1GHz, 2000 MIPS, processor with 512MB DDR3L 606MHZ RAM and 2GB 8bit EMMC flash on-board storage, all coming in at only 40 grams. The remaining specifications can be found in Appendix C. Usually, the autopilot system is designed around the selection of the central computer. However, in this case, the computer is considered modular since the subcomponents can be easily adapted to fit other similar linux-based embedded computers by simply modifying the routing and connection of the System Integration Board (SIB). Furthermore, Circuit Co, the makers of the Beaglebone embedded computers, have now 4 generations of boards whose form factor has not changed. Thus, in the future the computer may be able to be swapped with absolutely no modification at all. A prime example of this is the Beaglebone Black Sapphire, which was released in the middle of the development of Stabilis. Rather than being forced to design an entirely new autopilot that featured a more advanced processor, the old embedded computer was simply swapped out for the new one.

(a)

(b)

Figure 3.2: (a) Beaglebone Black embedded computer (b) VN200 MEMS Inertial Navigation System

Inertial Navigation System

For the sake of brevity, it is assumed that readers understand the differences between navigation sensors; interested readers are referred to [48] for more information regarding the differences between navigation sensors. It is common practice to integrate the INS in the autopilot package to reduce the wiring footprint, and the overall form factor of the autopilot. However, with the rapid advancement of microelectromechanical systems driven by the UAS market potential, INS are quickly increasing in precision. Furthermore, navigation sensor technologies such as ring laser gyros (RLG) and fiber optic gyros (FOG) are decreasing in price, making these technologies more available to users. Since, many COTS navigation sensors come in rugged, self-contained packages, and it is often advantageous to place navigation sensors in locations which are not ideal for the FCC, and vice-versa, this essential sensor was not included on the Systems Integration Board.

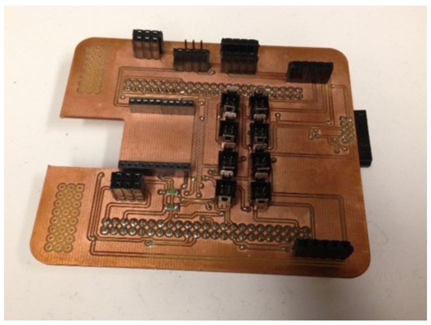
This allows users to integrate navigation sensors that match the application, both in price and form factor. The number of navigation sensors available on the market is staggering, see Table C.2 for examples. In the early development stages VectorNav's VN200, depicted in Figure 3.2b, was selected as the first navigation sensor to be implemented. Although, other navigation sensors were implemented such as the Epson M-G362, and the KVH 1750, all flight testing was conducted with the VN200.

Wireless Communication Device

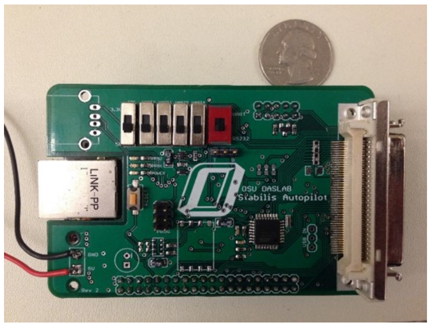
The fidelity of the connection between the ground control station and the UAV is paramount. The ground control station is a relay for all of the relevant information on-board the UAV. Similar to the navigation sensors, wireless communication technology is rapidly advancing, and largely varies in cost. Thus, this component is also located off board Stabilis. In this way, one can also reduce the amount of EMI introduced from other systems. Three low-cost, serial wireless communication modules were tested to determine the robustness of the electronics and connection: XBee 900 RPSMA, 3DR Radio Set, and JDrones jD-RF900. It was found that the JDrones jD-RF900 performed extremely.

3.1.2 Systems Integration Board

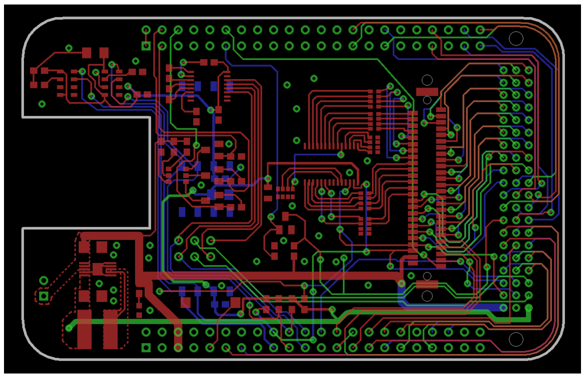
The purpose of the Systems Integration Board, or SIB, is the electronic integration of the FCC with the other sensors and components. The SIB was designed primarily with form factor and robustness in mind. The three design iterations, including the final design, are shown in Figure 3.3c. In order to mitigate issues with loose or faulty connections, as well as allow for quick connect/disconnect action, a screw type main connector was chosen. The servo control module was designed to go on a smaller board connected via a mezzanine connector towards the middle of the board. Two airspeed sensors and two barometers are provided on-board the SIB for robustness.



(a)



(b)



(c)

Figure 3.3: (a) Prototype systems integration board (b) First iteration of PCB design for systems integration board (c) Current PCB layout of systems integration board

Servo Control Unit

The servo control module is the most crucial component to allow for remotely piloting the vehicle. The servo control module, or SCM, is basically a multiplexer. In order to prevent accidents, pilots must be able to take manual control of the plane at any point in time. Thus, the SCM must be designed to function even if the autopilot were to fail entirely. The subsystem is shown in 3.4 for reference.

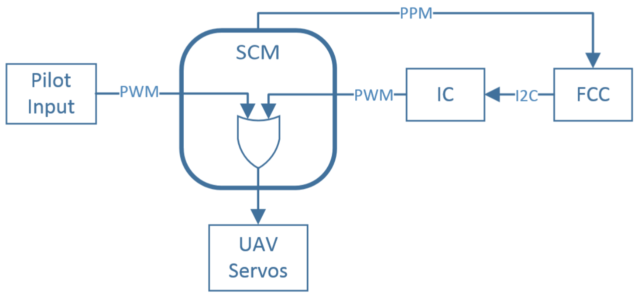


Figure 3.4: SCM Block Diagram

The pilot input signal to the SCM is given as a PWM signal which is then converted to a PPM signal and sent to the FCC for analysis. The relay of pilot input to the FCC will allow for aircraft stability augmentation and similar control techniques in the future. The FCC control signal, which is produced by the ACS, is then sent back to the SCM. The SCM muxes between the signals based on pilot input, so the pilot ultimately has the control to turn off the ACS as needed.

Systems Integration Board Sensors

The Honeywell, HSCMRRN001PD2A3, was chosen for its superior resolution, accuracy and form factor to provide the differential pressure reading of the airspeed sensor. Additionally, the Freescale MPL3115A2 Precision Barometric Altimeter was chosen as a simple, inexpensive option for sensing altitude. The sensor features a resolution of 0.3 meters, with maximum altitude of approximately 8000 meters.

3.1.3 Power Management System

Although there exist COTS power management systems that could be used for UAS, a simple SWAP analysis quickly showed that most COTS systems were too large or heavy for UAS applications. The remaining power management systems were found to lack quality or be extremely cost prohibitive. Furthermore, no standardization exists for on-board voltages for UAS platforms. Thus, the power management module must be able to handle a wide variety of input voltages, and in the case of a failure, be able to protect electronic components from voltage surges.

The Castle Creations Battery Eliminator Circuit, or CCBEC, was chosen as a voltage regulator due to the components small footprint, efficiency and excellent reputation among users. Despite its user-reported reliability, the voltage regulator was intended for RC hobbyist applications, so quality was still a large concern. In order to mitigate issues with quality and robustness, two CCBECs were placed in parallel with a surge and reverse voltage protection circuits. This configuration allows for a single CCBEC to fail open or closed without adverse consequences to the electronics being powered. Note that the CCBEC circuit could be replaced with any voltage regulator with similar specifications. The final configuration allowed for input voltages from 7.4V to 22.2V, with a maximum 10 amps continuous.

3.2 Software Design

Operating in the outdoor environment poses the challenge of onboard computation for all control and automation algorithms. Choosing an operating system is one of the most important aspects of the software development in the context of embedded development. The operating system must be light enough to devote most of the processor to core tasks, but minimize development time when trying to perform basic tasks. The following key requirements drove the choice of operating system:

- Tasks must be performed in a deterministic manner
- Operating system must be able to provide precise timing when executing tasks
- Mitigation of low-level, time-consuming programming.
- Level of development of the operating system itself
- Community support

The first two requirements above includes performing tasks in a given sequence, as well as, the ability to provide tolerances on the amount of time it takes to complete a given task with a 95% confidence level. Although students are exposed to embedded systems, most lack the experience to operate them without community support. Thus, it is extremely important that support, whether via the university or online, is available. Several key operating systems were looked at and can be found in Table 3.2 below. The lack of a real-time operating system, or RTOS, for the Beaglebone Black at the time the design decision was made forced authors to utilize Ubuntu.

Table 3.2: Operating System Design Matrix

	Angstrom	Ubuntu	Arch	QNX RTOS	Xenomai
Determinism	0	0	0	1	1
Timing	0	0	0	1	1
Integration	0	1	-1	-1	0
Level of Development	0	0	0	1	0
Support	0	1	0	-1	1
Total	**0**	**2**	**-1**	**1**	**3**

Since Ubuntu is not a RTOS, the sampling time will vary slightly over time. The sampling time variance was quantified using the experiment outlined in [49], and relevant statistical data is provided in Figure 3.5.

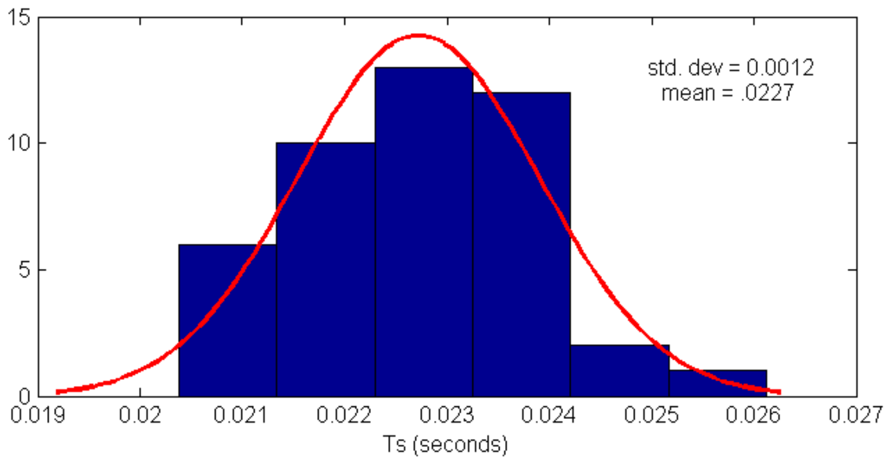


Figure 3.5: Histogram depicting sampling time variance

The standard deviation of the sampling time was significant; however, provided that the mean was used as the sampling time, it was concluded that the error associated with the controller would be within acceptable limits for research purposes. Recently, however, an RTOS by the name of QNX Neutrino RTOS was released for the BBB. Thus, due to the necessity for determinism in aerospace control systems, as well as the performance based motivations, this author strongly recommends the QNX Neutrino RTOS for future development.

3.2.1 Thread Design

A multithreaded architecture was selected for the autopilot in order to ensure the integrity and robustness of the software as whole. Performance is enhanced by allowing the OS to optimize those task that can be done is psuedo-parallel. The term pseudo-parallel is used here to describe those tasks that are executed in parallel, but on a single processor, and therefore, not parallel processing in its truest sense. The thread structure was broken up into distinct tasks based on functionality and hardware components. These threads are shown in Figure 3.6, and align well with the practices of past research [38]. In addition to executing the threads, the *main()* is tasked with autopilot initialization. This includes loading the configuration files such as, system gains, actuator limits and sensor profiles, as well as previous mission attributes must be loaded.

Multi-threading while advantageous, has its disadvantages as well. Thorough protection of global variables is required to prevent race conditions, a condition in which two threads attempt to read or write to the same variable simultaneously. Race

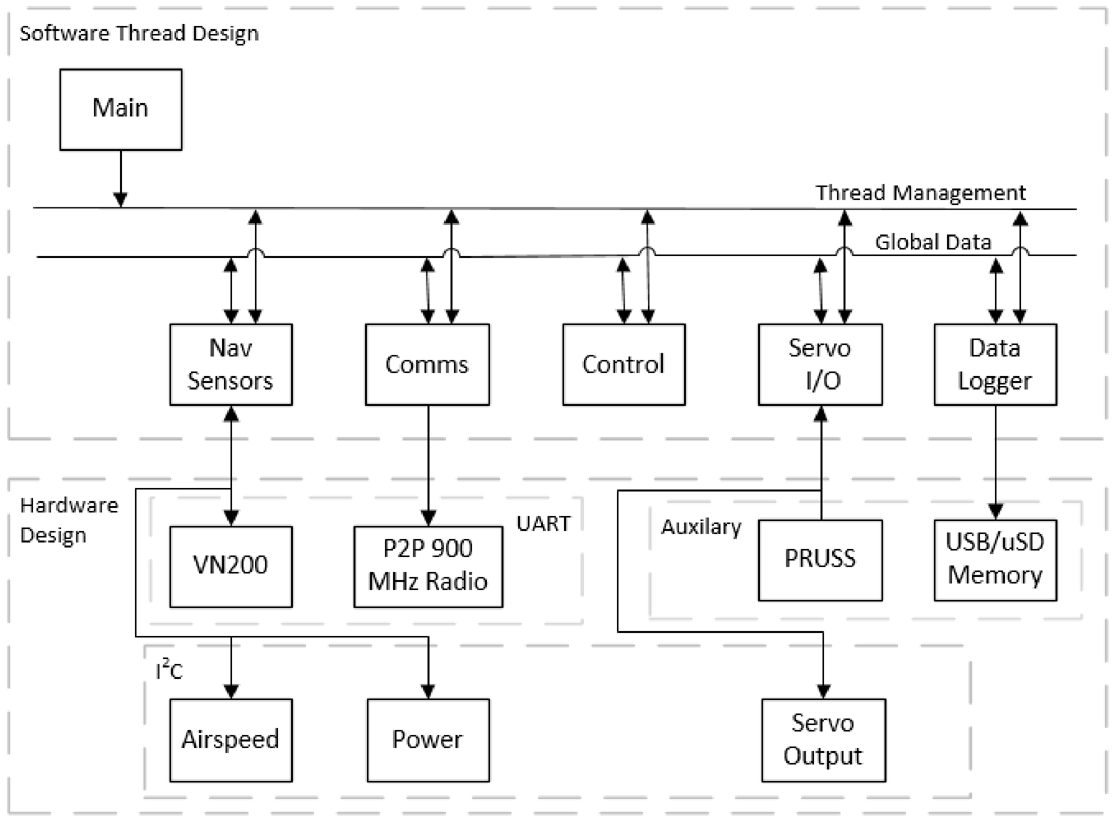
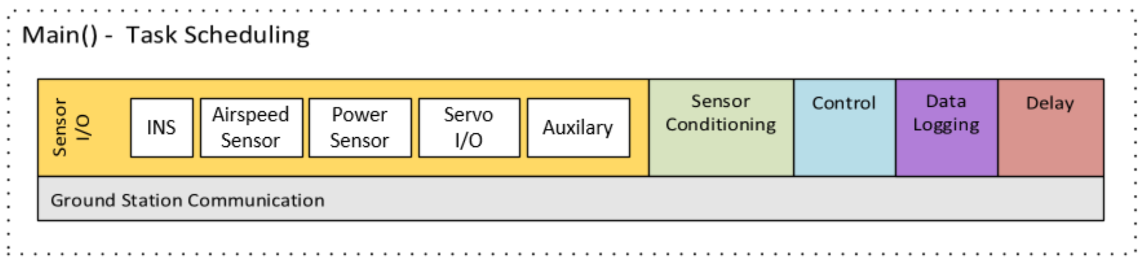
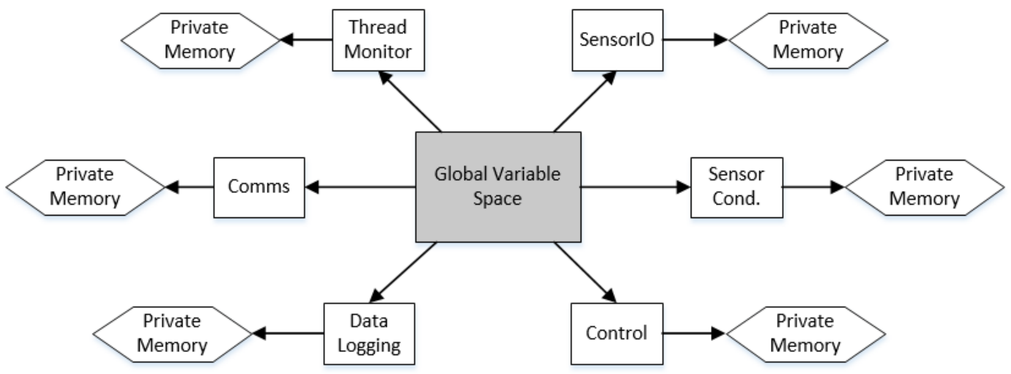


Figure 3.6: Thread design block diagram

conditions cause erratic behavior in software, and are extremely difficult to debug. Additionally, the tasks for each thread must be scheduled such that the control is executed properly. The task scheduling should be handled through the *main()* as shown in Figure 3.6d. Threads contain a variable space that is only accessible locally. However, the sole means by which threads can communicate is the global variable space. In order to prevent race conditions and protect the integrity of data in the global variable space, a mutex or semiphore must be used.



(d)



(e)

Figure 3.6: (a) Task scheduling performed by *main()* (b) Variable spaces associated with multithreaded architecture.

3.2.2 Ground Control Station Software

The ground control station is the primary means by which operators plan, execute and monitor UAS missions. Many ground control station software platforms exist, but none are as well documented, platform independent, and community supported as qGroundcontrol (QGC). The QGC software is compatible with the three major operating systems, Windows, Linux and Mac; features serial, UDP and mesh networks communication compatibility; and, posses real-time plotting and logging capabilities of onboard parameters. QGC is flexible and easily modified since it is Qt based. Lastly, QGC utilizes a highly efficient communication protocol called MAVLINK. MAVLINK is extensively tested and quite possibly the most widely used communication protocol in the UAS research community. It uses C-structs to efficiently pack information for transmission over serial and UDP connections. Readers are referred to the QGC website for user manuals and documentation.

3.3 Experimental Setup

3.3.1 Airframe

The Skyhunter was used as the primary experimental fixed wing platform. The platform is a COTS, boom-tail design that features tough EPO Construction, a large payload bay capable of carrying upwards of 7 pounds, and a 1.8 meter wingspan. The high wing design adds a significant amount of stability and robustness in the presence of wind. The aircraft features ailerons and an elevator as control inputs, but no rudder.

3.3.2 Autopilot Control Scheme

As a baseline, the autopilot control architecture from Chapters 6, 10 and 11 of [35] is utilized, although the implementation of the control algorithms developed in Chapter 2, should be included in future work. The path manager uses a path-fillet-path algorithm to smooth the transitions between waypoints.

3.3.3 Simulation Environment

Critical to the design of any system is a simulation environment that allows for mitigation of potential issues. The XPlane Simulator by Laminar Research was utilized to simulate aircraft dynamics in order to evaluate Stabilis. The Skyhunter airframe was modeled using plane maker and is shown in Figure 3.7.

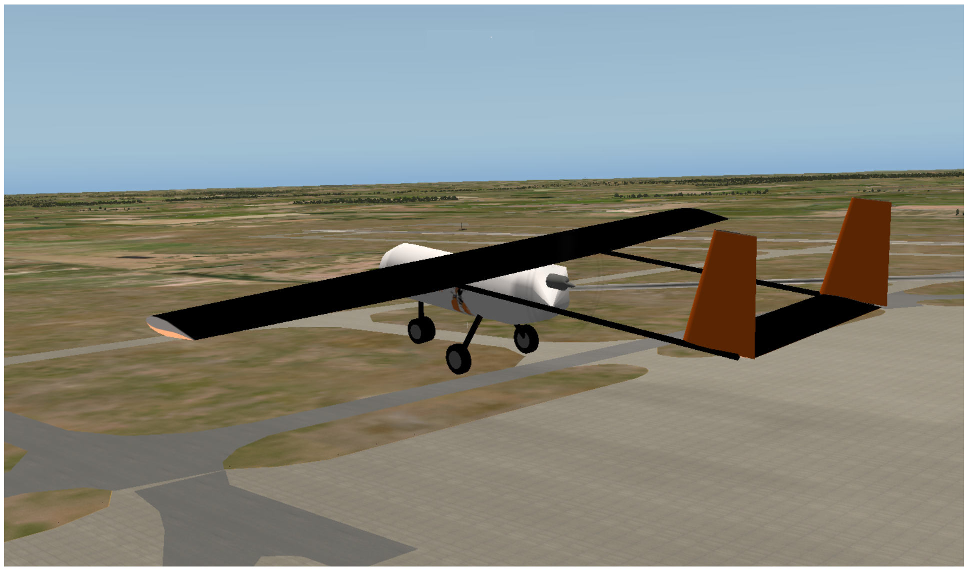


Figure 3.7: Skyhunter model in the X-Plane environment

3.4 Flight Test Results

A course consisting of five waypoints was used to evaluate the waypoint tracking capabilities of Stabilis. All of the flight test experimentation was performed at OSU's Unmanned Aircraft Flight Station. During experimentation wind disturbances on the course ranged from six to twelve knots, with gusts up to four knots.

3.4.1 Control Loop Tracking

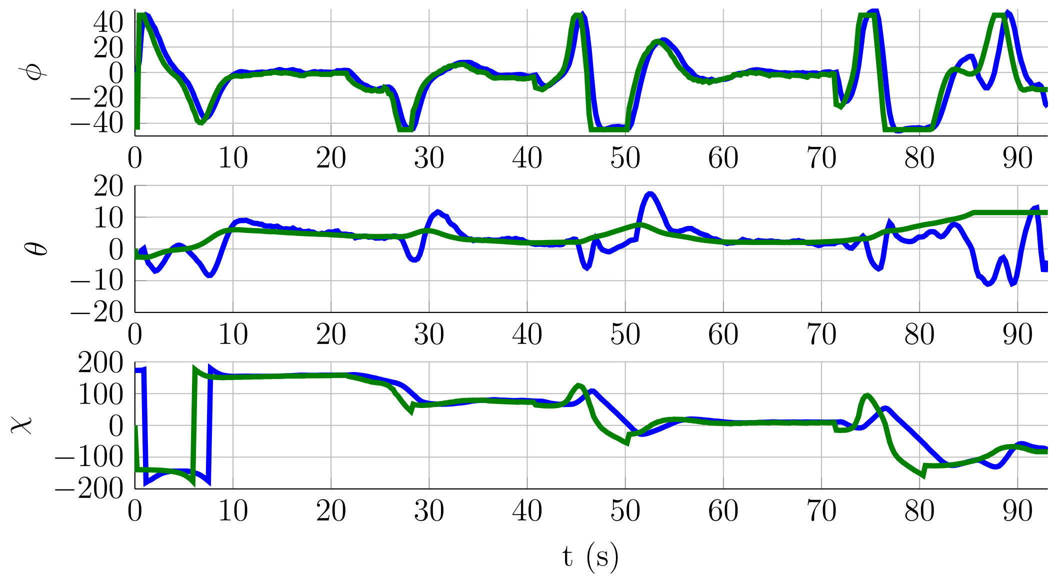
The outer loop and attitude control performance is shown in Figure 3.8. The controller parameters that proved to be successful in simulation were utilized in flight testing. Although the simulation model was fairly accurate each axis took some additional tuning during flight testing. While the controller parameters for tracking ϕ were relatively insensitive, tracking of the heading angle, χ, and pitch angle, θ, were found to be more sensitive. Although the fixed-wing aircraft tracks the waypoints, there is an issue with the logged data in Figure 3.8 (b). At various times in the altitude error, crosstrack error and airspeed error, there are zero values where the curve trends infer otherwise. Thus, it is safely inferred that there is either an error in the software or the hardware. This is clearly a serious issue that is currently being investigated and will be resolved.

3.4.2 Waypoint Tracking

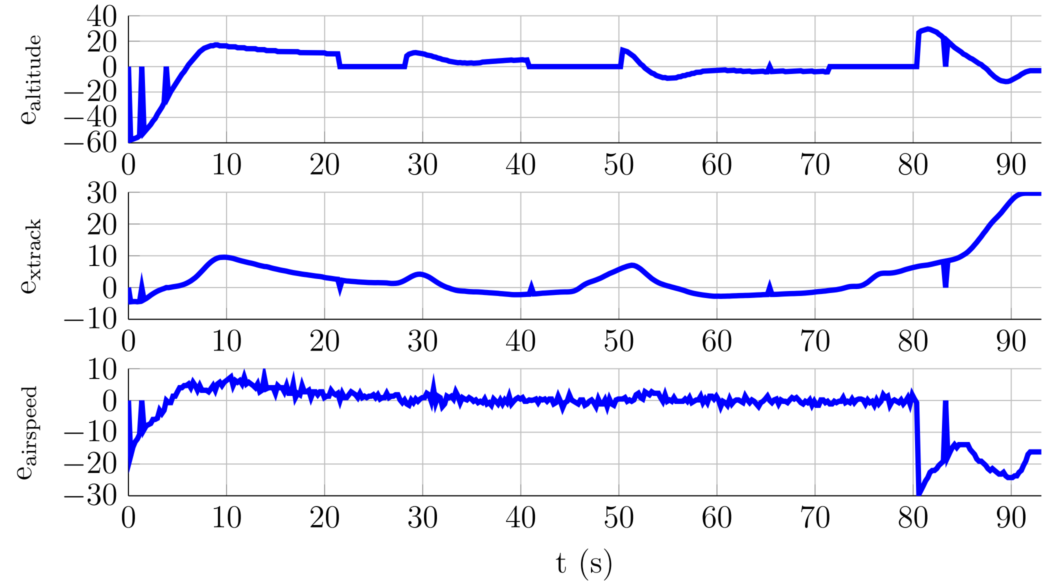
The waypoint tracking performance is shown in Figure 3.9. Arrows denote the entry point of the aircraft for two different flight tests. The autopilot was in fully autonomous mode for the entire duration of the each flight test. A path-fillet-path algorithm was used to make the smooth transitions and to avoid significantly overshooting each waypoint. To compensate for the wind, the aircraft experienced a significant amount of side slip between waypoints three and four. However, it maintained similar tracking performance across all waypoints.

3.5 Summary

With the exception of the ground control station, the control hardware and software is almost entirely custom in design. The hardware system optimized between weight, form factor, computational ability, and peripheral compatibility. The customized hardware was designed using a modular approach in order to allow for application-based selection of hardware. The mission planning, monitoring and execution is performed using the qGroundcontrol software, and XPlane was utilized as the primary simulation environment for control algorithm evaluation. Flight testing results show the ability of the autopilot to perform waypoint tracking.



(a)



(b)

Figure 3.8: (a) Attitude tracking performance of the aircraft, where desired and actual attitude is given by green and blue respectively (b) Longitudinal and lateral outer loop tracking

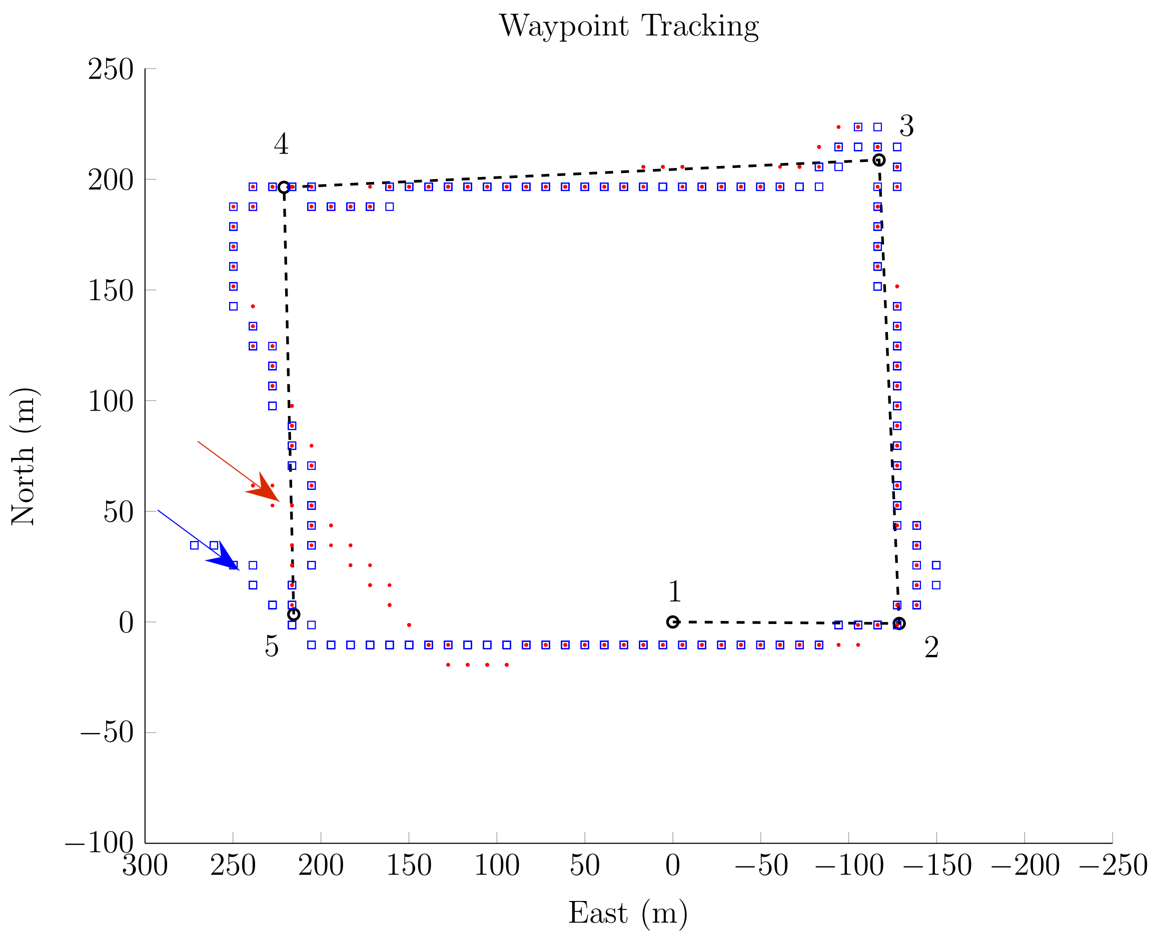


Figure 3.9: Straight line waypoint tracking performance over the course of two flights. Flights were completed with crosswinds up to 12 knots.

Chapter 4

Conclusion

4.1 Summary

This thesis seeks to address the problems of control-transfer associated with outdoor, fixed-wing flight in order to reduce development time and costs associated with new unmanned systems. A control architecture is formulated using a new class of data driven control, GP-MRAC. Feedback linearization was used to show the efficacy of modeling the underlying uncertainty between aircraft with online GP methods. The simulation results provided indicate the feasibility of RCT using online GP based methods.

To this end, this thesis also addresses a deficiency in current COTS autopilots to facilitate cutting edge research in control systems and autonomy. A modular, open-architecture autopilot was design, developed and evaluated in order to address this issue. Tests were conducted to show that the computational requirements of GP-MRAC were satisfied by autopilot hardware. Next, traditional control techniques were employed to show the baseline performance of the autopilot. Thorough simulation and ground testing were completed to mitigate risks associated with unmanned, autonomous flight. Finally, flight tests were conducted, and the results are provided for performance metrics.

4.2 Future Work

Recommendations for future work could include:

- Vetting fixed-wing lateral control architecture posited using simulation
- Flight testing of GP-MRAC fixed-wing control architecture
- Relaxation of Assumption 2.2 by way of an adaptive inner loop architecture
- Relaxation of Assumption 2.3 by way of the following:
  - The implementation of psuedo-control hedging to aid in protection of input saturation and undesirable adaptation
  - Establishing physical limitations of aircraft, e.g. acceptable structural load factors, by way of basic aircraft performance calculations to ensure the reference model falls within the operational abilities of the aircraft
- Implementation of RTOS system on flight control computer of Stabilis
- Mitigation of the sensor/software issues presented in flight testing results.
- Implementation of fault detection where mutiple sensors are used. See section 3.1.2

Appendix A

GP-MRAC in Lateral Motion

This section draws on the assumptions and discussion provided in 2.3.1. In lateral motion, the heading angle, defined $\chi = atan2(V_n, V_e)$, is used as for guidance; where, $X_q = atan(q_n, q_e)$. The guidance law is given by [35] as,

$$\chi^c(t) = \chi^\infty - \frac{2}{\pi}\chi^\infty atan(k_{path} e_{pxy}(t)) \ ; \tag{A.1}$$

where, k_{path} and χ^∞ are designer chosen constants of the reference model $\ddot{\chi} = -2\omega_n\zeta\dot{\chi} - \omega_n^2(\chi_c - \chi)$. For the lateral case, let $z = [\begin{array}{ccccc} p & r & \phi & \psi_{dot} & \dot{\chi}^T \end{array}]^T$, and let $\delta_a \in D_{\delta_a}$. Note that the ailerons are chosen for lateral input, since many fixed wing UAS platforms do not feature rudders. Then Assumption 2.2 for the lateral direction can be stated: For all $\delta_a(t) \in D_{\delta_a}$, there exists a finite value $B > 0$ such that $|p(t)| < B$ and $|r(t)| < B$. It is assumed that the outer loop system states for the lateral motion can be modeled by the following differential equation,

$$\begin{aligned} \dot{\chi}_1(t) &= \chi_2(t) \\ \dot{\chi}_2(t) &= f(z(t)) + b(z(t))\delta_a \end{aligned} \tag{A.2}$$

The function f is assumed to be Lipschitz continuous in $z, \dot{z} \in D_z$, and the systems are assumed to be finite input controllable. The following assumption characterizes the controller on the system for lateral motion [29]:

Assumption A.1 *For the source and transfer system, there exists a control law* $g : D_z \to D_{\delta_a}$ *such that* $\delta_a(t) = g(z, \ddot{\chi}_{rm})$ *drives* $\chi \to \chi_{rm}$ *as* $t \to \infty$. *Furthermore, the control law is invertible w.r.t.* δ_a, *hence the relation* $\ddot{\chi}_{rm} = g^{-1}(z, \delta_a)$ *holds. [29]*

.

The tracking error is defined as, $e_\chi = \chi_{rm} - \chi$, and the psuedocontrol, ν to be

$$\nu = \nu_{rm} + \nu_{pd} - \nu_p - \nu_{ad}. \tag{A.3}$$

where, $\nu_{rm} = \dot{x}_{2_{rm}}$, $\nu_{pd} = [K_1 \; K_2]\, e_c$ and the robustifying term $\nu_p = p$. The remainder of the lateral formulation follows directly from 2.3.1.

Appendix B

Autopilot Specifications

Specifications for the autopilots benchmarked in Table 3.1 are provided in Table B.1 and B.2 below. These specifications were used to aid in selecting components for Stabilis. It should be noted that many of the autopilot companies do not readily advertise the specifications of their product. Thus, unfortunately, a significant amount of information was not provided since it was not disclosed.

Table B.1: Autopilot compatible peripherals

	Serial I/O								Digitial I/O			ADC
	RS232	RS422	RS485	UART	I2C	SPI	CAN	Eth.	PWM	PPM	SBUS	
Kestrel 2.2	4 Serial Ports (STD, SPI, I2C)								10	-	-	2 - 12bit
MP 2028g	-	-	-	-	-	-	-	-	24	-	-	>1
Piccolo Nano	3	-	-	-	-	-	1	-		-	-	-
Piccolo LT	3	-	-	-	-	-	1	-		-	-	-
Piccolo II	3	-	-	-	-	-	1	-	16	-	-	up to 4
Unav3521	-	-	-	-	-	-	-	-	4	-	-	-
osflexPilot[2]	>1	>1	>1	>1	>1	>1	>1	-	>1	-	-	>1
osnanoPilot[2]	-	-	-	>1	>1	>1	>1	-	8	-	-	-
osflexQuad[2]	-	-	-	-	>1	-	>1	-	8	-	-	-
Slugs	-	-	-	-	-	-	-	-	-	-	-	-
PixHawk	-	-	-	5	1	1	2	-	-	1	1	1 - 12bit
Ardupilot	-	-	-	2	1	-	-	-	8	-	-	12
Swiftpilot	-	-	-	-	-	-	-	-	6	-	-	-
wePilot1000/3000	1	-	-	-	-	-	-	-	10	-	-	6 - 12bit
SkyCircuit-SC2[1]	-	-	-	-	-	-	-	1	6	-	-	-
SmartAP	-	-	-	1	-	-	-	-	6	-	-	2
Paparazzi	-	-	-	3	2	2	1	-	6	1	-	1
GNC1000	2	1	8	-	-	-	2	4	6	-	-	-

Note 1) Expansion boards available.

Note 2) Specific values not advertised

Table B.2: Autopilot sensor specifications

	Rate Gyroscopes			Accelerometers			Air Speed		Max	Op. Temp
	Range	Res.	Bias	Range	Bias	Noise	Range	Res.	Alt.	
	(°/s)	(°/s)	(°/vHr)	(g)	(mg/LSB)	(g/vHz)	(kts)	(kts)	(ft)	(°C)
Kestrel 2.2	300	0.0318	6	10	1.50	200	252	0.05	22k	−40 to +85
MP 2128g	-	-	-	-	-	-	500	-	12k	-
Piccolo Nano	-	-	-	-	-	-	-	-	-	−30 to +80
Piccolo LT	300	-	-	6	-	-	192	-	-	−30 to +80
Piccolo II	300	-	-	10	-	-	155	-	-	−40 to +80
Unav3521	300	-	-	-	-	-	-	-	-	-
osflexPilot	-	-	-	-	-	-	-	-	-	-
osnanoPilot	-	-	-	-	-	-	-	-	-	-
osflexQuad	-	-	-	-	-	-	-	-	-	-
Slugs	-	-	-	-	-	-	-	-	-	-
PixHawk[1]	250	0.0088	1.8	8	0.244	~600	193	1.1	>60k	−30 to +70
Ardupilot[1]	1000	32.8	3	8	0.2	400	60	0.5	>60k	−30 to +70
Swiftpilot	2000	-	-	-	-	-	-	-	-	0 to 70
wePilot1000/3000	100	-	-	10	-	-	-	-	10k	-
SkyCircuit-SC2	-	-	-	-	-	-	-	-	15	−20 to +60
SmartAP[1]	1000	32.8	3	8	0.2	400	External		>60k	−30 to +70
Paparazzi[2]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
GNC1000	200/1000	-	3	10	5	-	450	1	60k	-

Note 1) FS values assumed; the resolution and range are dependent on this assumption.

Note 2) User defined IMU

Appendix C

Component Benchmarking

C.1 Flight Control Computer Survey

A summary of some of the flight control computers that were considered are provided in C.1.

Table C.1: Embedded system specifications

Model	Clock Speed	Memory	Price	Peripherals
Arduino Due	84MHz	512Kb	$45	I2C(2), UART(2), GPIO(28), 12 bit ADC(2), USB
Arietta-G25	400 MHz	256MB	$40	I2C (2), UART (3), SPI (1), 10 bit ADC (4), PWM (4), onboard WiFi, USB
Beaglebone	720 MHz	256MB	40	I2C (3), UART (6), SPI(2), CAN (2), 12 bit ADC (8), 400MHz PRUSS (2), TCP/UDP, uSD, USB
Beaglebone Black	1 GHz	1 GB	$40	I2C (3), UART (6), SPI(2), CAN (2), 12 bit ADC (8), 400MHz PRUSS (2), TCP/UDP, uSD, USB, uHDMI
RaspberryPi	700 MHz	512MB	$40	I2C (1), UART (1), GPIO (8), SPI (1), 12 bit ADC x 2, uSD, USB, HDMI
Intel Edison	2 x 500 MHz	1GB	$50	I2C (1), UART (1), GPIO (20), PWM (4), SPI (1), uSD, onboard WiFi & Bluetooth
Gumstix Overo	1 GHz	512MB	$140	I2C (3), UART (2), GPIO (??), SPI (3), uSD, onboard WiFi & Bluetooth

C.2 Inertial Sensor Survey

The following list is a compilation of available COTS IMU/INS/AHRS sensors. Table C.2 was used primarily in the early design phases of Stabilis in order to characterize and select an appropriate inertial navigation system. It is provided as a reference to the reader.

Table C.2: List of Inertial Sensors [1]

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
Daisy-7	ACME Systems	MEMS	Yes	128.7
Adjacent Reality	open hardware	MEMS	no	
Spatial	Advanced Navigation	MEMS	Yes	3000
Spatial Dual	Advanced Navigation	MEMS	Yes	10000
Spatial FOG	Advanced Navigation	FOG	Yes	35000
ARN-NS0535	Aeron Systems	MEMS		
impactAIMS	AIMS	MEMS		
uMotion	AIMS	MEMS		
Navigation	AIMS	MEMS	yes	
FOG	AIMS	MEMS/FOG		
AHRS/INS	American GNC	MEMS		
AHRS/INS/GPS	American GNC	MEMS	yes	
AHRS/INS/DGPS	American GNC	MEMS	yes	
ADIS16355	Analog Devices	MEMS		600
Opal	APDM	MEMS		
AHR150A-1	Archangel Sys.	MEMS		
3D-Bird	Ascension	MEMS		1768
INU	Atair Aerospace	MEMS	yes	
Micro INS	Athena (Rockwell)	MEMS	yes	
SensorPac	Athena (Rockwell)	MEMS	yes	
SilMU 01	UTC Aerospace (BAE)	MEMS		
SilMU 02	UTC Aerospace (BAE)	MEMS		
SiNAV 02	UTC Aerospace (BAE)	MEMS	yes	
MMQ 50	BEI Systron Donner	MEMS		
MMQ-G	BEI Systron Donner	MEMS	yes	

Continued on next page

Table C.2 – *Continued from previous page*

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
C-MIGITS III	BEI Systron Donner	MEMS	yes	
MiniSense 2	CDLTD	MEMS	input	
TOGS	CDLTD	RLG	input	
MiniRLG2	CDLTD	RLG	input	
MiniPOS2	CDLTD	RLG	input	
MiniPOS/NAV	CDLTD	RLG	input	
MiniTilt	CDLTD	MEMS		
MicroTilt	CDLTD	MEMS		
MiniSense	CDLTD	MEMS		
ADAHRS	Chelton Avionics	MEMS	input	26000
CHR-6d	CH Robotics	MEMS	no	125
CHR-6dm	CH Robotics	MEMS	no	
CHR-6um6	CH Robotics	MEMS	no	
GP9	CH Robotics	MEMS	yes	320
Crista	Cloudcap	MEMS		2000
Piccolo	Cloudcap	MEMS	yes	6000
Terrella 6	Clymer Tech.	MEMS		1300
NAV 420	Crossbow	MEMS	yes	
NAV 425EX	Crossbow	MEMS	yes	
NAV 440	Crossbow	MEMS	yes	6000
AHRS500	Crossbow	MEMS		14200
IMU440	Crossbow	MEMS	yes	
IMU700CB	Crossbow	FOG		12000
Landmark 10 IMU	Gladiator Tech.	MEMS		2495
Landmark 10 IMU/GPS	Gladiator Tech.	MEMS	yes	4995
Landmark 20 IMU	Gladiator Tech.	MEMS		3995
Landmark 20 IMU/GPS	Gladiator Tech.	MEMS	yes	5995
Landmark 10 GPS/AHRS	Gladiator Tech.	MEMS	yes	
Landmark 10 AHRS	Gladiator Tech.	MEMS		
Landmark 30	Gladiator Tech.			6600
HG 1700	Honeywell	RLG		9000
AG-1	Icewire	MEMS	No	199
iNAV-FMS-T	iMAR	FOG	input	
iIMU-FSAS	iMAR	FOG		
iIMU-FR-M1	iMAR		input	
iVRU-FAS-C167-IGS	iMAR	FOG/MEMS	input	
iVRU-FC-C167-MSL	iMAR	FOG/MEMS	input	

Continued on next page

Table C.2 – *Continued from previous page*

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
iVRU-SSA-C167	iMAR	FOG/MEMS	input	
iVRU-SSKS-C167	iMAR	MEMS	input	
iVRU-SBA1-C167	iMAR	FOG/MEMS	input	
iVRU-FA-C167	iMAR	MEMS	input	
iVRU-FKS-C167	iMAR	FOG/MEMS	input	
iTGAC-FK	iMAR	FOG/MEMS		
iHRP(Y)	iMAR	FOG/RLG	input	
iNAV-FMS	iMAR	FOG/RLG	yes	
iDIS-FMS	iMAR	FOG	yes	
iFLY	iMAR			
iuIMU-02	iMAR	MEMS	yes	
iTraceRT-F200	iMAR	FOG	yes	
OptoAHRS	Inertial Labs	Optical/MEMS	no	7499
AHRS-1	Inertial Labs	MEMS	no	3499
AHRS-2	Inertial Labs	MEMS	no	2999
VG	Inertial Labs	MEMS	no	2699
OS3D	Inertial Labs	MEMS	no	999
OS3DM	Inertial Labs	MEMS	no	999
ISIS-IMU	Inertial Science	MEMS		
ISIS-GPS	Inertial Science	MEMS	yes	
DMARS-R	Inertial Science	MEMS	input	
DMARS-I	Inertial Science	MEMS	input	
DMARS-GARS	Inertial Science	MEMS	yes	
InertiaCube2	InterSense	MEMS		1500
InertiaCube3	InterSense	MEMS		1800
MPU-9150	Invensense	MEMS	no	80
MPU-6000	Invensense	MEMS		15
KN-4072	Kearfott	RLG		
KN-4072A	Kearfott	RLG	yes	
KN-4073B	Kearfott	RLG		
KN-4074	Kearfott	RLG	yes	
KN-4075	Kearfott	RLG	yes	
KN-4077	Kearfott	RLG	yes	
KN-4051/2/3	Kearfott			
KN-4051/2/3G	Kearfott		yes	
KI-4801	Kearfott	RLG		
KI-4901	Kearfott	RLG	input	

Continued on next page

Table C.2 – *Continued from previous page*

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
KI-4902	Kearfott	RLG	input	
TG-6000	KVH	FOG		25000
CNS-5000	KVH	FOG/MEMS	yes	30250
LPMS-B	LP Research	MEMS	no	500
LPMS-CU	LP Research	MEMS	no	400
micro IMU	Memsense	MEMS		
nano IMU	Memsense	MEMS		2730
MIDG II	Microbotics	MEMS	yes	6750
MP 2028g	MicroPilot	MEMS	yes	5000
3DM-GX1	MicroStrain	MEMS		1500
3DM-GX2	MicroStrain	MEMS		
3DM-GX3-25	MicroStrain	MEMS		2295
3DM-GX3-35	MicroStrain	MEMS	yes	3075
3DM	MicroStrain	MEMS		
3DM-DH	MicroStrain	MEMS		
INERTIA-LINK	MicroStrain	MEMS		
LN-200	Northrop-Grumman			
Summit 34203A	Omni Instr.	MEMS		
Falcon GX	O-Navi	MEMS		1000
Phoenix AX	O-Navi	MEMS	yes	1200
UM6	Pololu	MEMS	input	200
FreeIMU	open hardware	MEMS		
AHRS200A	Rotomotion	MEMS		
AHPRS200A	Rotomotion	MEMS	yes	
CHIMU	Ryan Mechatronics	MEMS	No	299
Nav Board M3	Ryan Mechatronics	MEMS	No	299
IG-500A	SBG Systems	MEMS	no	2208.7
IG-500N	SBG Systems	MEMS	yes	4483.7
IG-500E	SBG Systems	MEMS	yes	
Ekinox INS	SBG Systems	MEMS	yes	32500
Ekinox AHRS	SBG Systems	MEMS	no	
MoveIt Senspod	Sensaris	MEMS	yes	
STIM300	Sensaris	MEMS	yes	7800
STM32F3	ST Semiconductors	MEMS	no	10.66
65210A	Summit Instr.		yes	
65210E	Summit Instr.		yes	
CompaNav 2	Teknol	MEMS	input	

Continued on next page

Table C.2 – *Continued from previous page*

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
CompaNav 2T	Teknol	MEMS	input	
Autopilot	Teknol	MEMS	input	6000
Nanosatellite	Tethers Unlimited	MEMS	input	
CC2541 DevKit	Texas Instruments	MEMS	no	25
Colibri	Trivisio	MEMS	no	550
Colibri wireless	Trivisio	MEMS	no	800
Atom	UAV NAvigation	MEMS	input	
Polar	UAV NAvigation	MEMS	yes	
Vector	UAV NAvigation	MEMS	yes	
Proton	UAV NAvigation	MEMS	yes	
VN-100	Vectonav	MEMS	no	500
VN-200	Vectonav	MEMS	yes	500
VN-200 dev. kit	Vectonav	MEMS	yes	2500
x-IMU	x-io	MEMS	no	249
MTi-10 IMU	Xsens	MEMS		1170
MTi-20 VRU	Xsens	MEMS		2080
MTi-30 AHRS	Xsens	MEMS		2340
MTi-100 IMU	Xsens	MEMS		1820
MTi-200 VRU	Xsens	MEMS		3380
MTi-300 AHRS	Xsens	MEMS		4940
MTi-G-700 GPS/INS	Xsens	MEMS	yes	4940
3-Space USB	YEI	MEMS	no	145
3-Space Embedded	YEI	MEMS	no	99
3-Space Wireless 2.4G	YEI	MEMS	no	220
3-Space Bluetooth	YEI	MEMS	no	290
3-Space Data-logging	YEI	MEMS	no	180
3-Space Data-logging HH	YEI	MEMS	no	192

Bibliography

[1] "A small list of imu / ins / inu." http://damien.douxchamps.net/research/imu/. Accessed: 10-30-2013.

[2] P. A. Ioannou and P. V. Kokotovic, "Instability analysis and improvement of robustness of adaptive control," *Automatica*, vol. 20, no. 5, pp. 583–594, 1984.

[3] K. S. Narendra and A. M. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," *Automatic Control, IEEE Transactions on*, vol. 32, no. 2, pp. 134–145, 1987.

[4] S. Boyd and S. S. Sastry, "Necessary and sufficient conditions for parameter convergence in adaptive control," *Automatica*, vol. 22, no. 6, pp. 629–639, 1986.

[5] Astrom, *Adaptive control.* Mineola, N.Y: Dover Publications, 2008.

[6] K. Narendra, *Stable adaptive systems.* Mineola, N.Y: Dover Publications, 2005.

[7] G. Tao, *Adaptive control design and analysis.* Hoboken, N.J: Wiley-Interscience, 2003.

[8] E. Lavretsky, *Robust and adaptive control with aerospace applications.* London New York: Springer, 2013.

[9] E. N. Johnson, *Limited authority adaptive flight control.* PhD thesis, Citeseer, 2000.

[10] J. D. Boškovic and R. K. Mehra, "Intelligent adaptive control of a tailless advanced fighter aircraft under wing damage," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 5, pp. 876–884, 2000.

[11] S. Sastry and M. Bodson, *Adaptive control: stability, convergence and robustness.* Courier Dover Publications, 2011.

[12] R. W. Beard, N. B. Knoebel, C. Cao, N. Hovakimyan, and J. S. Matthews, "An l1 adaptive pitch controller for miniature air vehicles," in *AIAA Guidance, Navigation, and Control Conference, Keystone, CO*, 2006.

[13] C. Cao, N. Hovakimyan, and E. Lavretsky, "Application of l1 adaptive controller to wing rock," in *AIAA Guidance, Navigation, and Control Conference, Keystone, CO*, 2006.

[14] C. Cao and N. Hovakimyan, "Design and analysis of a novel l1 adaptive control architecture with guaranteed transient performance," *IEEE Transactions on Automatic Control*, vol. 53, no. 2, pp. 586–591, 2008.

[15] "L1 adaptive controller for systems with unknown time-varying parameters and disturbances in the presence of non-zero initialization error," *International Journal of Control*, vol. 81, no. 7, pp. 1147–1161, 2008.

[16] I. Kaminer, A. Pascoal, E. Xargay, N. Hovakimyan, C. Cao, and V. Dobrokhodov, "Path following for small unmanned aerial vehicles using l1 adaptive augmentation of commercial autopilots," *Journal of guidance, control, and dynamics*, vol. 33, no. 2, pp. 550–564, 2010.

[17] C. Cao and N. Hovakimyan, "Vision-based aerial tracking using intelligent excitation," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 5091–5096, IEEE, 2005.

[18] C. Cao, N. Hovakimyan, and J. Wang, "Intelligent excitation for adaptive control with unknown parameters in reference input," *Automatic Control, IEEE Transactions on*, vol. 52, no. 8, pp. 1525–1532, 2007.

[19] T. Yucelen and A. J. Calise, "Derivative-free model reference adaptive control of a generic transport model," in *AIAA Guidance, Navigation, and Control Conference, Toronto, ON*, 2010.

[20] T. Yucelen and A. J. Calise, "Derivative-free model reference adaptive control," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 933–950, 2011.

[21] G. Chowdhary and E. Johnson, "Concurrent learning for convergence in adaptive control without persistency of excitation," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, pp. 3674–3679, IEEE, 2010.

[22] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *Neural Networks, IEEE Transactions on*, vol. 1, no. 1, pp. 4–27, 1990.

[23] F. Lewis, "Nonlinear network structures for feedback control," *Asian Journal of Control*, vol. 1, no. 4, pp. 205–228, 1999.

[24] A. J. Calise and R. T. Rysdyk, "Nonlinear adaptive flight control using neural networks," *Control Systems, IEEE*, vol. 18, no. 6, pp. 14–25, 1998.

[25] B. S. Kim and A. J. Calise, "Nonlinear flight control using neural networks," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 1, pp. 26–33, 1997.

[26] A. J. Calise, N. Hovakimyan, and M. Idan, "Adaptive output feedback control of nonlinear systems using neural networks," *Automatica*, vol. 37, no. 8, pp. 1201–1211, 2001.

[27] E. N. Johnson, M. A. Turbe, A. D. Wu, S. K. Kannan, and J. C. Neidhoefer, "Flight test results of autonomous fixed-wing uav transitions to and from stationary hover," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference Exhibit, Monterey, CO*, 2006.

[28] G. Chowdhary and E. Johnson, "Adaptive neural network flight control using both current and recorded data," in *AIAA Guidance, Navigation, and Control Conference, AIAA-2007-6505. Hilton Head*, 2007.

[29] G. Chowdhary, T. Wu, M. Cutler, and J. P. How, "Rapid transfer of controllers between uavs using learning-based adaptive control," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 5409–5416, IEEE, 2013.

[30] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[31] G. Chowdhary, H. Kingravi, J. P. How, and P. A. Vela, "Bayesian nonparametric adaptive control using gaussian processes," *IEEE Transactions on Neural Networks*, 2013 (submitted).

[32] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 9, pp. 920–932, 1994.

[33] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural computation*, vol. 14, no. 3, pp. 641–668, 2002.

[34] R. Grande, G. Chowdhary, and J. How, "Experimental validation of bayesian nonparametric adaptive control using gaussian processes," *Journal of Aerospace Information Systems*, 2013 (Submitted).

[35] R. Beard, *Small Unmanned Aircraft Theory and Practice*. Princeton: Princeton University Press, 2012.

[36] R. Nelson, *Flight stability and automatic control*. Boston, Mass: WCB/McGraw Hill, 1998.

[37] E. Lavretsky and K. Wise, *Robust and Adaptive Control: With Aerospace Applications*. Springer, 2012.

[38] G. Cai, *Unmanned rotorcraft systems.* New York: Springer, 2011.

[39] E. Johnson, A. Calise, and J. E. Corban, "A six degree-of-freedom adaptive flight control architecture for trajectory following," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Princeton University Press, 2002.

[40] E. Frazzoli, M. A. Dahleh, and E. Feron, "Trajectory tracking control design for autonomous helicopters using a backstepping algorithm," in *American Control Conference, 2000. Proceedings of the 2000*, vol. 6, pp. 4102–4107, IEEE, 2000.

[41] S. Park, *Avionics and control system development for mid-air rendezvous of two unmanned aerial vehicles.* PhD thesis, Massachusetts Institute of Technology, 2003.

[42] G. J. Ducard, *Fault-tolerant flight control and guidance systems: Practical methods for small unmanned aerial vehicles.* Springer, 2009.

[43] G. Cai, B. M. Chen, T. H. Lee, and M. Dong, "Design and implementation of a hardware-in-the-loop simulation system for small-scale uav helicopters," *Mechatronics*, vol. 19, no. 7, pp. 1057–1066, 2009.

[44] J. M. Goppert, *An adaptable, low cost test-bed for unmanned vehicle systems research.* 2011.

[45] A. Bittar and N. M. de Oliveira, "Central processing unit for an autopilot: Description and hardware-in-the-loop simulation," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 557–574, 2013.

[46] N. Kim, "Improved methods in neural network-based adaptive output feedback control, with applications to flight control," 2003.

[47] H. Chao, Y. Cao, and Y. Chen, "Autopilots for small unmanned aerial vehicles: a survey," *International Journal of Control, Automation and Systems*, vol. 8, no. 1, pp. 36–44, 2010.

[48] P. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Boston: Artech House, 2008.

[49] J. H. Brown and B. Martin, "How fast is fast enough? choosing between xenomai and linux for real-time applications," in *proc. of the 12th Real-Time Linux Workshop (RTLWS12)*, pp. 1–17, 2010.

VITA

Jacob Stockton

Candidate for the Degree of

Master of Science

Thesis: MODULAR AUTOPILOT DESIGN AND DEVELOPMENT FEATURING BAYESIAN NON-PARAMETRIC ADAPTIVE CONTROL

Major Field: Aerospace Engineering

Biographical: Born in Fairfield, OH on September 24th, 1988.

Education:
Completed the requirements for the degree of Master of Science at Oklahoma State University, Stillwater, OK in May 2014.

Completed the requirements for the Bachelor of Science in Aerospace Engineering at Oklahoma State University, Stillwater, OK in May 2011.

Completed the requirements for the Bachelor of Science in Mechanical Engineering at Oklahoma State University, Stillwater, OK in May 2011.

Experience:
Flight Test Instrumentation Engineer, Bell Helicopter 2011-2012

Professional Memberships:
AIAA