SECURITY AND VERIFICATION OF UNMANNED VEHICLES

A Dissertation

Submitted to the Faculty

of

Purdue University

by

James M. Goppert

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Inseok Hwang, Chair

School of Aeronautics and Astronautics

Dr. Dengfeng Sun

School of Aeronautics and Astronautics

Dr. Martin Corless

School of Aeronautics and Astronautics

Dr. Eric Matson

School of Computer and Information Technology

Approved by:

Dr. Wayne Chen

Head of the School Graduate Program

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

SYMBOLS

Scalars

g	acceleration of gravity

Vectors

$\boldsymbol{R}_b^{OP}$	vector from point O to point P, expressed in frame b
${}^b\boldsymbol{V}_c^{OP}$	derivative of vector from point O to point P, w.r.t. frame b, expressed in frame c
${}^n\boldsymbol{V}_b^P$	generic velocity (base point fixed in derivative frame) of point P, w.r.t. frame n, expressed in frame b
$\dot{\boldsymbol{R}}_b^{OP}$	derivative of $\boldsymbol{R}_b^{OP}$, w.r.t. the expressed coordinate frame (frame b)
${}^n\boldsymbol{\omega}_c^b$	angular velocity of frame b, w.r.t. frame n, expressed in frame c

Rotations

$\boldsymbol{q}_{nb}$	quaternion representing rotation $\boldsymbol{R}_n^{OP} = \boldsymbol{q}_{nb} \otimes \boldsymbol{R}_b^{OP} \otimes q_{nb}^{-1}$
C^{nb}	direction cosine matrix representing rotation $\boldsymbol{R}_n^{OP} = C^{nb}\boldsymbol{R}_b^{OP}$

Coordinate Frames

b	body fixed frame, Forward-Right-Down, right-handed
n	navigation frame, fixed in earth at local tangent plane (North-East-Down), assumed inertial

Points

O	origin of local (North-East-Down) frame, frame n
P	origin of body fixed frame, frame b, assumed to be the location of the IMU

Overbars

$\boldsymbol{v}$	true vector
$\hat{\boldsymbol{v}}$	estimate of $\boldsymbol{v}$

Operators and Functions

$\boldsymbol{a} \circ \boldsymbol{b}$	dot product of $\boldsymbol{a}$ and $\boldsymbol{b}$
$\boldsymbol{a} \times \boldsymbol{b}$	cross product of $\boldsymbol{a}$ and $\boldsymbol{b}$
$\boldsymbol{a} \otimes \boldsymbol{b}$	quaternion product of $\boldsymbol{a}$ and $\boldsymbol{b}$
$\mathcal{N}(\mu, \sigma^2)$	normal distribution with mean μ and variance σ^2

ABSTRACT

Goppert, James M. Ph.D., Purdue University, December 2018. Security and Verification of Unmanned Vehicles. Major Professor: Inseok Hwang.

This dissertation investigates vulnerabilities in unmanned vehicles and how to successfully detect and counteract them. As we entrust unmanned vehicles with more responsibilities (e.g. fire-fighting, search and rescue, package delivery), it is crucial to ensure their safe operation. These systems often have not been designed to protect against an intelligent attacker or considering all possible interactions between the physical dynamics and the internal logic. Robust control strategies can verify that the system behaves normally under bounded disturbances, and formal verification methods can check that the system logic operates normally under ideal conditions. However, critical vulnerabilities exist in the intersection of these fields that are addressed in this work. Due to the complex nature of this interaction, only trivial examples have previously been pursued. This work focuses on efficient real-time methods for verification and validation of unmanned vehicles under disturbances and cyberattacks. The efficiency of the verification and validation algorithm is necessary to run it onboard an unmanned vehicle, where it can be used for self diagnosis. We begin with simple linear systems and step to more complex examples with non-linearities. During this progression, new methods are developed to cope with the challenges introduced. We also address how to counter the threat of unmanned aerial systems (UASs) under hostile control by developing and testing an estimation and control scheme for an air-to-air counter UAS system.

1. INTRODUCTION

Unmanned vehicles are susceptible to cyberattacks and logic errors, and yet these vulnerabilities have not been thoroughly investigated. Many unmanned vehicles, including unmanned aerial systems (UASs), rely solely on the encryption of data channels to prevent cyberattacks [1]. While data encryption is a key component of multi-layered security strategy, relying on it as the sole defense against a cyberattack is misguided [2]. The successful cyberattacks in the last decade demonstrate that data encryption is not sufficient. In September of 2011, a malware key-logger program infected UAS command and control at Creech Air Force Base [3]. In a report to Congress in 2011, the U.S.-China Economic and Security Review Commission revealed that the Landsat-7 and Terra EOS AM-1 satellites were controlled by foreign agents through internet connect ground stations [4]. Since the attacker has command and control of the system in these scenarios, communication encryption and authentication do not protect the UAS. Additionally, there are multiple sensor attacks that can be employed by an adversary to corrupt the state of a UAS without needing to break encryption. Examples of these attacks include spoofing GPS or Automatic Dependent Surveillance-Broadcast (ADS-B) signals [5–7]. As commercial UASs become more prevalent, it is also beneficial to guarantee their safe operation. A Russian postal UAS recently crashed during a demonstration and could have harmed those observing the flight [8].

If an attacker were to compromise an unmanned vehicle, the consequences could be disastrous. When an individual unmanned vehicle is compromised, it may fail to complete a potentially vital mission, such as active combat, combat support, military or law enforcement surveillance, fire fighting, or wilderness search and rescue. A compromised unmanned vehicle may also leak intelligence information, as was the case in 2009 when Iraqi militants gained access to live video feeds from US military

UASs [9]. Finally, a compromised unmanned vehicle poses a significant threat to human life and property if the attacker accesses any onboard weapon systems or uses the vehicle itself as a kinetic weapon. These dangers are amplified when multiple vehicles are formed into a network. As more vehicles are added to the network, the number of vulnerabilities that an attacker can exploit increases and the communication links formed between the nodes of this network provide new avenues of attack. With the ability to compromise more vehicles, the attacker can cause more damage than with a single vehicle [10].

Given the many avenues of attack and the high cost of failure, it is important that the security and verification of unmanned vehicles is considered in the design process. However, attempting to analyze unmanned vehicle vulnerabilities is a daunting task. An unmanned vehicle is a cyber-physical system, which is a system characterized by a tight coupling of the logical behavior of a computing system and an underlying physical processes [1, 11]. Ensuring the security of a cyber-physical system requires analysis of the interactions between these digital and physical components. Additionally, each unmanned vehicle design is physically distinct. The mass properties, propulsion system, sensors, actuators, control system, and aerodynamics of the vehicle all contribute to the system dynamics. These unique dynamics determine the system's vulnerabilities to cyberattack, which makes a generalized protection system difficult to implement. A detailed study of unmanned vehicle cyberattack vulnerabilities has not been conducted previously due to the scope of the problem and the lack of a well-defined measure of attack severity. In this work, we address these vulnerabilities of unmanned vehicles by systematically studying their behavior, identifying a measure of attack severity, studying particularly severe attacks, and developing methods of verifying system performance under severe disturbances such as cyberattacks or severe weather.

1.1 Verification and Validation (V&V) of Unmanned Vehicles

Verification and Validation (V&V) of an unmanned vehicle is not a trivial task due to the coupled discrete and continuous dynamics. The interacting discrete state (or mode) transitions and continuous dynamics can be modeled as a hybrid system and thus a hybrid model checker can be used for V&V of an unmanned vehicle. If the system is fully autonomous and the mode transitions are state dependent, then the hybrid system model can be approximated using an approximate quotient transition system (AQTS) that partitions the state space. In AQTSs, the continuous states are handled by computing invariant sets (or flow pipes) for each discrete mode. The transition between flow pipes is governed by state dependent guards. The AQTS creates a finite state machine representation of the original hybrid system so that standard model checkers can be used to verify its properties [12]. If the invariant partitions are represented using convex polyhedrons, then the approximation is called a polyhedral invariant hybrid automaton (PIHA) [13]. The CheckMate tool implements a PIHA based model checker and is integrated with the Matlab Simulink development environment [14]. Unfortunately, CheckMate cannot be efficiently extended to systems with unknown disturbances [15]. The d/dt model checker uses the concept of orthogonal polyhedra and is capable of handling unknown input/disturbances that are bounded by a convex polyhedron but at an increased computational cost [15, 16]. HyTech is one of the first hybrid model checkers but only applies to systems that can be represented by $\dot{x} = Ax < b$, where x is the state vector, A is a time invariant matrix, and b is a constant vector, which does not account for unknown disturbances [17].

We choose to employ the PIHA model due to its speed and the popularity of CheckMate; however, for the problem of an unmanned vehicle system there are several significant differences between systems typically modeled by PIHAs. First, an unmanned vehicle is typically regulated around a nominal trajectory. Second, the vehicle is subject to disturbances. If we naively apply the PIHA analysis without accounting for disturbances, the problem rapidly becomes trivial since the trajectory

approaches the nominal trajectory. Ignoring the disturbances therefore dramatically underestimates the reachable set of the system, and model checking does not identify errors that may exist in the control software.

In this dissertation, we propose augmenting the computed flow pipes of existing PIHA methods for linear systems with the H_∞ norm of the disturbance to the output transfer function or constructing flow pipes using invariant sets from Lyapunov theory. The H_∞ norm is often used in robust control to account for system disturbances and represents the steady-state bound of the worst case sinusoidal disturbance to the system. While this method is not strictly conservative, since transients and non-sinusoidal inputs could exceed the H_∞ norm, the H_∞ norm still gives an adequate bound for practical disturbances and can be used to identify previously undetected errors in hybrid systems. When more computational power is available, a linear matrix inequality (LMI) can be constructed to find a Lyapunov based invariant set for the system. This invariant set can be used to construct a flow pipe that is strictly conservative given the disturbances are bounded. These two approaches are aligned with our objective of developing an efficient real time model checking algorithm for adaptive systems. For example, if a UAS is damaged in flight and still has adequate data to conduct system identification on itself, it could apply this computationally efficient model checking method to ensure it can still complete its mission.

1.2 Polyhedral Invariant Hybrid Automaton (PIHA) based Model Checking

In the section, we discuss the existing methods for V&V of PIHAs [12, 13]. This knowledge is fundamental for understanding the usefulness of the method and how we extended the existing methods to account for disturbances in Section 3.1. The main distinguishing property of a PIHA is that it uses convex polyhedra to describe the invariant sets and transitions. We use the definition of a PIHA from [12]:

Definition 1.2.1 *A PIHA is a tuple* $H = (X, X_0, F, E, I, G)$ *where*

- *$X = X_C \times X_D$, where $X_C \subseteq R^n$ is the continuous state space and X_D is a finite set of discrete locations.*
- *F is a function that assigns to each discrete location $u \in X_D$ a vector field $f_u(\cdot)$ on X_C.*
- *$I : X_D \mapsto 2^{X_C}$ assigns $u \in X_D$ an invariant set of the form $I(u) \subseteq X_C$ where $I(u)$ is a non degenerate convex polyhedron.*
- *$E \subseteq X_D \times X_D$ is a set of discrete transitions.*
- *$G : E \mapsto 2^{X_C}$ assigns to $e = (u, u') \in E$ a guard set that is a union of faces of $I(u)$.*
- *$X_0 \subseteq X$ is the set of initial states of the form $X_0 = \cup_i (P_i, u_i)$ where each $P_i \subseteq I(u_i)$ is a polytope and $u_i \in U$; here, the notation (P, u) means the set $\{(x, u) \in X \mid x \in P\}$.*
- *I, G, and E must satisfy the following covariance requirements:*
  1. *for each u, $\partial I(u) = \cup_{e \in E \mid e=(u,u')} G(e)$ for some $u' \in X_D$, that is the guards for u cover the faces of the invariant for u.*
  2. *for all $e = (u, u') \in E$, $G(e) \subseteq I(u')$, that is events do not lead to transitions that violate invariants.*

1.2.1 Approximate Quotient Transition Systems (AQTS)

A standard approach for V&V of hybrid systems is to create a finite state bisimulation [12]. Bisimulations simulate the original system (have the same state histories given the same input) and can be simulated by the original system. Bisimulations can be created through partitioning the continuous state space into discrete regions and this is known as a quotient transition system (QTS). Unfortunately, it has been shown that finite state bisimulations only exist for hybrid systems with trivial continuous dynamics [18]. However, a QTS is a simulation of the system, meaning that

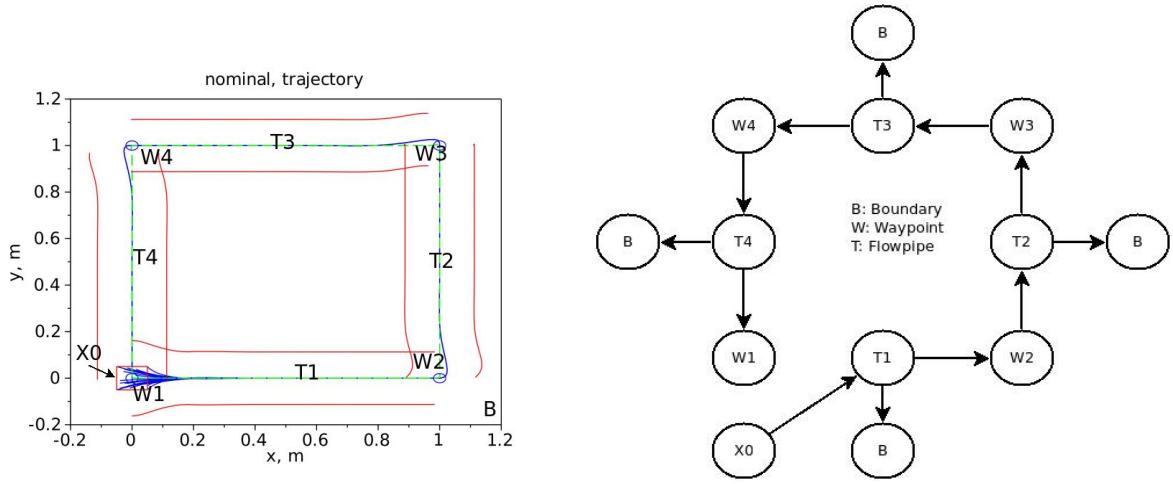
any universal specifications (i.e., specifications that must be true for all possible trajectories) that are true for the QTS are also true for the hybrid system. Universal specifications include specifications such as safety and reachability [12].

In order to construct a QTS, for all partitions the reachable set of the continuous states must be computed. Consequently constructing a precise QTS is only possible for simple systems such as those with clock dynamics [18]. For more complicated continuous dynamics, it is only possible to construct an approximate quotient transition system (AQTS). An AQTS can either over-approximate the reachable set of the continuous system and be used to show that certain states are not reachable (safety), or under-approximate the reachable set and show that certain states are reachable (reachability).

1.2.2 Computation of Flow Pipes

The reachable set for a convex partition of an autonomous nonlinear systems can be computed by propagating the boundaries of the convex partition. For linear systems, this can be accomplished more efficiently by using a cached propagation shape for the reachable set that can be scaled as time progresses [12].

In Figure 1.1, we show how the hybrid system can be converted to an AQTS that is capable of verifying safety and reachability specifications. A flapping wing system moves from the square initial set X_0 at waypoint 1, $W1$, to flow pipe 1, $T1$, to waypoint 2, $W2$, to flow pipe 2, $T2$, to waypoint 3, $W3$, to flow pipe 3, $T3$, and finally to waypoint 4, $W4$. From the initial set, several trajectories are propagated using Monte-Carlo simulation (shown in blue). The existing flow pipe computation would enclose the trajectories originating from X_0. As can be seen, for the tracking problem shown in Figure 1.1(a), the blue trajectories rapidly collapse to the green reference trajectory. The red outer bounds shown in the plot are computed by our method and represent the bounds considering unknown disturbances which we discuss in Section 3.1 in detail.



(a) Nominal Trajectory

(b) Radius Boundary

Fig. 1.1. The generation of an AQTS from computed flow pipes. Note that the green line is the nominal trajectory, the blue lines are Monte-Carlo simulations starting from the initial set X_0, and the red bounds are constructed using the H_∞ norm which we discuss in Section 3.1.

1.3 Structure of Dissertation

In Chapter 2, we investigate vulnerabilities of an autopilot system for UAS. In Chapter 3, an H_∞ norm verification method is applied to an insect-size flapping wing robot whose dynamics are simple and can be approximated as linear after dynamic inversion. This is followed by Chapter 4, where a more sophisticated Lyapunov based method is applied to a non-linear two-wheel self-balancing robot. Finally, Chapter 5 discusses design of autopilot estimation algorithms for Verification and Validation. Due to the inherent non-linearities in rigid body rotation, we leverage Lie group theory to construct an invariant set in the Lie algebra. The invariant set in the Lie algebra is then used to construct a flow pipe in the Lie Group and allow application of the PIHA method for model checking as discussed in Chapter 3. Finally in Chapter 6, we address the scenario of a UAS under hostile control. We develop a simple estimation

and control scheme for a hunter UAS that successfully intercepts a hostile UAS during flight testing.

2. NUMERICAL CYBERATTACK ANALYSIS

Unmanned Aerial Systems (UASs) currently assist in many applications, including surveillance, law enforcement, and military missions. The unmanned nature of these systems and the subsequent lack of direct monitoring leaves them vulnerable to cyber-attacks that can jeopardize the mission, the vehicle, and potentially lives and property. Given the severity of such attacks, it is important to assess the vulnerabilities inherent in these cyber-physical systems, complex systems composed of both a monitoring computer (e.g. autopilot of a UAS) and physical components. In this chapter, we propose a method for identifying cyberattack vulnerabilities in a UAS, assuming that the attacker has full access to the system in order to identify worst-case scenarios. We establish the time till failure metric of the system as a measure of attack severity. We categorize intents and outcomes for typical attacks and perform simulations to estimate the severity of attack combinations. This resulting vulnerability analysis can be utilized in the UAS design process to address weaknesses. When combined with traditional cybersecurity analysis, this comprises a comprehensive computer science and control systems approach to the security of cyber-physical systems. This chapter is based on the journal article [19].

A traditional computer science approach to UAS cybersecurity focuses on identifying ways in which an attacker can gain access to the system. In this chapter, we consider the problem of UAS cybersecurity from a control systems perspective. We presume that an attacker has already compromised the system and has the ability to make arbitrary modifications. We seek to identify which modifications this privileged attacker would make in order to have the greatest negative impact on the UAS. Identifying these vulnerabilities allows designers to create a more robust system. Applying this strategy along with the traditional cybersecurity analysis results in a system that both denies attackers access and restricts what they can accomplish

with that access should they obtain it. This comprehensive approach represents a coupling of computer safety and control system safety in much the same way that cyber-physical systems couple the discrete dynamics of computing systems with the continuous dynamics of a physical system.

This controls domain approach to securing UASs complements other methodologies, including robust control approaches to identifying system sensitivities [20]. The benefit of our numerical approach is that it allows easy and rapid analysis of different attacks that may be difficult to model or may violate key assumptions, e.g. that noise is Gaussian. The assumptions required in analytical approaches, e.g. that the attacked system is linear and time invariant, limit the analysis to a subset of possible attacks on idealized systems. [21–27] Accordingly, these approaches do not extend well to the high complexity and non-linearity of full UAS models. The proposed approach does not require simplifying assumptions, and is rather designed to analyze the complete system, allowing the effective and efficient analysis of UAS vulnerabilities to a wide range of cyberattacks. Hardware testing also allows for the system to be simulated as implemented, incorporating all of the underlying discrete dynamics of the system that may be lost in the abstractions of the system used in other analysis. Due to variation in cyber-physical systems, the results of a single analysis will not generalize well. Accordingly, we have designed our testbed to easily accommodate new UAS models.

The contributions of this chapter are as follows:

- In Section 2.1, we propose a metric for attack severity. This metric, time till failure, is the amount of time the system operates until the failure criteria are met.

- In Section 2.2, we introduce the UAS subsystem models we created. These models were chosen to be representative of a typical UAS.

- In Section 2.3, we perform the integration of UAS subsystems into a complete software model of a typical UAS cyber-physical system. This model employs

JSBSim, a C++ flight dynamics model library, to model the aircraft dynamics. It utilizes the Scicos block diagram environment to model the control, guidance, and navigation systems as well as cyberattacks. We also show the response of the system to selected cyberattacks.

- In Section 2.4, we describe the hardware-in-the-loop (HIL) UAS testbed we created. This testbed uses an open source hardware and software autopilot that implements the UAS subsystems along with an open source ground control station with JSBSim modeling the aircraft dynamics. This allows for simulation of cyberattacks using the actual UAS hardware and software. The response of the system to selected cyberattacks is shown, along with a comparison to the software-in-the-loop (SIL) results.

The methodology and tools presented are easily adaptable to a variety of system models, and represent a suitable means of identifying cybersecurity vulnerabilities in a control system. The analysis presented can be combined with traditional cybersecurity analysis [28–32] to give a comprehensive vehicle vulnerability analysis encompassing both the computer science and control systems domain.

2.1 Cyberattack Measures

2.1.1 Attack Intent Classification

In this analysis, we investigate vulnerabilities of typical UASs. We classify potential cyberattacks by the attacker's intent in order to design the parameters used to quantify the severity of the attack. The three intents we discuss are mission obstruction, control acquisition, and vehicle destruction. Modeling and extensively testing a representative system allow us to draw conclusions on the relative effectiveness of these attacks.

- **Mission Obstruction**

In a mission obstruction attack, the attacker aims to prevent the UAS from completing the assigned mission objectives. This can be accomplished using several methods. For example, the vehicle can be delayed such that the time requirement of the mission is not met, or the vehicle can be forced to waste fuel or battery power so that the mission objectives are no longer feasible. Unpredictable errors could also be inserted into the navigation system in order to degrade the state awareness of the vehicle. Another possibility is that the control system could be corrupted to the point that its sensors begin to perform poorly, introducing issues such as highly oscillatory motion. One example we consider leverages the vehicle's collision avoidance system to obstruct the vehicle. By inserting a phantom vehicle in the path of the target vehicle, an attacker can cause the target vehicle to perturb its flight path in order to avoid collision. It should be noted that in a mission obstruction attack, the attacker does not have the ability to control the vehicle directly or does not attempt to do so. If the attacker can control the system directly, it would be considered a control acquisition attack.

- **Control Acquisition**

  In a control acquisition attack, the objective of the attacker is to assume direct control of the vehicle. An example of this would be the use of GPS spoofing to shift the flight path of the UAS to suit the purposes of an attacker. For this type of attack, it may be possible for an attacker to have differing levels of control, e.g. an attacker may be able to gain control of vehicle subsystems without gaining control of the entire vehicle. If the attacker is able to gain complete control of a vehicle, there is a possibility of a man-in-the-middle attack. In this attack, the attacker would send falsified data to the original controller to make it appear that the vehicle is behaving normally, when it is actually being controlled by the attacker. Such an undetectable attack is especially dangerous.

- **Vehicle Destruction**

In some cases, the attacker's intent may be simply to destroy the vehicle. It is possible for an attacker to have sufficiently limited control over one state that they cannot perform a meaningful control acquisition attack, but still have the capability to destroy the vehicle. For instance, if the attacker has control of the altitude of the vehicle they may command the aircraft to fly into the ground. However, the primary area of danger, and thus the focus of this analysis for vehicle destruction attacks, will be the introduction of instability into the control and navigation system of the vehicle. An instability in these critical systems will result in a crash.

2.1.2 Failure Criteria

To evaluate whether a cyberattack has been successful, we must establish a criterion for failure. Based on the attack intents described above, we identify two failure modes, described below. In order to quantify the severity of an attack, we use the time elapsed when any of these failure criteria were met, referred to as time till failure.

- **Flight Envelope Failure**

  Flight envelope failure is defined as failure of the vehicle airframe, which typically leads to destruction of the vehicle. The failure criteria is quantified using the angular rates of the vehicle, which are a rough approximation of the wing loading that would cause structural failure. Currently, 33% of all UAS system failures are caused by the UAS exceeding its designed flight envelope [33]. UAS flight envelope enforcement is a strategy for fail-safe recovery that can be used to mitigate such attacks [33–35].

- **Mission Envelope Failure**

  Mission envelope failure is defined as violating the stated mission requirements of the UAS. The UAS state can be compared to the mission requirements to determine if the mission envelope has been violated. For this study, we selected

mission requirements representative of a UAS reconnaissance mission, which should be similar to the requirements for other mission types. The mission envelope parameters identified in this study are summarized in Table 2.1.

Table 2.1.
Mission Envelope Parameters

Mission theater	Geographic region to which the UAS is confined
Altitude window	Range of acceptable vehicle altitudes
Battery/fuel level	Required reserve battery power and fuel
Target window	Area vehicle must reach
Target time window	Duration vehicle must be inside the target window
Flight envelope	Flight envelope to avoid airframe failure

2.2 System Model

A typical UAS is composed of many interconnected components as shown in Figure 2.1. The complex nature of these interconnections makes the system difficult to model. Analytical approaches to analyzing UASs often require abstractions, such as no jitter or latency in system events and timing or Gaussian noise, in order to make the problem tractable. While these approaches can provide valuable insight, these abstractions limit the accuracy of the analysis and they only protect against attacks that can be included in the model. In our numerical approach, we do not restrict the complexity of the system, and the effects of all types of cyberattacks can be determined through the careful observation of the output of the system in response to different stimuli. In this section, we create a model of an example UAS to better illustrate the proposed method. However, the proposed cyberattack analysis method is general enough to be applied to various types of UASs. This section focuses on the systems specific to the Purdue Hybrid Systems Lab software simulation testbed

and there are commonalities with the hardware-in-the-loop configuration presented in Section 2.4.

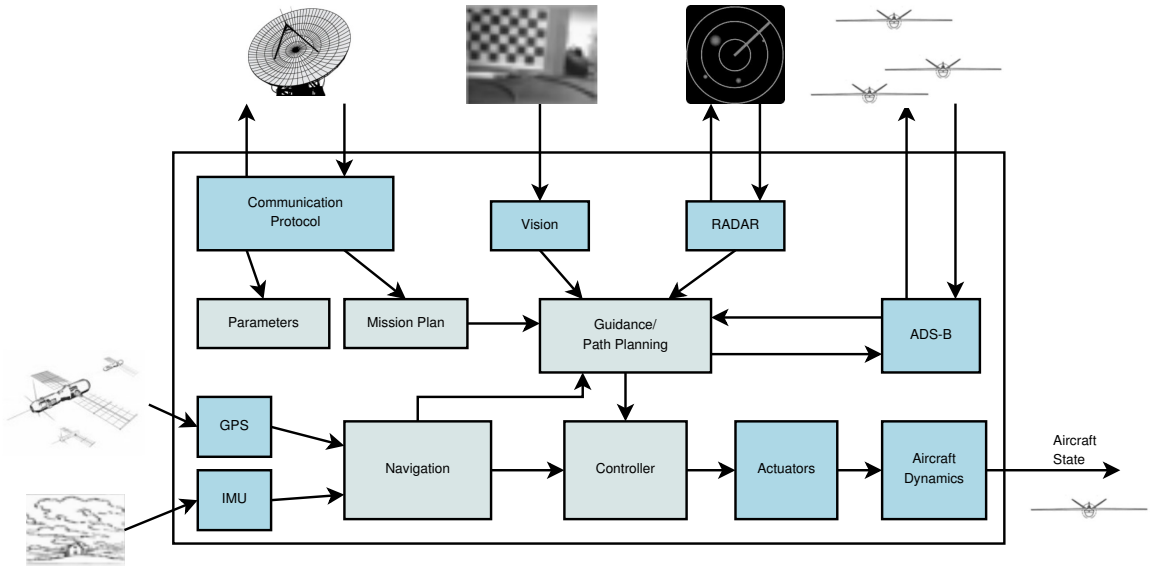


Fig. 2.1. Typical UAS components.

2.2.1 Aircraft

For this chapter, we chose to analyze the MultiPlex Easy Star, shown in Figure 2.2, which is widely used for UAS research. It is stable, inexpensive, and durable. For control surfaces, it has an elevator and rudder on the tail but no ailerons on the wing. Although we selected the Easy Star as an example in this chapter, other UASs can be evaluated using the same approach presented in this chapter. We created a model of the vehicle dynamics using a combination of data from the Boeing wind tunnel at Purdue and the USAF Digital DATCOM software [36, 37]. We simulate this model using JSBSim,[1] which is an open source flight dynamics model written in C++. It uses XML descriptions of an aircraft, complete with look-up tables for aerodynamic and propulsion data, in order to accurately simulate flight. JSBSim

[1] "JSBSim Open Source Flight Dynamics Model," http://jsbsim.sourceforge.net/index.html

is utilized by various open source flight simulators, such as FlightGear,[2] and is also used to drive the motion-based research simulators at the University of Naples, Italy, and in the Institute of Flight System Dynamics and the Institute of Aeronautics and Astronautics at RWTH Aachen University, Germany.

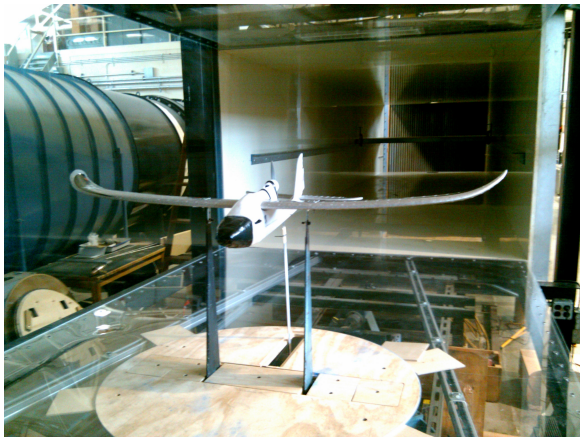


Fig. 2.2. The MultiPlex Easy Star airframe in the Boeing Wind Tunnel at Purdue University.

2.2.2 Controller

A digital PID controller is used to generate the control signals for the actuators. This vehicle uses a backside control strategy, in which the elevator is used to control the velocity and the throttle is used to control the altitude. The elevator controls velocity by controlling angle of attack, allowing this strategy to work at both high and low flight speeds. In contrast, a frontside control strategy uses the throttle for velocity and the elevator for altitude. This approach is more complicated to implement in a control system due to a gain reversal on the back side of the power required curve, requiring more throttle to go slower [38].

The controller model is shown in Figure 2.3. In this model, the altitude, velocity, yaw rate, bearing, and bank angle command error are fed into a bank of digital

[2] "FlightGear Flight Simulator," http://www.flightgear.org

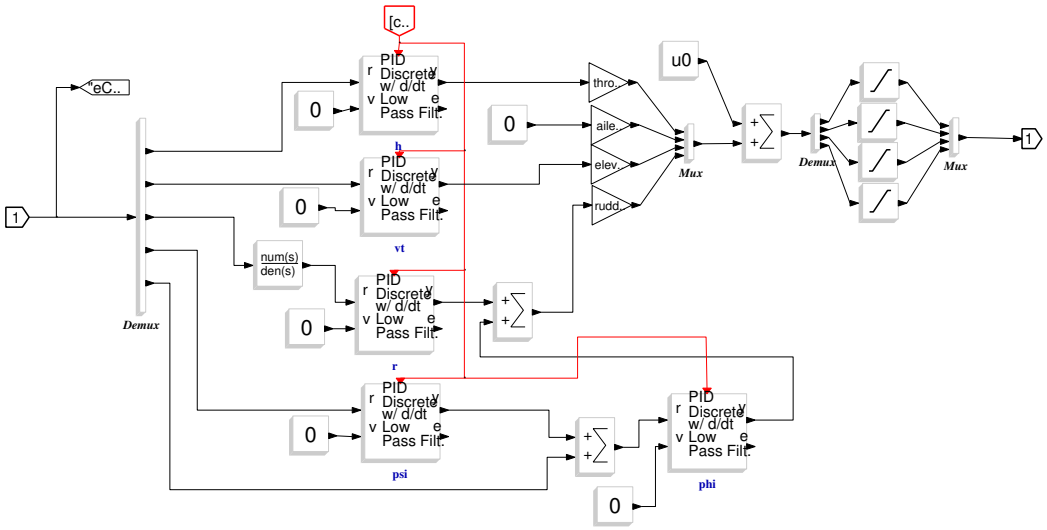


Fig. 2.3. The digital controller model in ScicosLab.

PID controllers. Because there are no ailerons on the Easystar, the output of the PID controller for the yaw rate, bearing, and bank angle are multiplexed into the single rudder input. The controller outputs are then added to the trim values for each actuator to give the commanded position for each actuator. These commanded positions are then passed through saturation blocks to limit their value to within the acceptable range for the design before being sent to the actuators.

2.2.3 Navigation System

The navigation system is responsible for using noisy sensor measurements to generate an estimate of the UAS's current state, including position, velocity, and attitude. This is done using the integration of accelerometer and gyroscope measurements to provide a state estimate that is periodically corrected using GPS and magnetometer measurements. The navigation system must interface with external sensors and the environment, creating potential avenues of attack. Additionally, it can be difficult to validate the measurements that are made due to the high cost of redundant sensors

and the noise present in those measurements, creating the possibility of an attack that cannot be detected. The degraded state awareness that would result from a successful attack on the navigation system can also have serious ramifications for mission objectives and vehicle safety. Accordingly, ensuring that the navigation system is resilient to cyberattack is of particular importance.

In order to evaluate the effects of malicious states changes in the navigation system, it is necessary to carefully construct a typical GPS/INS navigation system model. The dynamics of the inertial navigation system have to be derived as well as the models for sensor measurements. These nonlinear models must then be linearized in order to apply the typical Extended Kalman Filter (EKF) navigation methods.

Navigator Dynamics

The navigator state consists of the quaternions, the velocity in the navigation frame, and the global position. Here we denote the quaternions as a, b, c, d, the north velocity as V_N, the east velocity as V_E, the downward velocity as V_D, the latitude as L, the longitude as l, and the altitude as h. The full navigation state is given by $\mathbf{x} := [a\ b\ c\ d\ V_N\ V_E\ V_D\ L\ l\ h]^T$. The navigator input vector consists of the measured body angular rates, ω_x, ω_y, ω_z, and the measured accelerations, f_x, f_y, f_z. Here x, y, z, represent a right handed coordinate frame fixed in the body with the x axis pointing forward and the y axis pointing out the right wing. Thus, the input to the navigator is given by: $\mathbf{z}_{IMU} := [\omega_x\ \omega_y\ \omega_z\ f_x\ f_y\ f_z]^T$. Taking into account the rotational velocity of the Earth, Ω, and the distance from the center of the Earth R, the dynamics of the navigator are given by:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{z}_{IMU}(t)) \tag{2.1}$$

The non-linear function $\mathbf{f}$ is defined in (2.2), where $\mathbf{C}^{nb}$ is the direction cosine matrix from the body to the navigation frame, $\mathbf{q}_{nb}$ is the attitude quaternion from the body to the navigation frame, $\boldsymbol{\omega}^{nb} := [P\ Q\ R]^T$ is the angular velocity of the body frame with respect to the navigation frame, $\boldsymbol{\alpha}^b$ is the acceleration measured in the body

frame, $\boldsymbol{\omega}^{ie}$ is the angular velocity of the earth with respect to the inertial frame, $\boldsymbol{\omega}^{en}$ is the angular velocity of the navigation frame with respect to the earth frame, $\mathbf{g}^n$ is the acceleration of gravity expressed in the navigation frame, R_E is the radius of earth, $\times$ is the vector cross product, and $V_N\ V_E\ V_D$ are the components of the velocity in the navigation frame [39].

$$\mathbf{f}(\mathbf{x}(t), \mathbf{z}_{IMU}(t)) = \begin{pmatrix} \frac{1}{2}\begin{pmatrix} 0 & -P & -Q & -R \\ P & 0 & R & -Q \\ Q & -R & 0 & P \\ R & Q & -P & 0 \end{pmatrix} \mathbf{q}_{nb} \\ \mathbf{C}^{nb}\boldsymbol{\alpha}^b - (2\boldsymbol{\omega}^{ie} + \boldsymbol{\omega}^{en}) \times \mathbf{v}^n + \mathbf{g}^n \\ \frac{V_N}{R_E} \\ \frac{V_E}{\cos(L)\, R_E} \\ -V_D \end{pmatrix} \tag{2.2}$$

GPS measurements

The GPS sensor measures the velocity and position related to the navigator's states. Let $\mathbf{z}_{GPS}$ be the GPS measurement:

$$\mathbf{z}_{GPS} = \mathbf{h}_{GPS}(\mathbf{x}) + \mathbf{v}_{GPS} = \begin{bmatrix} V_N & V_E & V_D & L & l & h \end{bmatrix}^T + \mathbf{v}_{GPS} \tag{2.3}$$

where $\mathbf{v}_{GPS}$ is the zero-mean white Gaussian measurement noise with deterministic covariance. Usually, the GPS performance is given by the estimation errors of velocity, position and altitude measurement, which are denoted by the variance of Gaussian distributions: σ_V^2, σ_P^2 and σ_A^2. Then, the covariance of the white noise can be calculated via:

$$\mathbb{E}[\mathbf{v}_{GPS}\ \mathbf{v}_{GPS}^T] := \mathbf{R}_{GPS} = diag\left(\sigma_V^2, \sigma_V^2, \sigma_V^2, \frac{\sigma_P^2}{R^2}, \frac{\sigma_P^2}{cos(L)^2 R^2}, \sigma_A^2\right) \tag{2.4}$$

IMU measurements

The IMU provides noisy measurements for the input U to the navigator dynamics. Let $\mathbf{z}_{IMU}$ be the IMU measurement:

$$\mathbf{z}_{IMU} = \begin{bmatrix} \omega_x & \omega_y & \omega_z & f_x & f_y & f_z \end{bmatrix}^T + \mathbf{v}_{IMU} \tag{2.5}$$

where $\mathbf{v}_{IMU}$ is the zero-mean white Gaussian noise whose covariance $\mathbf{R}_{IMU}$ is given by the product data-sheet of the IMU.

Magnetometer measurements

To derive the non-linear observation model for the magnetometer, we need to compute the local magnetic field. The National Oceanic and Atmospheric Administration (NOAA) provides a software library that gives the magnetic inclination M_I, and the declination M_D of the magnetic field lines at the current location.[3] Expressed in the north-east-down frame the magnetic field vector $[B_N \; B_E \; B_D]^T$ is given by:

$$\begin{bmatrix} B_N \\ B_E \\ B_D \end{bmatrix} = \begin{bmatrix} \cos(M_I)\cos(M_D) \\ \sin(M_I)\cos(M_D) \\ \sin(M_D) \end{bmatrix} \tag{2.6}$$

Using the quaternion rotation to rotate this magnetic field unit vector into the frame of the aircraft, the non-linear observation model of the magnetometer is given by:

$$\mathbf{z}_{mag} = \mathbf{h}_{mag}(\mathbf{x}) + \mathbf{v}_{mag} \tag{2.7}$$

where

$$\mathbf{h}_{mag}(\mathbf{x}) := \begin{bmatrix} B_N\,(-d^2 - c^2 + b^2 + a^2) + B_D\,(2\,b\,d - 2\,a\,c) + B_E\,(2\,a\,d + 2\,b\,c) \\ B_E\,(-d^2 + c^2 - b^2 + a^2) + B_D\,(2\,c\,d + 2\,a\,b) + B_N\,(2\,b\,c - 2\,a\,d) \\ B_D\,(d^2 - c^2 - b^2 + a^2) + B_E\,(2\,c\,d - 2\,a\,b) + B_N\,(2\,b\,d + 2\,a\,c) \end{bmatrix} \tag{2.8}$$

[3] NOAA, "The World Magnetic Model", http://www.ngdc.gov/geomag/WMM/soft.shtml

and $\mathbf{v}_{mag}$ is the zero-mean white Gaussian noise with covariance $\mathbf{R}_{mag}$ specified by the magnetometer manufacturer.

The block diagram of the magnetometer is shown in Figure 2.4. This block uses the true UAS position and the NOAA library contained in the geoMag block to compute M_I and M_D with additive noise. The quaternion representation of the true UAS attitude, converted to a direction cosine matrix in the euler2DCM block, is then used to rotate the calculated magnetic field line into the frame of the aircraft to provide simulated measurements.

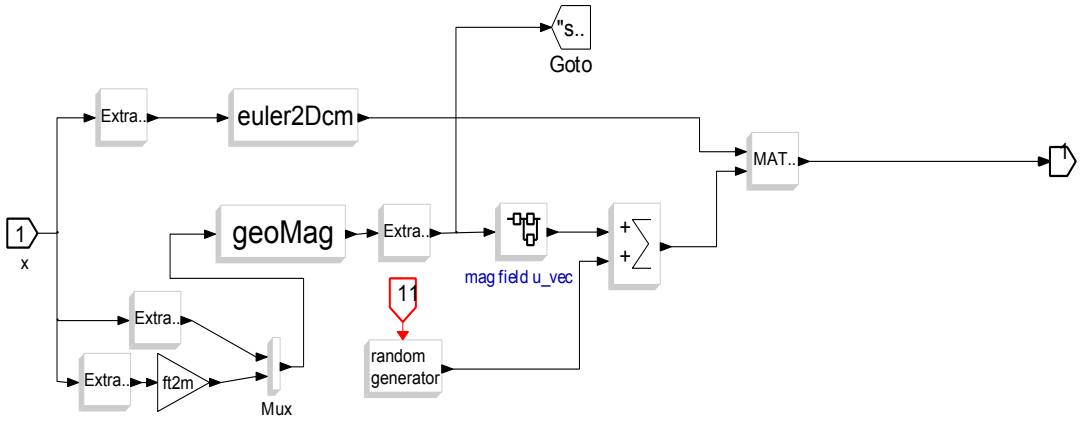


Fig. 2.4. The magnetometer model in ScicosLab.

Extended Kalman Filter (EKF)

Equations (2.1), (2.3), (2.5) and (2.7) yield the navigator system model:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}\left(\mathbf{x}(t), \mathbf{z}_{IMU}(t)\right) \\ \mathbf{z} &= \begin{bmatrix} \mathbf{z}_{GPS} \\ \mathbf{z}_{mag} \end{bmatrix} = \mathbf{h}\left(\mathbf{x}\right) + \mathbf{v} \end{aligned} \tag{2.9}$$

where $\mathbf{h}\left(\mathbf{x}\right) = \left[\mathbf{h}_{GPS}\left(\mathbf{x}\right)\ \mathbf{h}_{mag}\left(\mathbf{x}\right)\right]^T$ and $\mathbf{v} = \left[\mathbf{v}_{GPS}\ \mathbf{v}_{mag}\right]^T$ with the covariance matrix $\mathbf{R} = diag\left(\mathbf{R}_{GPS}, \mathbf{R}_{mag}\right)$. With this model, the EKF can be applied to perform state

estimation for the navigator. Let $\hat{\mathbf{x}}$ be the state estimation and $\mathbf{P}$ be the estimate covariance. The algorithm consists of two steps: propagation and correction which are detailed as follows:

- **Propagation:** The propagation involves computing the evolution of the prior estimate $\hat{\mathbf{x}}(t_{k-1}) = \hat{\mathbf{x}}^{k-1|k-1}$ and $\mathbf{P}(t_{k-1}) = \mathbf{P}^{k-1|k-1}$ when there is no arrival of new observations. Let Δt be the interpolation time interval of the continuous dynamics in Equation (2.9). The navigator's state estimate and its covariance can be updated using Equations (2.10) and (2.11) during the time interval from $(k-1)\Delta t$ till $k\Delta t$ [40]:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}\left(\hat{\mathbf{x}}(t), \mathbf{z}_{IMU}^{k-1}\right) \tag{2.10}$$

$$\dot{\mathbf{P}}(t) = \mathbf{F}\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}^T + \mathbf{G}\mathbf{R}_{IMU}\mathbf{G}^T \tag{2.11}$$

  where $\mathbf{z}_{IMU}^{k-1}$ is the last measurement from the IMU and $\mathbf{F} \equiv \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}^{k-1|k-1}, \mathbf{z}_{IMU}=\mathbf{z}_{IMU}^{k-1}}$ is the Jacobian matrix of $\mathbf{f}(\mathbf{x}, \mathbf{z}_{IMU})$ evaluated at $\hat{\mathbf{x}}^{k-1|k-1}, \mathbf{z}_{IMU}^{k-1}$ and $\mathbf{G} \equiv \left.\frac{\partial \mathbf{f}}{\partial \mathbf{z}_{IMU}}\right|_{\mathbf{x}=\hat{\mathbf{x}}^{k-1|k-1}, \mathbf{z}_{IMU}=\mathbf{z}_{IMU}^{k-1}}$ is the Jacobian matrix of $\mathbf{f}(\mathbf{x}, \mathbf{z}_{IMU})$ evaluated at $\hat{\mathbf{x}}^{k-1|k-1}, \mathbf{z}_{IMU}^{k-1}$.

- **Correction:** The correction step computes the posterior estimate $\hat{\mathbf{x}}^{k|k}$ and $\mathbf{P}^{k|k}$ from the prior estimate $\hat{\mathbf{x}}^{k|k-1} = \hat{\mathbf{x}}(t_k)$ and $\mathbf{P}^{k|k-1} = \mathbf{P}(t_k)$ by using the measurements from the GPS and the magnetometer. Since the GPS and magnetometer measurements are updated at different frequencies (GPS: 10 Hz, magnetometer: 50 Hz), the correction steps may happen at different times. If the cross-correlation between the position and attitude states is neglected, a more computationally efficient algorithm is obtained. This requires two separate EKFs where $\mathbf{x} := [\mathbf{x}_{att}\ \mathbf{x}_{pos}]$ and $\mathbf{P} \approx diag(\mathbf{P}_{att}, \mathbf{P}_{pos})$. Here $\mathbf{x}_{att} := [a\ b\ c\ d]^T$ and $\mathbf{x}_{pos} := [V_N\ V_E\ V_D\ L\ l\ h]^T$. This is a typical approximation in UASs [41]. Under

this decomposition, the following equations can be used for GPS measurement correction:

$$\begin{aligned}
\mathbf{S}_{GPS} &= \mathbf{H}_{GPS}\mathbf{P}_{pos}^{k-1|k-1}\mathbf{H}_{GPS}^{T} + \mathbf{R}_{GPS} \\
\mathbf{K}_{GPS} &= \mathbf{P}_{pos}^{k-1|k-1}\mathbf{H}_{GPS}^{T}\mathbf{S}_{GPS}^{-1} \\
\mathbf{y}_{GPS} &= \mathbf{z}_{GPS} - \mathbf{H}_{GPS}\hat{\mathbf{x}}_{pos}^{k-1|k-1} \\
\hat{\mathbf{x}}_{pos}^{k|k} &= \hat{\mathbf{x}}_{pos}^{k-1|k-1} + \mathbf{K}_{GPS}\mathbf{y}_{GPS} \\
\mathbf{P}_{pos}^{k|k} &= \left[\mathbf{I} - \mathbf{K}_{GPS}\mathbf{H}_{GPS}\right]\mathbf{P}_{pos}^{k-1|k-1}
\end{aligned} \tag{2.12}$$

where $\mathbf{H}_{GPS} = \left.\frac{\partial \mathbf{h}_{GPS}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}^{k-1|k-1}}$ is the Jacobian matrix of $\mathbf{h}_{GPS}$ evaluated at $\hat{\mathbf{x}}^{k-1|k-1}$.

During the arrival of magnetometer measurements, the following equations can be used for correction:

$$\begin{aligned}
\mathbf{S}_{mag} &= \mathbf{H}_{mag}\mathbf{P}_{att}^{k-1|k-1}\mathbf{H}_{mag}^{T} + \mathbf{R}_{mag} \\
\mathbf{K}_{mag} &= \mathbf{P}_{att}^{k-1|k-1}\mathbf{H}_{mag}\mathbf{S}_{mag}^{-1} \\
\mathbf{y}_{mag} &= \mathbf{z}_{mag} - \mathbf{H}_{mag}\hat{\mathbf{x}}_{att}^{k-1|k-1} \\
\hat{\mathbf{x}}_{att}^{k|k} &= \hat{\mathbf{x}}_{att}^{k-1|k-1} + \mathbf{K}_{mag}\mathbf{y}_{mag} \\
\mathbf{P}_{att}^{k|k-1} &= \left[\mathbf{I} - \mathbf{K}_{mag}\mathbf{H}_{mag}\right]\mathbf{P}_{att}^{k-1|k-1}
\end{aligned} \tag{2.13}$$

where $\mathbf{H}_{mag} = \left.\frac{\partial \mathbf{h}_{mag}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}^{k-1|k-1}}$ is the Jacobian matrix of $\mathbf{h}_{mag}$ evaluated at $\hat{\mathbf{x}}^{k-1|k-1}$.

In the navigation system used for this analysis, one EKF is nested within another as shown in Figure 2.5. The inner EKF fuses the attitude measurement data from the magnetometer with the integrated gyroscope rates from the inertial measurement unit (IMU). Outside this loop, the other EKF fuses the GPS position and velocity information with the integrated accelerations from the IMU. Note that the directions of these accelerations strongly depend on the attitude estimate of the inner loop.

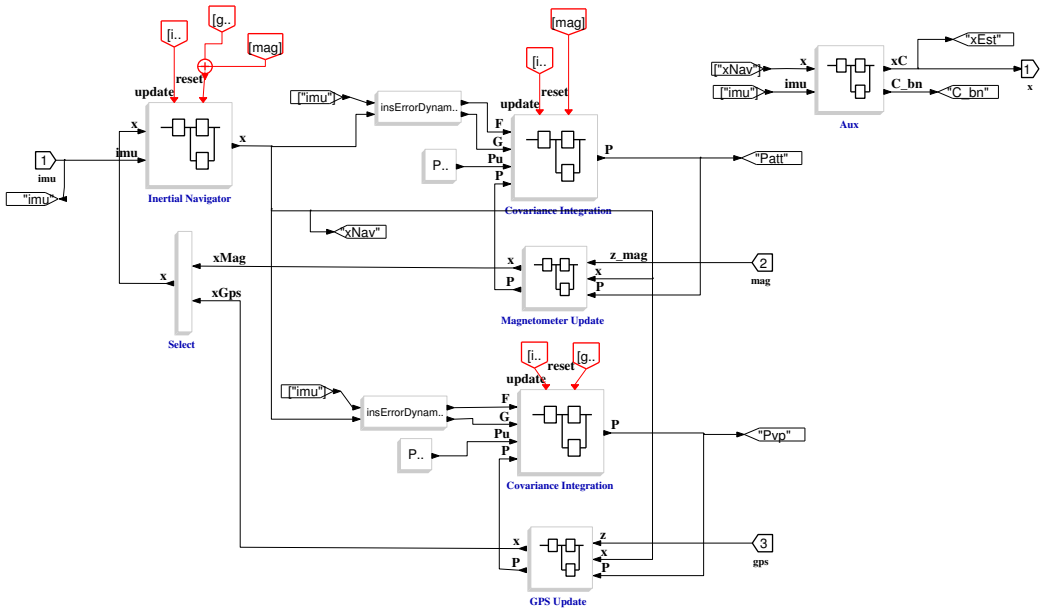


Fig. 2.5. The EKF based navigation system.

2.2.4 Fault Detection

Fault detection systems are used on UASs to determine when a failure has occurred so that corrective action can be attempted. While many of these systems were designed to detect hardware failures, they can also be used to detect abnormal behavior caused by a cyberattack. For this system, fault detection is performed by testing the assumption that the Kalman filter residuals, $\mathbf{y}_{GPS}$ and $\mathbf{y}_{mag}$, are normally distributed with a known covariance. Specifically, a fault can be detected by testing the contradictory hypothesis shown in Equations (2.14) and (2.15).

$$\mathcal{H}_0 : \mathbf{y}_{mag} \sim \mathcal{N}(0, \mathbf{S}_{mag}) \text{ and } \mathcal{H}_1 : \mathbf{y}_{mag} \not\sim \mathcal{N}(0, \mathbf{S}_{mag}) \tag{2.14}$$

$$\mathcal{H}_2 : \mathbf{y}_{GPS} \sim \mathcal{N}(0, \mathbf{S}_{GPS}) \text{ and } \mathcal{H}_3 : \mathbf{y}_{GPS} \not\sim \mathcal{N}(0, \mathbf{S}_{GPS}) \tag{2.15}$$

If $\mathcal{H}_0$ and $\mathcal{H}_2$ are true, then the residual powers, $\mathbf{y}_{mag}^T \mathbf{S}_{mag}^{-1} \mathbf{y}_{mag}$ and $\mathbf{y}_{GPS}^T \mathbf{S}_{GPS}^{-1} \mathbf{y}_{GPS}$, will have a χ^2 distribution. These hypotheses can therefore be tested by comparing the value of these residual powers to a threshold value, τ, which can be calculated

to provide an acceptable accuracy or can be determined experimentally. If either of these hypotheses is rejected, then the corresponding opposing hypothesis, $\mathcal{H}_1$ or $\mathcal{H}_3$, is accepted. This indicates that the measured state evolution does not conform to expectations, and a fault state is detected [42, 43]. For this system, there are limited resources for fault identification or mitigation, so when a fault state is triggered, the system reverts to the constant, open-loop trim inputs in an attempt to maintain normal flight. At that point, the operator can be signaled to intervene and attempt to restore operations.

2.2.5 ADS-B Modeling

ADS-B stands for Automatic Dependent Surveillance-Broadcast. This is a method of sharing data among aircraft in a vicinity through mutual information broadcasts [6]. A major use of ADS-B is the broadcast of navigation information so that neighboring aircraft are able to maintain safety, including minimum separation distances, in crowded airspace. ADS-B is not used in the hardware-in-the-loop testing.

Data Packet

In this analysis, we focus on the navigation information in the ADS-B broadcast. This includes the position and velocity of the aircraft. In the future, aircraft intent will also be included. Other services exist for weather, terrain, and general flight information.

Table 2.2.
ADS-B Packet Information

Position	latitude, longitude, altitude
Velocity	north, east and down velocities
Time Stamp	date and time of broadcast

Collision Avoidance Algorithm

Simple collision avoidance was achieved using the velocity of the vehicle relative to the obstacle to move the obstacle into a static reference frame. Then the only requirement for avoiding collision is that the relative velocity be shifted such that it will not violate the separation distance [44]. The variables used for these calculations are defined in Table 2.3.

Table 2.3.
Collision Variables

Variable	Physical Meaning	Units
Ψ_c	Bearing to obstacle from vehicle	rad
$\Psi_{v/o}$	Bearing of vehicle commanded velocity relative to obstacle	rad
Ψ_r	Required $\Psi_{v/o}$ to maintain separation	rad
r_c	Distance to obstacle	ft
r_s	Separation distance	ft
α	Difference between $\Psi_{v/o}$ and Ψ_c	rad
β	Magnitude difference in bearing between a collision course and a course tangent to the separation window	rad
γ	Change in $\Psi_{v/o}$ required for vehicle path to be tangent to separation window	rad

The initial commanded bearing, Ψ_v will always be chosen to point directly towards the commanded waypoint. If this bearing will cause the vehicle to violate the

separation distance at any point in the future, a desired vehicle velocity relative to the obstacle will be calculated as:

$$\alpha = \Psi_{v/o} - \Psi_c \tag{2.16}$$

$$\beta = \arcsin \frac{r_s}{r_c} \tag{2.17}$$

$$\gamma = sgn(\alpha)(\beta - \alpha) \tag{2.18}$$

$$\Psi_r = \begin{cases} -\Psi_c & if\ r_c \leq r_s, |\alpha| \geq \frac{\pi}{2} \\ \Psi_c - \frac{\pi}{2} & if\ r_c \leq r_s, |\alpha| < \frac{\pi}{2}, \alpha < 0 \\ \Psi_c + \frac{\pi}{2} & if\ r_c \leq r_s, |\alpha| < \frac{\pi}{2}, \alpha \geq 0 \\ \Psi_{v/o} + \gamma & if\ r_c > r_s, |\alpha| < \beta \\ \Psi_{v/o} & otherwise \end{cases} \tag{2.19}$$

β is calculated using trigonometric operations on a right triangle formed with the distance to the obstacle, r_c, as the hypotenuse and the radius of the circle and vehicle path tangent to the separation window forming the legs. If the vehicle is within the separation window, this triangle cannot be formed and β is undefined. In this case, the relative velocity vector is chosen to be orthogonal to a collision course vector if the vehicle is in front of the obstacle and inside the separation window, and is chosen to point directly away from the obstacle if the vehicle is behind the obstacle and inside the separation window. If the separation distance is not violated, the direction of the relative velocity vector will be rotated such that it is tangent to the separation window in the case that the current relative velocity intersects the separation window. Collision avoidance is not implemented on the hardware-in-the-loop model.

2.2.6 Subsystem Integration

These system components are then integrated together to give a complete vehicle autopilot. Figure 2.6 illustrates the typical analysis process for the unmanned vehicles in the lab. The hardware-in-the-loop (HIL) and software-in-the-loop (SIL) components of this process are presented in Sections 2.3 and 2.4.

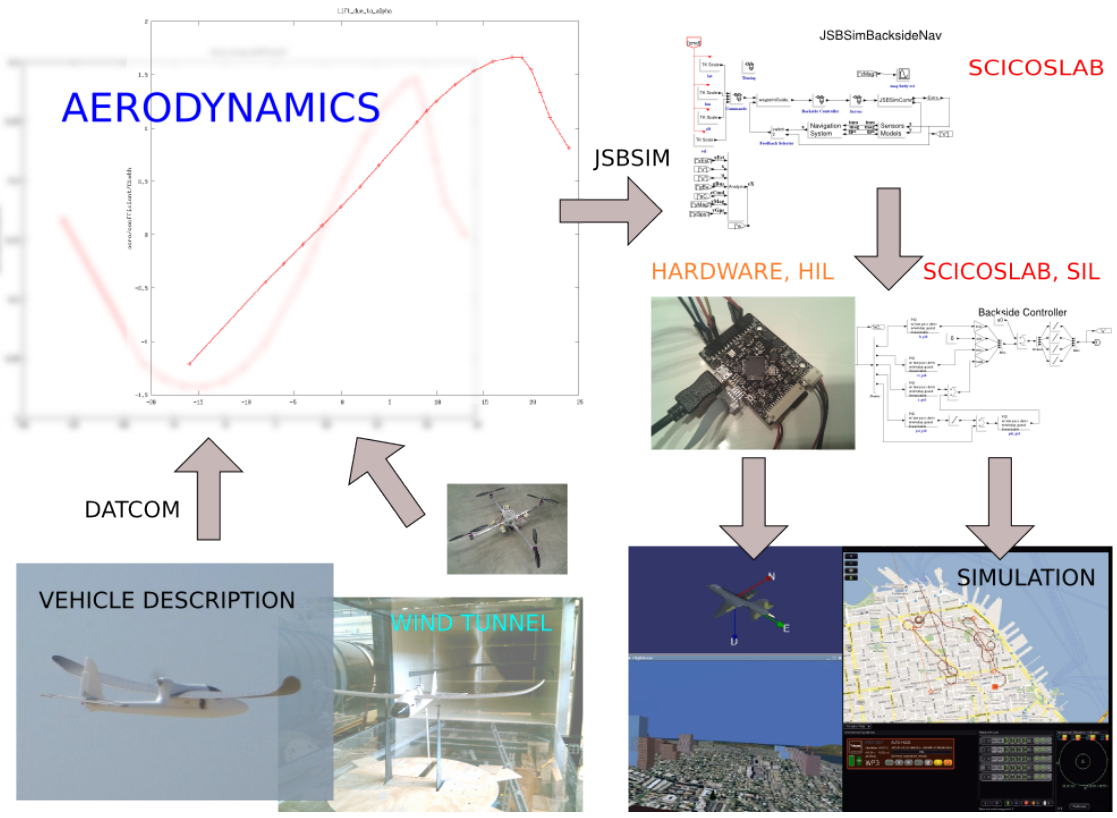


Fig. 2.6. The Purdue Hybrid Systems Lab simulation testbed.

2.2.7 Cyberattack Models

Single mode attacks are attacks in which only one attack avenue is pursued. The different types of identified single mode attacks are categorized in Table 2.4.

Table 2.4.
Single Attacks Considered

fuzzing attack	Introduction of extra noise to sensor data.
actuator attack	Physical modification of actuators (rudders, ailerons, etc.).
digital update rate attack	Slowing the processing rate of the controller/navigator.
navigator state attack	Modification of the on-board navigator state.
sensor spoofing attack	Providing false sensor data.
disguised attack	An attack masquerading as another attack.
undetectable attack	An attack that can't be detected.

When multiple single mode attacks are used on a target simultaneously, it is considered a combined attack. Successful combined attacks are especially dangerous because they give an attacker additional degrees of freedom with which to achieve their objective. If an attack can be intelligently designed, these additional freedoms can be used to amplify the effect of the attack, reduce the detectability of the attack, and/or achieve a result that is not possible with a single attack. The time till failure metric presented above and the fault detection time will be used to determine the effectiveness of the attack, which is function of how effectively failures are introduced, how easily the attack is detected by the fault detection system, and how effective the fault detection system is at preventing failures by reverting to the trim inputs.

2.3 Software-in-the-Loop Analysis of Cyberattacks

In this section, the configuration of the UAS software testbed is presented. The power of this environment is the ability to simulate high-fidelity models using a proven C++ library while interfacing to a block diagram environment. A block diagram environment is very useful for the rapid prototyping and analysis of guidance, navigation and control systems. The current testbed has been utilized by the Purdue Hybrid

Systems Lab for the analysis of autonomous quadrotors, rovers, and fixed wing aircraft.

2.3.1 Software Model

For the software simulation, the navigator and controller shown previously were implemented in ScicosLab, an open source block diagram simulation environment similar to Simulink from MathWorks that provides a mechanism for rapid analysis. The complete Scicos block diagram used for this analysis is shown in Figure 2.7. In this diagram, the commands block provides the commanded waypoint and velocity to the vehicle, and the waypoint guidance block then uses this information combined with information about nearby obstacles to compute a desired bearing and speed for the aircraft. The backside controller block implements digital PID controllers to regulate the error in the control surfaces. The controller in this analysis was updated at 50 Hz, which is the typical update rate for hobby servos commonly used on research UASs. The servos block models the lag in the actuators using a first order approximation. The JSBSimComm block sends the actuator signals to JSBSim where the aircraft state derivative is computed. The ScicosLab block then uses a variable step size integration scheme for propagating the state. The computed state and outputs from the JSBSimComm block are then sent to the sensor models. The sensor models use the state of the aircraft and random noise generation to simulate realistic data from the sensors. This data is then fed into the navigation system, which uses this information to estimate the state of the aircraft.

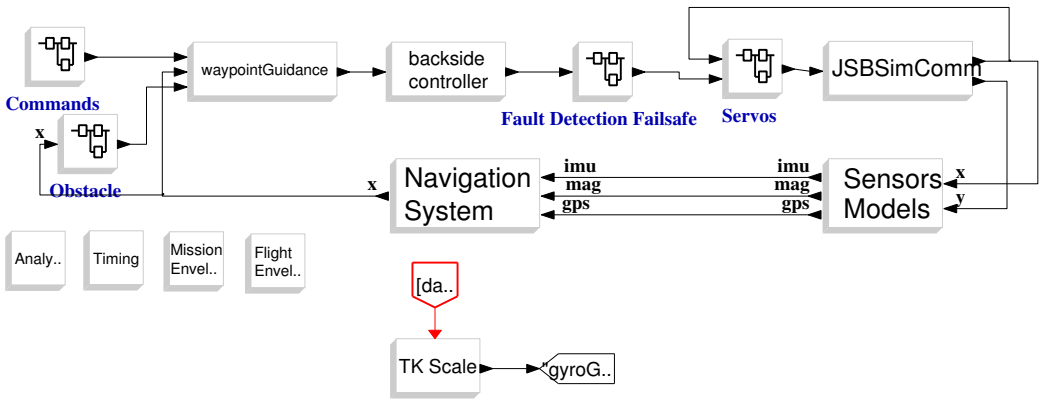


Fig. 2.7. The block diagram of the UAS autopilot in ScicosLab.

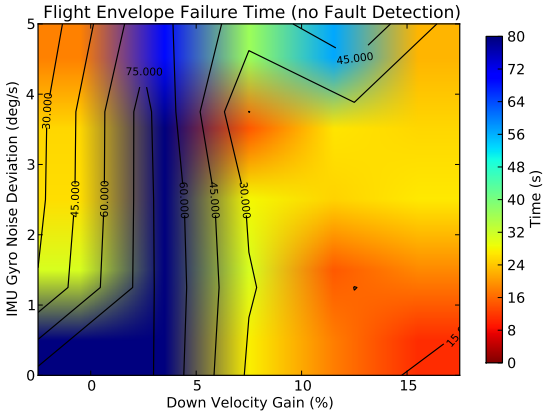
2.3.2 Simulation Results

Cyberattacks were inserted into the block diagram model of the system in Figure 2.7 and vehicle flight was then simulated with attacks of varying magnitude and in different combinations. The flight and mission envelope failure times were recorded for each test, as well as the fault detection time. Each attack was simulated twice, once when the fault detection system reverted to the open-loop trim inputs on a fault state and again when a fault did not change vehicle operations. In the SIL analysis, 37 different attacks were introduced, including sensor gain modifications, sensor fuzzing, navigator state modification, reducing actuator effectiveness, and slowing the controller and navigator update rate. Every two-attack combination from this set was simulated and the failure times analyzed. Of these combinations, approximately one third were unable to introduce a failure or trigger a fault state, indicating no vulnerabilities. Several attacks were particularly effective, including velocity sensor gain modifications, combinations of accelerometer and gyroscope gain and noise modifications, direct manipulation of attitude states, and changing controller and navigator update rates. Selected results are shown below. However, due to the variability of vehicle and autopilot systems, these results may not be applicable in general. The process of identifying the vulnerabilities of a specific vehicle is therefore of greater

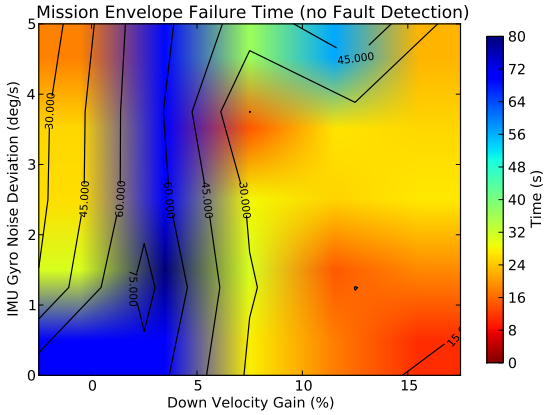
importance than the results. Due to the large number of results, we will focus our discussion on two specific attack combinations.

Figure 2.8 shows the result of a combined attack that modifies the down velocity gain and varies the deviation of the gyroscope noise. The flight envelope and mission envelope failure times are shown for the case in which fault detection is used and when it is not. The time at which the fault is detected is also shown. The region of relative stability at unity down velocity gain in Figures 2.8(a) to 2.8(d) shows that the gyroscope noise attack is not especially effective as it is unable to induce failure without being combined with another attack. Similarly, reducing the down velocity gain below unity is unable to introduce a flight envelope failure without the addition of gyroscope noise, although it is able to introduce a mission envelope failure. The fact that these attacks are more effective at inducing flight envelope failure when used together demonstrates the benefit to the attacker of coupling these two attacks.

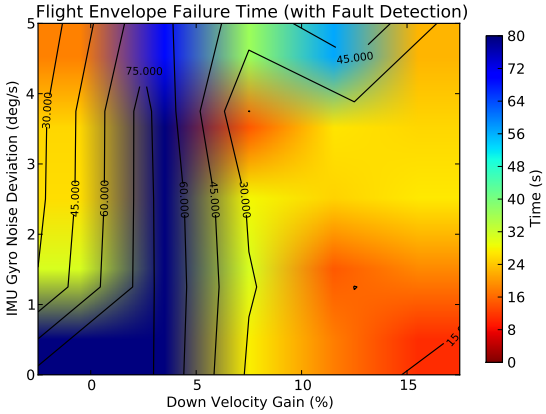
The results also show that the fault detection system is clearly unable to mitigate these attacks, as shown by the identical responses of the system to the attacks for the case when fault detection is used and when it is not. While the system may not be able to automatically mitigate the attack, it does detect the fault for most attack values, as shown in Figure 2.8(e), and will alert the operator to the presence of the attack. There is one isolated region in the fault detection plot in Figure 2.8(e) in which no fault is detected before the flight envelope is violated. Given that most of the other combinations of these two attacks that cause failure are detectable within ten seconds, this attack is particularly effective in that no fault is detected within the approximately 30 seconds it takes to induce failure. The flight envelope of the vehicle will therefore have already been violated by the time the automatic monitor can trigger operator intervention. It is unclear if this result will hold under Monte Carlo analysis.



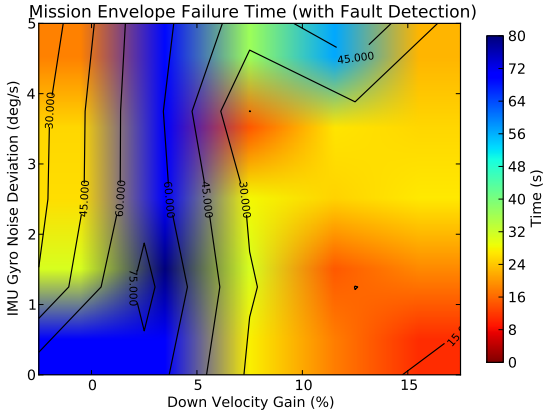
(a) Flight envelope failure time without fault detection



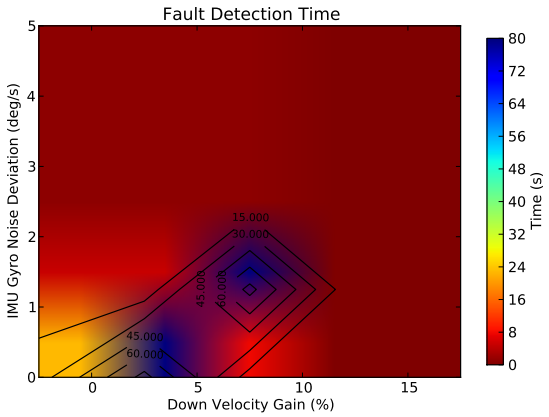
(b) Mission envelope failure time without fault detection



(c) Flight envelope failure time with fault detection

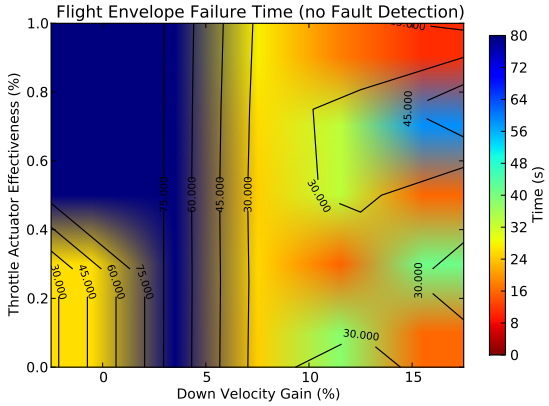


(d) Mission envelope failure time with fault detection

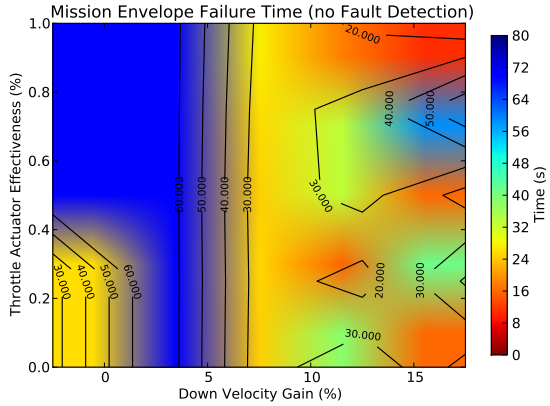


(e) Fault detection time

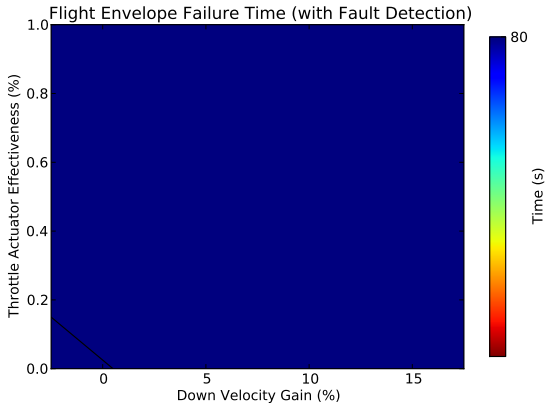
Fig. 2.8. Down Velocity Gain and IMU Gyro Noise Deviation Attack



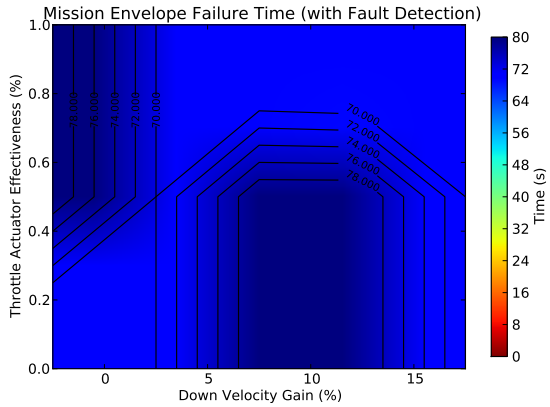
(a) Flight envelope failure time without fault detection



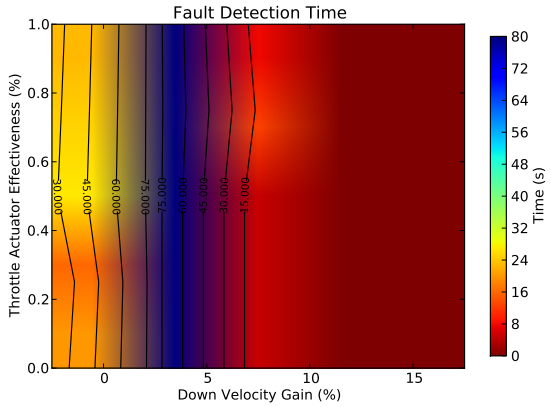
(b) Mission envelope failure time without fault detection



(c) Flight envelope failure time with fault detection



(d) Mission envelope failure time with fault detection



(e) Fault detection time

Fig. 2.9. Down Velocity Gain and Throttle Actuator Effectiveness Attack

Figure 2.9 shows an attack that modifies the down velocity gain and reduces the effectiveness of the throttle actuator. While this attack includes the same down velocity gain modifications as in Figure 2.8, it can be seen from the plots that combining this attack with a throttle effectiveness attack is less effective than combining it with the gyroscope noise insertion attack. In particular, this attack is not successful against systems that revert to the trim input in a fault state. Comparing the flight and mission envelope failures in the no fault detection case, shown in Figures 2.9(a) and 2.9(b), to those of the fault detection case, shown in Figures 2.9(c) and 2.9(d), shows that the fault detection that reverts to the trim inputs during a fault completely removes the ability of the system to induce flight envelope failures in the vehicle. The mission envelope failures that it induces occur at 70 seconds, which is the time at which the vehicle is required to be at the destination waypoint. This would indicate that the mission failures are due to an inability of the vehicle to correctly guide itself to the waypoint, which is expected since the vehicle does not attempt waypoint guidance when a fault has been detected. This is therefore an example of an attack that, while able to compromise an unprotected system, is fairly easily detected and mitigated, and thus not particularly relevant.

Undetectable and Disguised Attacks

An undetectable attack is an attack that is not discovered by the fault monitoring systems, as in Figure 2.9, an attack that targets an unmonitored subsystem, or an attack that is able to induce an irrecoverable instability before being discovered by the monitoring systems. In Figure 2.10, the GPS altitude noise is plotted against initial error in the navigator down velocity and the processor is running at 20% of the nominal speed. The 1 sigma, 3 sigma, and 9 sigma measurement ellipsoids are plotted. These ellipsoids represent the ability of the fault detection system to detect an attack. As the attack moves farther from the nominal point, the probability of the attack being detected increases.

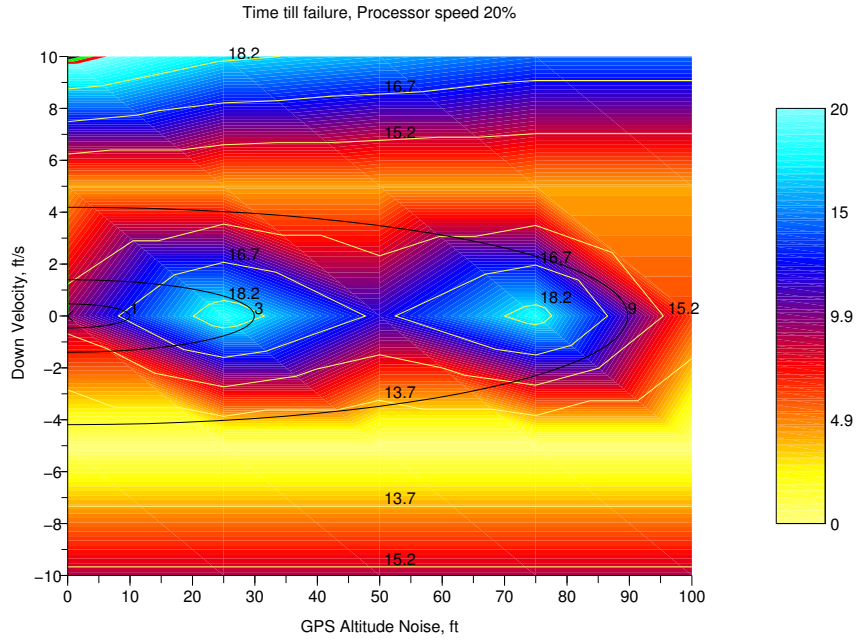


Fig. 2.10. The effect of GPS altitude noise and initial down velocity error on failure time with the processor running at 20% nominal speed. The detection ellipsoids for 1, 3 and 9 sigma (covariance ellipsoids) are plotted.

Conversely, a disguised attack is an attack that is designed to be detected but identifies as a different type of attack. This false identification could cause the vehicle to perform fault mitigation actions that, while effective against the type of attack that was identified, can be leveraged by the actual attack to further its objectives. In this way, the vehicle's defense systems are themselves a vulnerability that can be exploited by an attacker. This type of attack requires a more detailed fault detection scheme for analysis and is a topic for future work.

2.4 Hardware-in-the-loop (HIL) Analysis of Cyberattacks

2.4.1 Background

While conducting software-in-the-loop (SIL) testing is beneficial for rapid prototyping and algorithm development, it is important to verify the functionality of the autopilot on the hardware and software that will be used in the final system as well. In hardware-in-the-loop (HIL) testing, the autopilot hardware is connected to a simulator that sends simulated data to the autopilot in real-time, allowing verification of timing deadlines. The simulator then reacts to the control output of the autopilot. In analyzing cyberattacks, HIL testing is an effective way to discover vulnerabilities in the design and implementation of the system as a supplement to the design analysis provided by SIL testing.

However, transmitting the sensor data from the simulator to the autopilot requires considerable data bandwidth and processing the messages at a high frequency can consume a large portion of the processing power of typical embedded processors. When these resources are not available to conduct *sensor-level* HIL testing, a lower fidelity simulation, *state-level* HIL can be conducted (see definitions below).

HIL Testing Type Definitions

- **sensor-level**: A flight dynamics model simulator (e.g. JSBSim) is used to provide simulated sensor data. The sensor data is sent to the autopilot, which processes all of the sensor data messages and sends the data to the navigation computer. The navigation computer then computes the state of the vehicle, which is used in turn to compute the control output of the autopilot.

- **state-level**: A flight dynamics model simulator (e.g. JSBSim) is used to provide the simulated state of the vehicle, which is sent to the autopilot. The autopilot then processes all of the vehicle state messages and uses the data to compute the control output of the autopilot, bypassing the navigation computer.

Because *state-level* HIL testing does not verify the correct operation of the navigation system, it is preferable to use *sensor-level* HIL testing when the data bandwidth and processing resources are available.

2.4.2 PX4 Autopilot

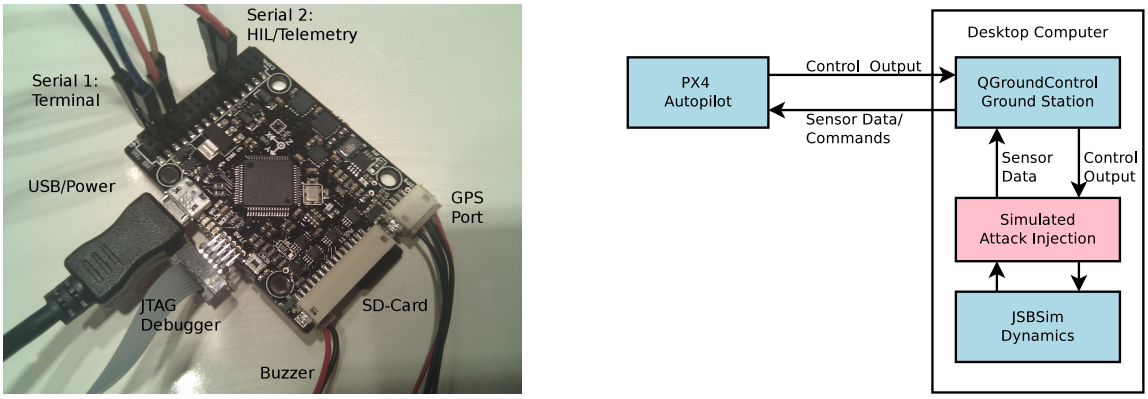
We use the PX4 [4] autopilot for the cyberattack analysis in this chapter. This autopilot is the most recent iteration of open source/open hardware autopilots. The PX4 utilizes a 168 MHz Cortex M4F ARM processor and has 196 KB RAM and 1 MB Flash. The onboard sensors include a 3D accelerometer, 3D gyro, 3D magnetometer, and a barometer. A microSD card is also available for onboard data logging.

The PX4 autopilot software is built upon the open source NuttX real-time operating system (RTOS)[5]. It is fully preemptible, has FIFO or round-robin scheduling, and supports priority inheritance, task controls, named message queues, counting semaphores, clocks/timers, signals, and pthreads similar to those defined by the POSIX [6] standards. It has a task management system and watchdog timers similar to VxWorks [7].

The PX4 autopilot supports the MAVLink air vehicle communication protocol[8]. This protocol allows the PX4 to communicate with all MAVLink compatible ground stations including QGroundControl [9]. The MAVLink protocol has built-in support for both *sensor-level* and *state-level* HIL testing. Previously, PX4 only supported *state-level* HIL, but we have developed support for *sensor-level* HIL. For the analysis in this chapter, *sensor-level* HIL will be used. If *state-level* is used, the interactions of cyberattacks with the hardware navigation system will not be analyzed.

[4] "PX4 Autopilot Platform," https://pixhawk.ethz.ch/px4
[5] "NuttX Real-Time Operating System," http://nuttx.org
[6] "POSIX Standard," http://standards.ieee.org/develop/wg/POSIX.html
[7] "VxWorks RTOS," http://www.windriver.com/products/vxworks
[8] "MAVLink Micro Air Vehicle Communication Protocol," http://qgroundcontrol.org/mavlink/start
[9] "QGroundControl GCS," http://qgroundcontrol.org

The PX4 autopilot and the data flow for *sensor-level* HIL are shown in Figure 2.11. The cyberattack is injected between the JSBSim flight dynamics simulator and the QGroundControl ground station via a Python script. QGroundControl sends both the command and control messages and the sensor data to the PX4 autopilot using the MAVLink protocol. The autopilot then computes a control response and sends this data back to QGroundControl via the MAVLink protocol. QGroundControl forwards the control packets to the JSBSim simulator.



(a) PX4 Autopilot

(b) Data flow

Fig. 2.11. Hardware-in-the-loop attack injection setup for PX4 Autopilot.

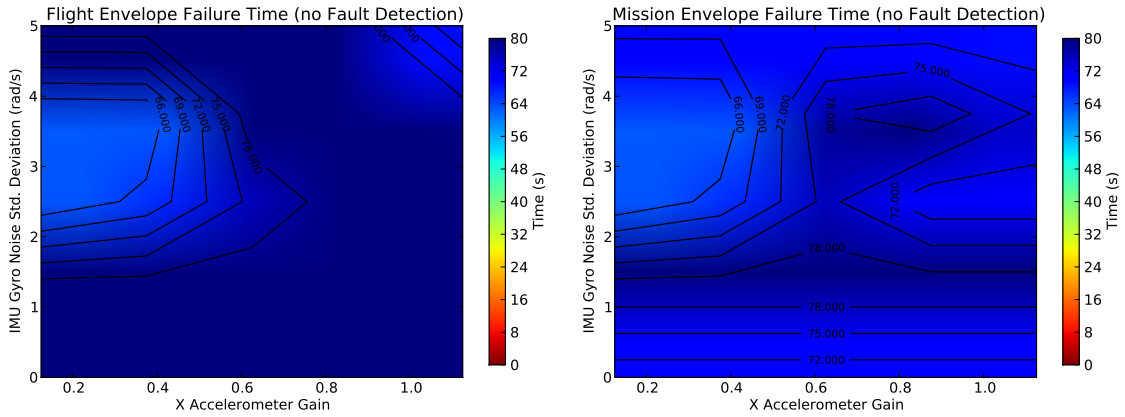
2.4.3 Simulation

Sensor-level hardware validation of selected SIL results was performed, and the results are shown below. While HIL testing provides a more accurate view of the system response to an attack, the added complexity of using the autopilot software instead of a block diagram model makes the modeling of cyberattack inputs more difficult. Additionally, the test must be performed in real-time and cannot be parallelized without using additional hardware, resulting in an increased simulation run time. The required coordination of multiple computers and the corresponding communication channels also makes the automation of long-running simulation difficult. Accordingly,

the HIL testing performed in this analysis is used only to validate interesting results from the SIL simulation. This serves to verify that the vulnerabilities discovered in the system model are present in the implementation of the system. However, without exhaustive hardware testing vulnerabilities that are not present in the model of the system but are introduced in the implementation are not found. Our analysis can be extended to that case as needed.

2.4.4 Selected Results

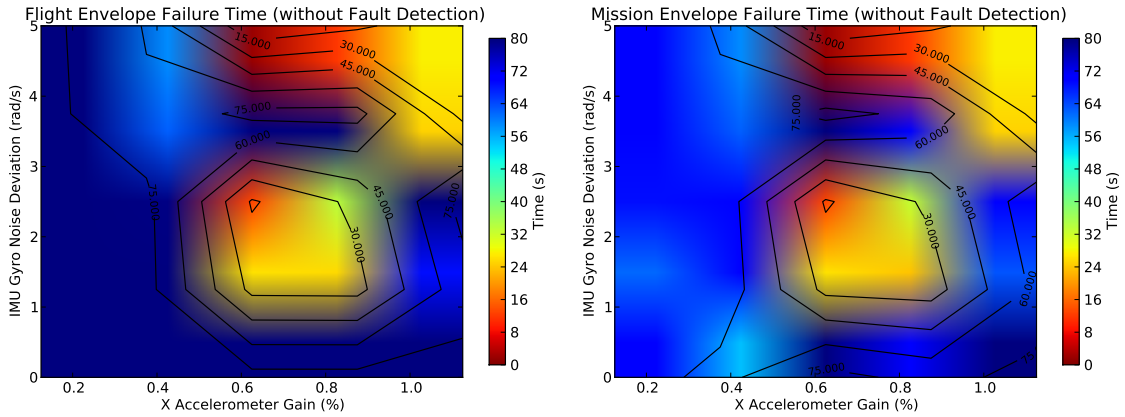
Figure 2.12 shows the flight envelope and mission envelope failure times for the software simulation of a cyberattack that inserts Gaussian noise into the IMU gyroscope measurements and adds a gain to the x accelerometer. The results of the hardware simulation for the same attack are shown in Figure 2.13.



(a) Flight envelope failure time without fault detection

(b) Mission envelope failure time without fault detection

Fig. 2.12. IMU Gyroscope Noise and X Accelerometer Gain Attack



(a) Flight envelope failure time without fault detection

(b) Mission envelope failure time without fault detection

Fig. 2.13. X Accelerometer Gain and None Attack

While these two simulations have some pronounced differences, there are also some significant similarities. The successful attack regions take the same general shape for both simulations, with a failure region at accelerometer gains close to nominal and large gyroscope noise in the corner of the plot, and a separate failure region towards the center of the plot that is a combination of both attacks. The exact location of these regions is different for both simulations, and the failure occurred much faster in the hardware simulation. The differences between these simulations are difficult to explain analytically, given the complexity of the systems, but that we were able to use the SIL results to correctly identify which attack pairings were likely to produce a failure in the HIL analysis and the overall similarities in the results serves to validate the approach.

2.5 Conclusion

In this chapter, we have presented a numerical cyber security analysis method for UASs, using both hardware and software simulation, designed to test the robustness of UASs subject to cyberattack and identify their vulnerabilities. A high-fidelity

model of the vehicle dynamics was created and interfaced with a proven flight simulation software package, enabling the accurate simulation of UAS flight. For the software simulation, an autopilot model was created in a block diagram environment and interfaced to the flight dynamics software to provide high fidelity simulation of UAS operations as designed. For the hardware simulation, the autopilot hardware and accompanying software are interfaced with the flight dynamics software running on a desktop PC via the ground control software, giving a high fidelity simulation of UAS operations as implemented. This capability was leveraged to simulate the response of a UAS to several identified cyberattacks and combinations of cyberattacks, including sensor noise injection, changing the system update rate, modifying sensor gains, and modifying the navigator state. These attacks were shown through simulation to be capable of impeding mission objectives and introducing instability into the vehicle, resulting in airframe failure. This analysis represents a methodology by which control system vulnerabilities can be discovered and mitigated. This approach is particularly useful for complex cyber-physical systems such as UASs and gives insight complimentary to that provided by analytical methods. The use of this analysis enables a comprehensive security approach that incorporates traditional cybersecurity solutions that attempt to deny an attacker access to the system with a secure control approach that reduces the efficacy of tampering if an attacker does gain access.

3. CASE STUDY: INSECT-LIKE FLAPPING WING MAV (MICRO-AIR-VEHICLE)

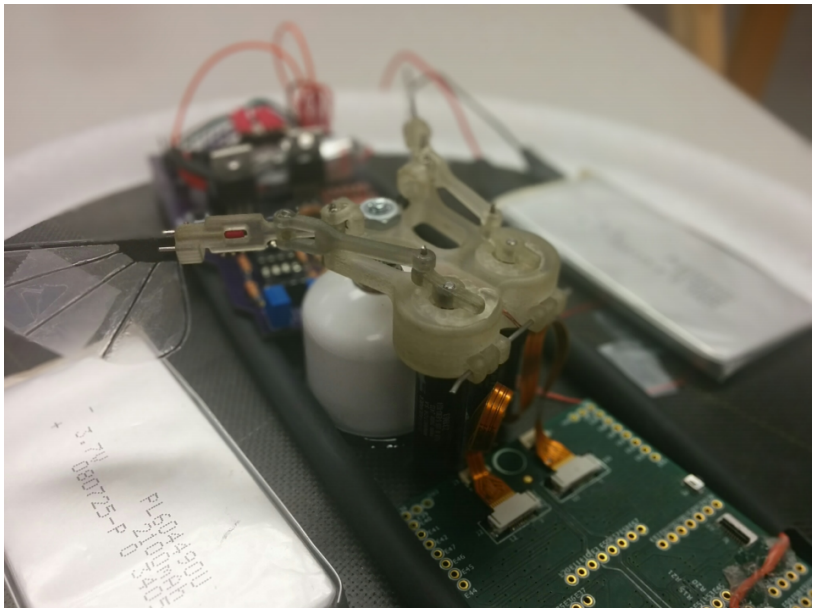


Fig. 3.1. Flapping Wing MAV vehicle used in joint project between Purdue and Wright State University [45]

This chapter proposes a model checking method for a trajectory tracking controller for a flapping wing micro-air-vehicle (MAV) under disturbance. Due to the coupling of the continuous vehicle dynamics and the discrete guidance laws, the system is a hybrid system. Existing hybrid model checkers approximate the model by partitioning the continuous state space into invariant regions (flow pipes) through the use of reachable set computations. There are currently no efficient methods for accounting for unknown disturbances to the system. Neglecting disturbances for the trajectory tracking problem underestimates the reachable set and can fail to detect when the system would reach an unsafe condition. For linear systems, we propose the use of

the H_∞ norm to augment the flow pipes and account for disturbances. We show that dynamic inversion can be coupled with our method to address the nonlinearities in the flapping-wing control system. This chapter is based on the book chapter [46].

3.1 H_∞ Norm Flow Pipe Augmentation

The H_∞ norm is the maximum singular value of a linear system. It can be computed rapidly and can be precomputed for each mode of a linear hybrid system before computing the reachable set of the system. The core idea of this chapter is to use the H_∞ norm to augment the computed flow pipe for the AQTS. We have not yet implemented a model checker with this modification, but we do verify the concept through Monte-Carlo simulation. For linear systems, the computation of the flow pipes can also be done efficiently, so computation of the H_∞ norm augmented flow pipes in real time for adapting systems is feasible.

In Figure 3.2, a computational flow pipe is shown for the set X_0. The H_∞ norm is used to augment the bound based upon the magnitude of the H_∞norm for the current face of the flow pipe.

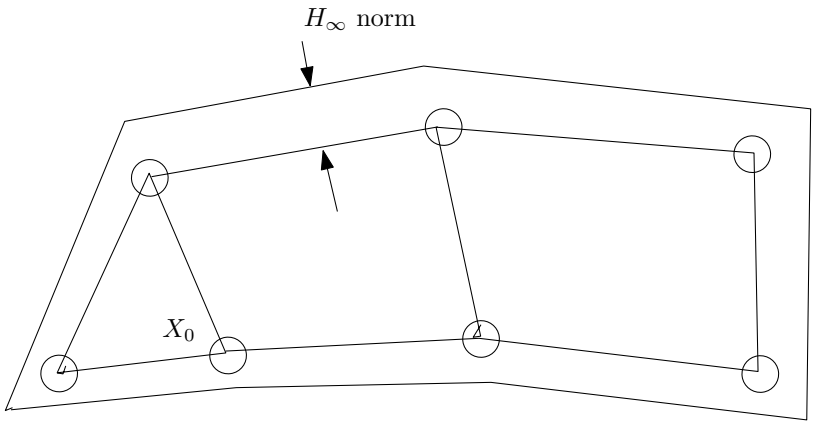


Fig. 3.2. The initial starting triangular set X_0 is propagated using standard reachability techniques to create the inner flow pipe with boundaries indicated by circles. The H_∞ norm augments the existing flow pipe to account for disturbances.

3.2 Flapping Wing Dynamics and Dynamic Inversion Based Control Law

In this section, our goal is to formulate the trajectory tracking control problem so that it can be verified through the use of the H_∞ augmented flow pipe based model checking method described in Section 3.1. A precise model of the flapping wing aerodynamics would be difficult to analyze. Therefore, we analyze a cycle average force and moment model [47]. The vehicle analyzed in this chapter is shown in Figure 3.3. The flapping wing vehicle is mounted rigidly on top of a disc that is hovering on an air table. In this way, we can design the control system without concern for lifting the weight of the vehicle. The air table also limits the motion of the vehicle by keeping it level on the surface of the table.

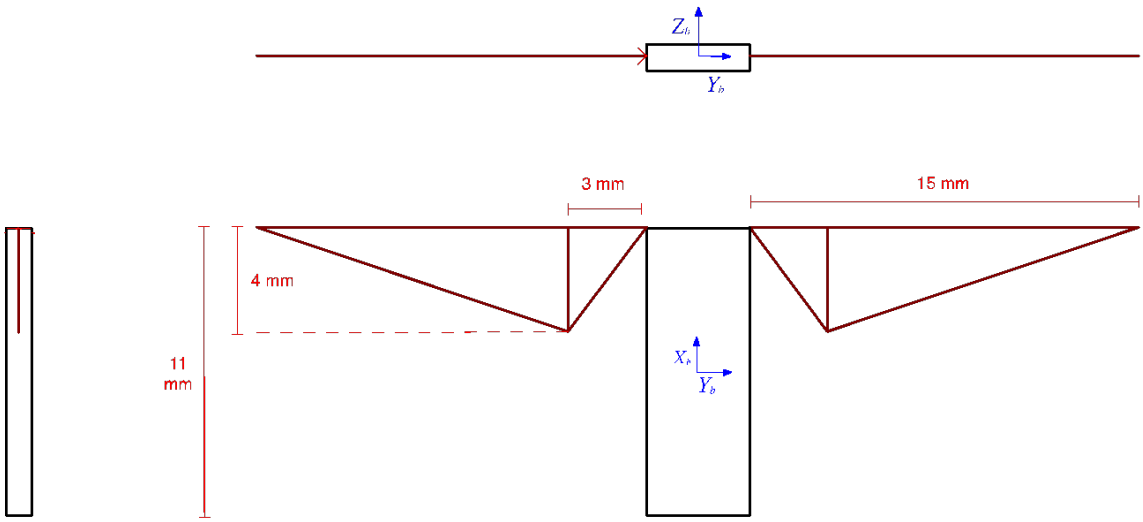


Fig. 3.3. The flapping wing vehicle analyzed in this chapter.

For this flapping wing system, control is obtained through modulation of the flapping cycle. The flapping cycle has two parameters for each wing, the flapping frequency and the flapping delta shift which moves the trough of the function. The effect of the delta shift parameter on the flapping cycle is shown in Figure 3.4.

We assume that a wing flap cycle produces:

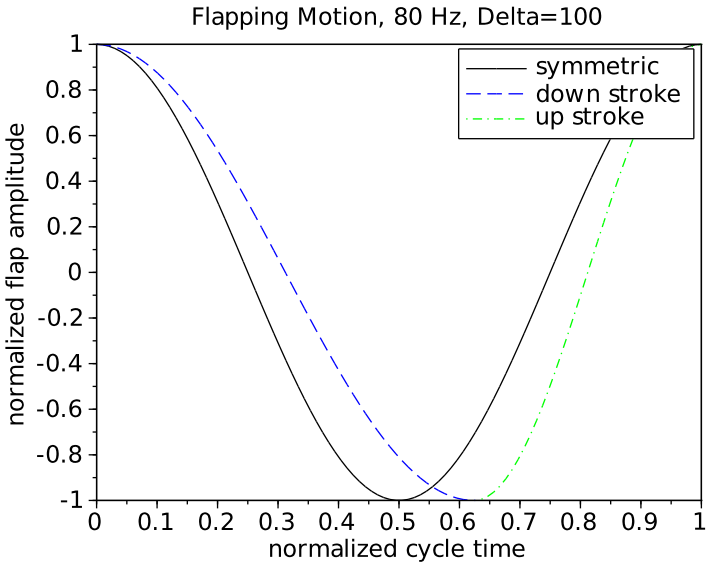


Fig. 3.4. Flapping wing cycle modulation through variation of the trough position of the stroke.

- A force in the body x direction proportional to the average dynamic pressure of the wing (dependent on the flapping frequency and the distance from the axis of rotation to the center of the wing).
- A force in the body y direction proportional to the average dynamic pressure of the wing (dependent on the flapping frequency and the distance from the axis of rotation to the center of the wing). It is assumed that this force has less magnitude since it is not aligned with the lifting direction of the wing.
- A force in the body z direction proportional to the product of the delta-shift of the flapping cycle and the average dynamic pressure of the wing.

Therefore we can derive the following aerodynamic force and moment model.

$$F_{aero} = \rho l_f^2 s/2 \begin{bmatrix} C_x(\omega_R^2 + \omega_L^2) \\ C_y(\omega_R^2 - \omega_L^2) \\ C_z(\delta_R \omega_R^2 + \delta_L \omega_L^2) \end{bmatrix} \tag{3.1}$$

$$M_{aero} = \rho l_f^2 l_w s/2 \begin{bmatrix} C_z(\delta_R \omega_R^2 - \delta_L \omega_L^2) \\ 0 \\ C_x(\omega_R^2 - \omega_L^2) \end{bmatrix} \tag{3.2}$$

where ρ is the air density, l_f is the distance from the flapping axis of rotation to the center of the wings, l_m is the moment arm of the wings, s is the reference area of the wing, (ω_R, ω_L) are the angular velocity of the (right,left) wings, (δ_R, δ_L) are the flapping cycle shift parameters for the (right,left) wings, and (C_x, C_y, C_z) are the (x, y, z) aerodynamic force coefficients.

Due to the air table, there are reaction moments in y and z that keep the flapping wing vehicle level and a reaction force in x, the normal force from the air table. We neglect friction forces and aerodynamic drag due to the low speed of the vehicle. We obtain the following equations of motion for the system:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & \sin(\phi) & \cos(\phi) \\ 0 & -\cos(\phi) & \sin(\phi) \\ 1 & 0 & 0 \end{bmatrix} \left(F_{aero} + \begin{bmatrix} R_{Fx} - mg \\ 0 \\ 0 \end{bmatrix} \right) + \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \tag{3.3}$$

$$J \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & \sin(\phi) & \cos(\phi) \\ 0 & -\cos(\phi) & \sin(\phi) \\ 1 & 0 & 0 \end{bmatrix} \left(M_{aero} + \begin{bmatrix} 0 \\ R_{My} \\ R_{Mz} \end{bmatrix} \right) + \begin{bmatrix} d_\phi \\ d_\theta \\ d_\psi \end{bmatrix} \tag{3.4}$$

where ϕ is the roll angle of the MAV (here, the roll angle is about the vertical axis on the air table), R_{F_x} is the reaction force in the body x direction (the vertical axis on the table), R_{M_y} is the reaction moment about the y axis (side axis on the table), R_{M_z} is the reaction moment about the z axis (forward axis on the table), d_x, d_y, d_z are force disturbances, and d_ϕ, d_θ, d_ψ are moment disturbances.

For the trajectory tracking controller we need a model to track. We choose a simple system where the reference trajectory is specified by the velocity commands as shown in (3.5). The heading of the vehicle on the air table, ϕ, is chosen so that the vehicle regulates its heading in the direction of its velocity vector. This choice was

made to keep the primary accelerations in the direction of the body z axis, pointing forward on the air table, where the wings have the most control authority since they are most aligned with the direction of the lift force from the wings. The ability of the wings to generate a side force through disparity of flapping frequency has less control authority than variation of the delta shifts in the flapping pattern to generate a forward force. This enables the model to match the desired linear system dynamics over a wider range of the flight envelope.

$$\begin{bmatrix} \dot{x_r} \\ \dot{y_r} \\ \dot{\phi_r} \end{bmatrix} = \begin{bmatrix} V_{rx} \\ V_{ry} \\ atan2(V_{ry}, V_{rx}) \end{bmatrix} \tag{3.5}$$

where $atan2$ is the quadrant accurate inverse tangent function, (x_r, y_r) are the Cartesian coordinates of the reference trajectory, (V_{rx}, V_{ry}) are the reference trajectory's velocities, and ϕ_r is roll angle in body or heading on the air table.

We now set the trajectory tracking error dynamics through dynamic inversion. We are only concerned with the states that are not constrained by the air table (x, y, z, ϕ). We invert the dynamics so that they match the model:

$$\begin{bmatrix} \ddot{e_x} \\ \ddot{e_y} \\ \ddot{e_z} \\ \ddot{e_\phi} \end{bmatrix} + 2\zeta\omega_n \begin{bmatrix} \dot{e_x} \\ \dot{e_y} \\ \dot{e_z} \\ \dot{e_\phi} \end{bmatrix} + \omega_n^2 \begin{bmatrix} e_x \\ e_y \\ e_z \\ e_\phi \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \\ d_z \\ d_\phi \end{bmatrix} \tag{3.6}$$

where $e_x = x_r - x$, $e_y = y_r - y$, $e_z = z_r - z$, w_n is the natural frequency of the 2^{nd}order system and ζ is the damping ratio. The dynamics for each state are not coupled, so it is not necessary to have the same natural frequency or damping ratio, but we choose this for simplicity.

We now can compute the force and moments as a function of the tracking errors in order to match the above model. We can neglect F_x because the flapping wing vehicle is constrained to remain on the surface of the air table since the lift is less than the weight. In the equations, the value F_x sets the steady state flapping frequencies.

$$\begin{bmatrix} M_{bx} \\ F_{by} \\ F_{bz} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ \sin(\phi) & -\cos(\phi) & 0 \\ \cos(\phi) & \sin(\phi) & 0 \end{bmatrix} \begin{bmatrix} -\omega_n^2 e_x - 2\zeta\omega_n \dot{e_x} \\ -\omega_n^2 e_y - 2\zeta\omega_n \dot{e_y} \\ -\omega_n^2 e_\phi - 2\zeta\omega_n \dot{e_\phi} \end{bmatrix} \tag{3.7}$$

Because there are 4 control variables and 4 degrees of freedom, there is a unique mapping of the controls to the desired forces and moments.

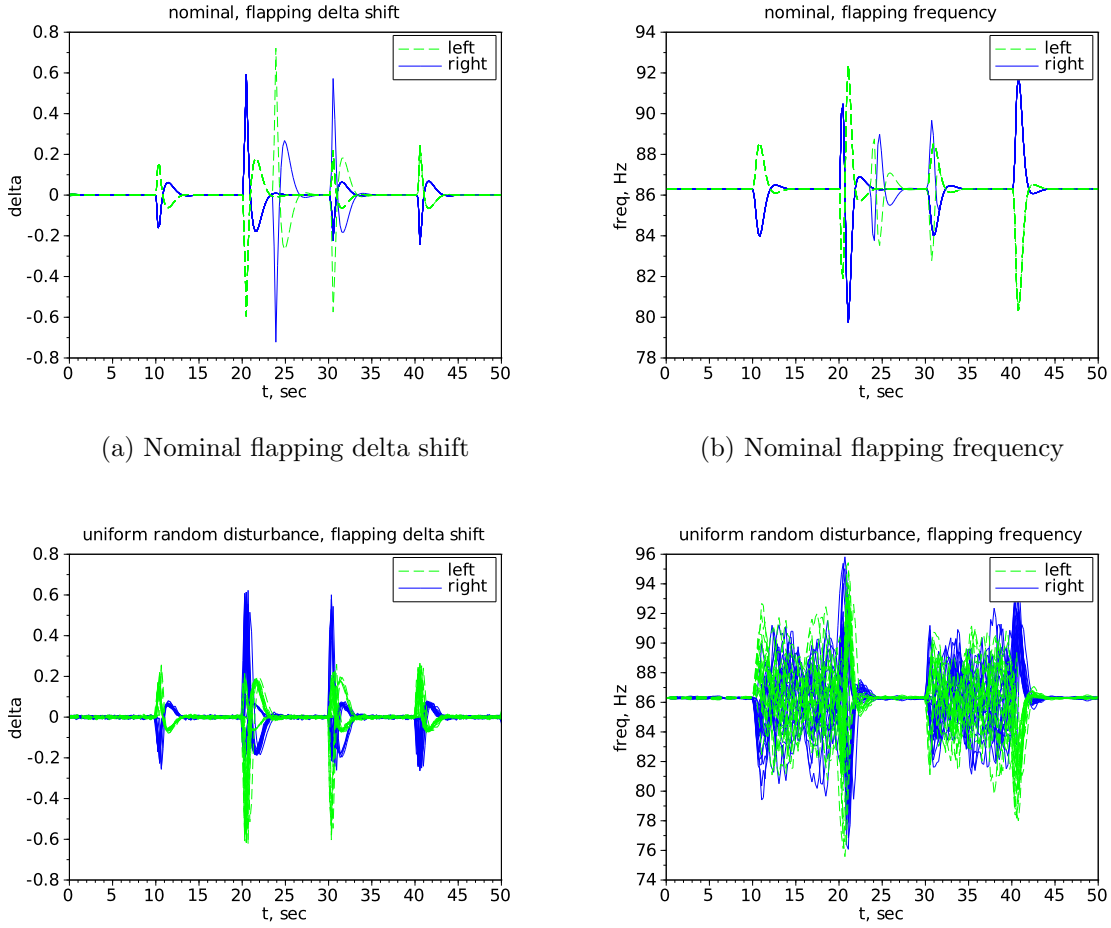
$$\begin{bmatrix} \omega_R^2 \\ \omega_L^2 \\ \delta_R \\ \delta_L \end{bmatrix} = \frac{1}{\rho l_f^2} \begin{bmatrix} \frac{F_{bx}}{C_x} + \frac{F_{by}}{C_y} \\ \frac{F_{bx}}{C_x} - \frac{F_{by}}{C_y} \\ \frac{1}{\omega_R^2}\left(\frac{F_{bz}}{C_z} + \frac{M_{bx}}{C_z l_w}\right) \\ \frac{1}{\omega_L^2}\left(\frac{F_{bz}}{C_z} - \frac{M_{bx}}{C_z l_w}\right) \end{bmatrix} \tag{3.8}$$

Note in (3.8) that the ω_R^2 and ω_L^2 used in computation of δ_R and δ_L could be expressed instead as explicit functions of the desired forces and moments as given by the first two elements of (3.8).

In Figure 3.5, we plot the delta shifts and flapping frequencies required to generate the forces and moments experienced during a simulation of the flapping wing trajectory tracking control system for both the nominal and expected disturbance case. Both the delta shift and flapping frequencies are in reasonable ranges. The spikes in delta shift and flapping frequency occur when the vehicle is at a new waypoint and beginning to move in a new direction. Dynamic inversion based control is susceptible to modeling error, but these modeling errors can also be included as unknown disturbances to the system. Therefore, dynamic inversion integrates well with the proposed H_∞ based flow pipe augmentation model checking method we propose.

3.3 Flapping Wing Controller Model Checking

Now that we have successfully inverted the dynamics of the flapping wing trajectory tracking control system, we can use a PIHA model of the system with the H_∞ norm augmented flow pipes to check the control system in the presence of bounded disturbances. We will use Computational Tree Logic (CTL) to express the universal



(a) Nominal flapping delta shift

(b) Nominal flapping frequency

(c) Disturbance flapping delta shift

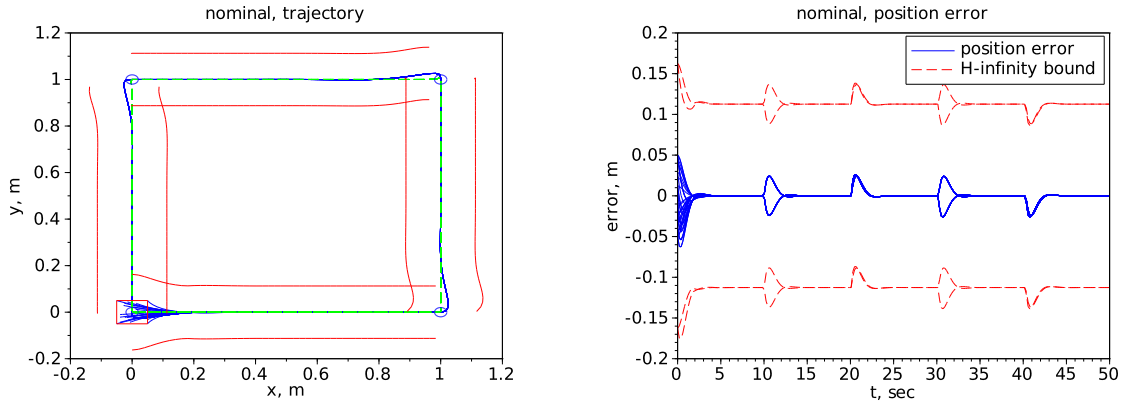
(d) Disturbance flapping frequency

Fig. 3.5. The time history of the delta shift and flapping frequency required for the dynamic inversion based control for both the nominal and uniform random disturbance case. Uniform random noise was used since it is bounded. Bounded Gaussian noise could also be used.

properties we wish to verify [48]. CTL is a branching time logic which can represent non-deterministic transitions and is used by model checkers such as NuSMV [49]. For example, we wish to verify the property $AF\ AG\ W1$ (always end at waypoint 1) in the presence of bounded disturbances of magnitude 0.4 Newtons of force and 0.4 Newton-meters of torque. Disturbances that are H_∞ worst case frequency sinusoids and disturbances that are uniform random numbers sampled at 20 Hz will be tested

with the given magnitudes. For the Monte-Carlo test, the phase of the H_∞ sinusoids will be sampled from a uniform distribution.

In Figure 3.6, the nominal trajectory of the vehicle is shown. The H_∞ norm augmented flow pipes are shown as the red lines in both the trajectory and position error plots. Note that the H_∞ norm is a constant that is added to the reachable set from the initial state (the blue region). For these plots both the x and y errors are plotted on the position error plot and that is why there are two sets of bounds. At the beginning of the simulation, the trajectories are widely dispersed but they quickly converge to the reference trajectory due to the tracking control system. The bounds could be improved by recomputing the flow pipes with no disturbance given the initial set from the previous flow pipe with disturbance.



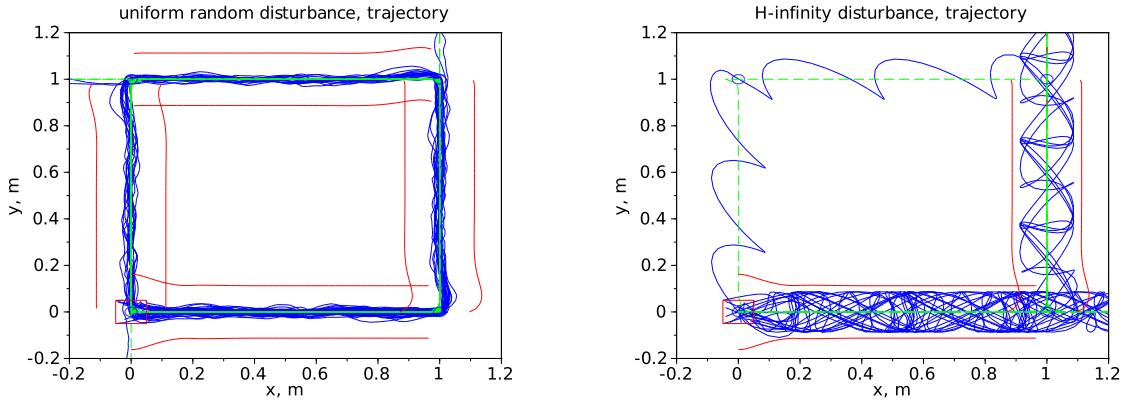
(a) Nominal trajectory

(b) Nominal position error

Fig. 3.6. A Monte-Carlo simulation with no disturbance propagated from initial set x_0 (the square at (0,0))

For the existing AQTS representing the trajectory tracking control system as shown in Figure 1.1, we plot Monte-Carlo simulations in Figure 3.7. The uniform and H_∞ disturbances are both able to cause the system to violate the specification. This occurs because the MAV transitions to the next flow pipe only if the current state is within a set radius of the waypoint. When the disturbance level is large

enough, it becomes possible for the vehicle not to pass inside this radius and the vehicle fails to return to $W1$.



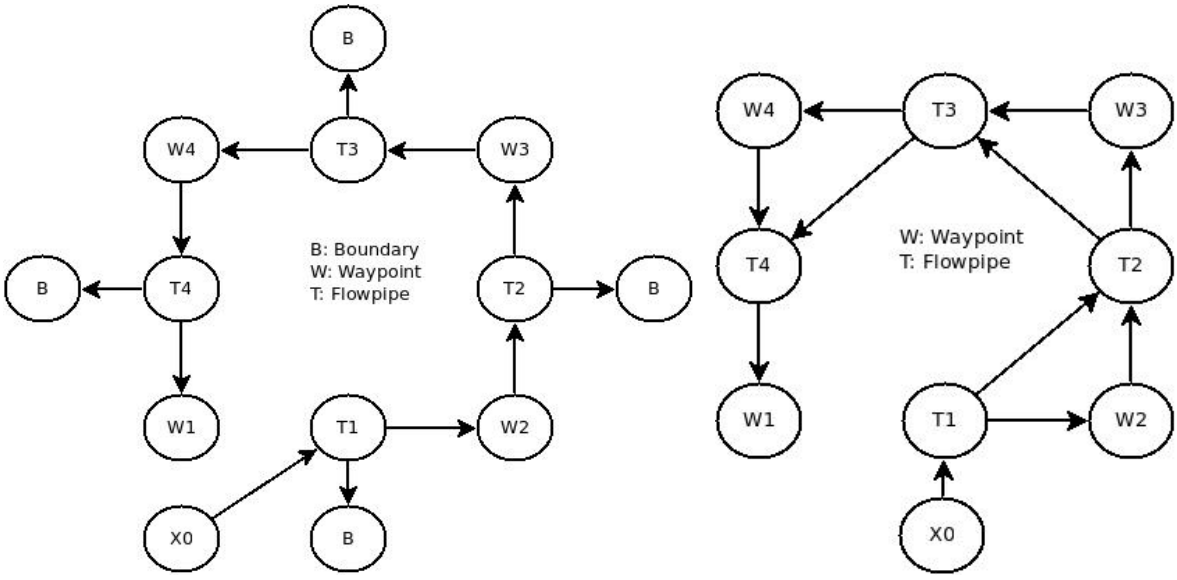
(a) Uniform random disturbance

(b) H_∞ disturbance

Fig. 3.7. A Monte-Carlo simulation showing failures of the radius based waypoint transition guard.

If we no longer use a transition guard that depends on the distance of the vehicle from the waypoint, and we now use the along track distance of the vehicle, a more robust control system can be obtained. The along track distance is the distance between the previous waypoint and the projection of the current vehicle position onto the line between the waypoints. The new guidance law will transition to the next waypoint when the along track distance is greater than the distance between the waypoints, meaning the vehicle has passed the waypoint. In Figure 3.8, the modification to the AQTS is shown. Note that in the modified system it is possible that the vehicle will not be within the defined radius, but it will always reach $W1$.

Finally, we check the new modified AQTS with the along track guidance law using Monte-Carlo simulations with H_∞ and uniform random disturbances in Figure 3.9. We see that there are no trajectories that enter the boundary and all trajectories return to $W1$. We see that for the new guidance law the specfication $AF\ AG\ W1$ (always end at waypoint 1) is satisfied.



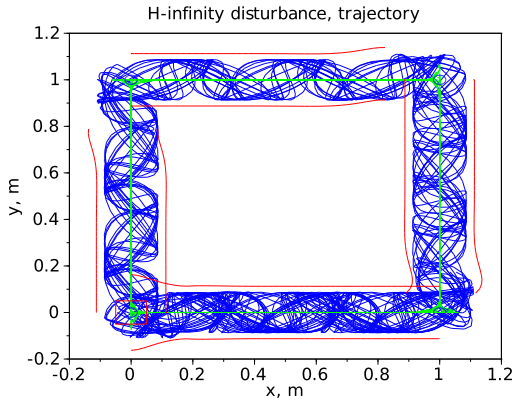
(a) Radius waypoint guard (b) Along track waypoint guard

Fig. 3.8. The AQTSs representing the flapping wing system with the original radius based waypoint guard and the along track based waypoint guard.

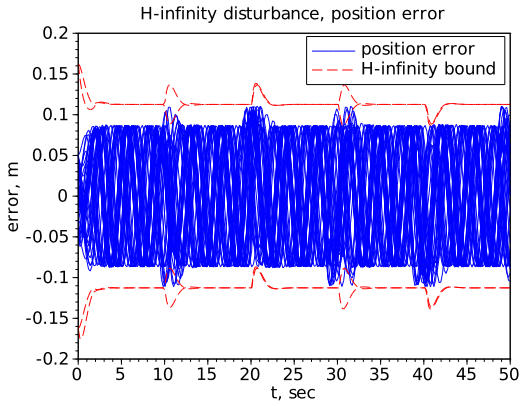
The guidance logic that we have improved in this simple example would be obvious to any autopilot designer, but the complications that disturbances have on more intricate components of a guidance system can be more difficult to discover and having an automated model checking program to verify these systems in the presence of large disturbances will improve the safety of such systems.

3.4 Conclusion

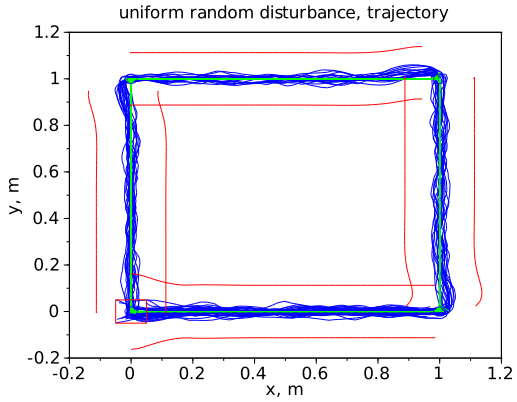
We have proposed a method for verification and validation (V&V) of a flapping-wing micro air vehicle controller (MAV). We have extended the existing algorithms for V&V of Polyhedral Invariant Hybrid Automatons (PIHAs) to account for bounded disturbances in linear hybrid systems using the H_∞ norm. While the H_∞ norm does not strictly bound all possible disturbances, because it only bounds the worst case steady state sinusoidal disturbance, it does successfully bound disturbances expected



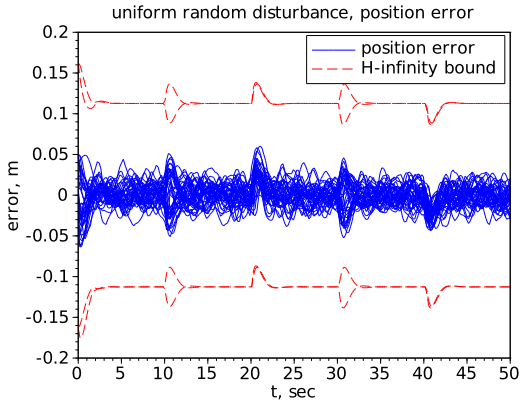
(a) H-infinity trajectory



(b) H-infinity position error



(c) Uniform random disturbance trajectory



(d) Uniform random position error

Fig. 3.9. A Monte-Carlo simulation showing that the along track based waypoint transition guard is resilient in the presence of disturbances.

in practice. If large unbounded disturbances are expected, it will be necessary to consider probabilistic model checking and the probability density function of the regulated system around the nominal trajectory. In order for the H_∞ norm bound to be applicable, the system dynamics must be linear over the flight envelope; however, we have shown how our method can be coupled with dynamic inversion based controllers to extend our method to nonlinear hybrid systems. The H_∞ norm of the system can be computed efficiently and only requires updating when the linear system model changes. Coupled with the efficient reachable set computations for linear systems, this makes our approach attractive for runtime-assurance of adaptive systems.

4. CASE STUDY: TWO-WHEEL SELF-BALANCING ROBOT

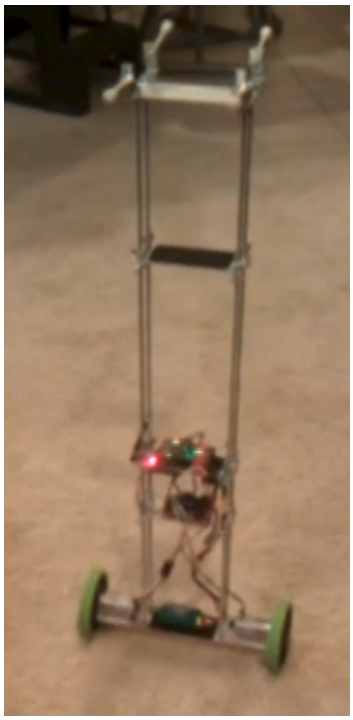


Fig. 4.1. Two-wheel self-balancing robot (see video)

In this chapter, we focus on a two-wheel self-balancing robot. This robot has relatively simple dynamics compared to robots in later chapters but introduces the complexity of under-actuated dynamics, that can't be easily linearized. There are two degrees of freedom, the pitch of the robot, θ, and the position of the robot $\mathbf{x}$, but only one control input, the motor voltage. One approach to deal with under-actuated dynamics is to use dynamic inversion to linearize the output of interest and show that the other internal dynamics, referred to as zero dynamics, are bounded using

Lyapunov functions. This approach is difficult since there is no mechanical way to find the required Lyapunov function. The other approach is to use time constant separation or cascaded dynamic inversion, where an inner loop is linearized, and the outer loop is linearized using the reference command for the inner loop but at a slower speed.

We will first analyze the dynamics and use system identification to fit the parameters of our dynamic model to the data. We will attempt to make the dynamic model as simple as possible while still fitting the data well so that application of PIHA based model checking is feasible.

4.1 Motor Dynamics

A simplified model of a DC motor neglecting electrical inductance is given by:

$$J\ddot{\alpha} + \tau + \left(\nu + \frac{k_b k_\tau}{R}\right)\dot{\alpha} - \frac{k_\tau V}{R} = 0$$

where α is the shaft angle, J is the moment of inertia, V is the voltage, τ is the externally applied torque, R is the resistance of the motor, k_b is the back electromotive force constant, k_τ is the torque coefficient, and ν is the viscous damping.

Manufacture specifications are typically given in terms of no load and stall conditions. These conditions can be used with the motor equation of motion to derive the relationship to the previously defined motor parameters.

$$R = \frac{V_s}{I_s}, \quad k_b = \frac{1}{I_s\omega_{nl}}\left(-I_{nl}V_s + I_s V_{nl}\right), \quad k_\tau = \frac{\tau_s}{I_s}, \quad \nu = \frac{I_{nl}\tau_s}{I_s\omega_{nl}}$$

where V is the voltage, I is the current, and ω is the angular velocity for stall (s), and no load (nl) conditions, R is the resistance of the motor, k_b is the back electromotive force constant, k_τ is the torque coefficient, and ν is the viscous damping.

We can simplify this by defining two new coefficients:

$$\ddot{\alpha} + \tau/J + \dot{\alpha}k_{damp}/J - Vk_{emf}/J = 0$$

where $k_{damp} = \left(\nu + \frac{k_b k_\tau}{R}\right)$ and $k_{emf} = \frac{k_\tau}{R}$.

4.2 Overall Dynamics

The overall dynamics of the self-balancing robot (neglecting the yaw degree of freedom, and assuming no wheel slip) can be found by solving 5 simultaneous equations (the no slip condition, the derivative of the no slip condition, the motor equation, the translation equation of motion, and the rotational equation of motion). The equations of interest, the two degrees of freedom (θ, $\mathbf{x}$) and are given by:

$$\ddot{\theta} = \frac{2J_m l m r_w^2 \sin(\theta)\left(\dot{\theta}\right)^2 - 4J_m r_w^2 v + r_w(glm\sin(\theta) - 2w)\left(2J_m - 2J_w + mr_w^2 + 2m_w r_w^2\right) + 2\left(-2J_w + mr_w^2 + 2m_w r_w^2\right)\left(Vk_{emf}r_w + k_{damp}r_w\dot{\theta} - k_{damp}\dot{x}\right)}{r_w(2JJ_m - 2JJ_w + Jmr_w^2 + 2Jm_w r_w^2 + 4J_m J_w + 2J_m l m r_w \cos(\theta) - 2J_m m r_w^2 - 4J_m m_w r_w^2)}$$

$$\ddot{x} = \frac{-lmr_w^2(J - 2J_m)\sin(\theta)\left(\dot{\theta}\right)^2 + 2r_w^2 v(J - 2J_m) + r_w(2J_m + lmr_w\cos(\theta))(glm\sin(\theta) - 2w) + 2(J + lmr_w\cos(\theta))\left(Vk_{emf}r_w + k_{damp}r_w\dot{\theta} - k_{damp}\dot{x}\right)}{2JJ_m - 2JJ_w + Jmr_w^2 + 2Jm_w r_w^2 + 4J_m J_w + 2J_m l m r_w \cos(\theta) - 2J_m m r_w^2 - 4J_m m_w r_w^2}$$

where r_w is the radius of the wheel, J_w is the moment of inertia of the wheel, m_w is the mass of the wheel, J_m is the moment of inertia of the motor, l, is the distance of the center of mass above the axle, w is a disturbance torque on the robot, v is a disturbance force on the axle, x is the position of the robot, and θ is the pitch of the robot.

The mass and moment of inertia of the vehicle structure (m, J) are much larger than the mass and moment of inertia of the motor and wheels; therefore, we can greatly simplify the above expression using the approximation $J_w = 0, m_w = 0, J_m = 0$. When fitting the model, the existing parameters will accommodate for the total mass and inertia properties.

$$Jr_w\ddot{\theta} = 2Vk_{emf}r_w + 2k_{damp}r_w\dot{\theta} - 2k_{damp}\dot{x} + r_w\left(glm\sin\left(\theta\right) - 2w\right)$$

$$\begin{aligned} Jmr_w^2\ddot{x} = &- Jlmr_w^2\sin\left(\theta\right)\left(\dot{\theta}\right)^2 + 2Jr_w^2 v + lmr_w^2\left(glm\sin\left(\theta\right) - 2w\right)\cos\left(\theta\right) + \ldots \\ &2\left(J + lmr_w\cos\left(\theta\right)\right)\left(Vk_{emf}r_w + k_{damp}r_w\dot{\theta} - k_{damp}\dot{x}\right) \end{aligned}$$

4.2.1 Motor Test: No external torque

In this test, we are identifying the rise time and the amplitude of the response to voltage input for the motors. This corresponds to the parameters k_{damp}/J and k_{emf}/J in the motor equation of motion. We isolate this response by lifting the vehicle off the ground so that there is no external torque applied.

When the external torque, τ, is zero, the differential equation can be written in the form:

$$Y = X\Theta$$

$$\Theta = X^{+}Y$$

where

$$Y = \begin{bmatrix} \ddot{\alpha}_0 \\ \ddot{\alpha}_1 \\ \vdots \end{bmatrix} X = \begin{bmatrix} -\dot{\alpha}_0 & V_0 \\ -\dot{\alpha}_1 & V_1 \\ \vdots & \vdots \end{bmatrix} \qquad \Theta = \begin{bmatrix} k_{damp}/J \\ k_{emf}/J \end{bmatrix} \tag{4.1}$$

The pseudo-inverse can be then used to obtain the least squares solution of the parameters. This method was chosen for its simplicity and speed.

$$\Theta = (X^T X)^{-1} X^T Y$$

In Figure 4.2, we see that the fit matches the validation data set well. This test can only reveal two parameters, the relative damping and torque response to voltage, but it cannot at the same time determine the coefficient for the response to an external torque. This is clear since both of the identified parameters are divided by J_m. In order to find J_m, we need to apply an external torque, we do this in the next experiment using the gravity torque of the vehicle.

4.2.2 Motor Test: Gravity torque

To identify the motor moment of inertia, J_m, we mount the vehicle so that it is above the ground with the wheels clamped in place and can rotate freely about the

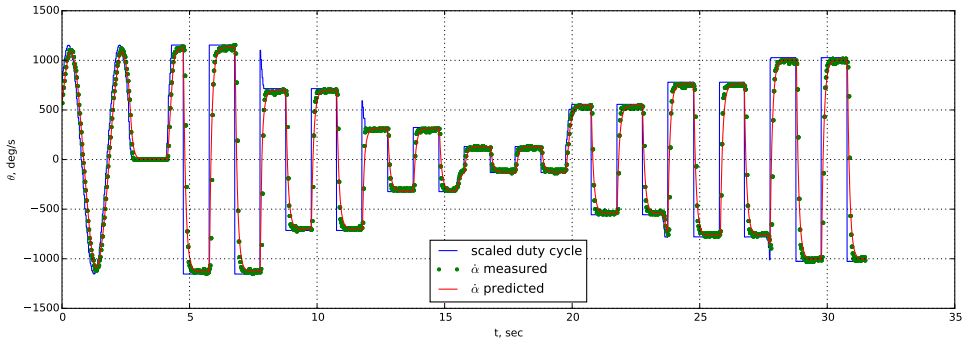


Fig. 4.2. Open loop system identification of wheel-motor (validation data).

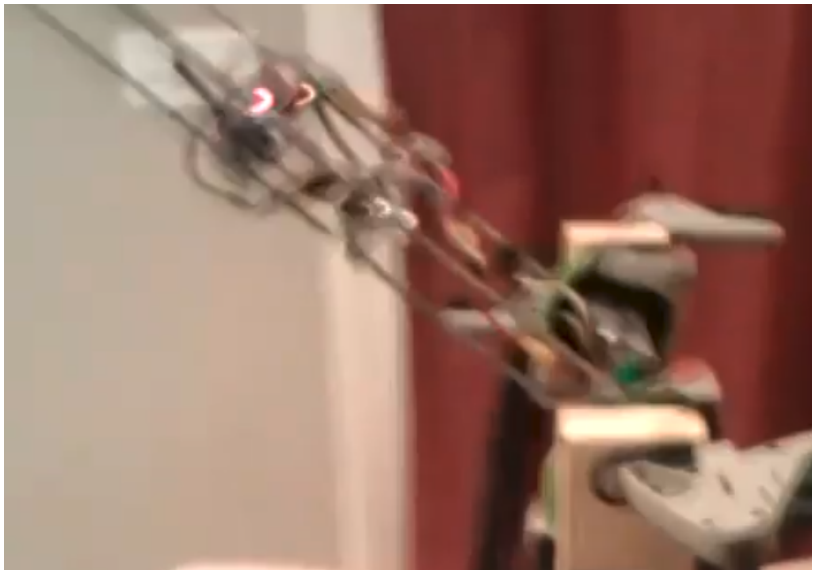


Fig. 4.3. Gravity torque experiment in progress (see video)

wheels without ever touching the ground. We then can let the vehicle act as a simple pendulum. The measured pitch angle is then used to compute the applied gravity torque at each time instant.

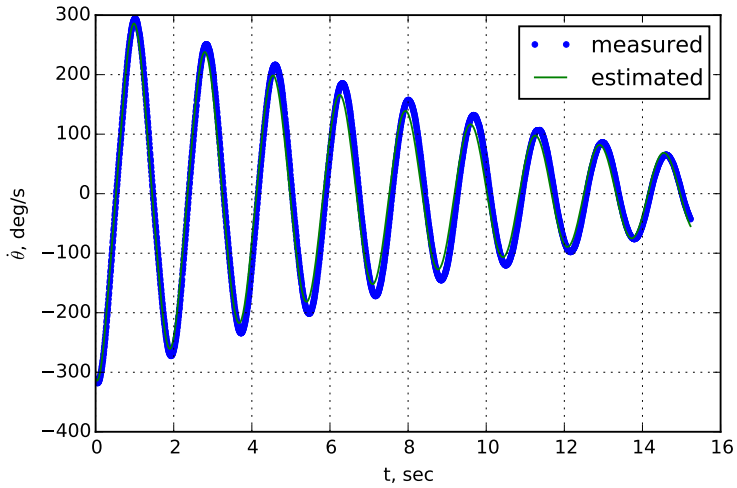


Fig. 4.4. Open loop system identification validation data for gravity torque experiment.

4.3 Closed Loop System Identification

Now that the vehicle parameters have been identified, we can use the model of the vehicle to detect damage or a cyberattack. We use the data from the available sensor coupled with the mathematical model to obtain pseudo-measurements for unmeasurable model parameters such as the mass of the robot. This algorithm is mechanized using an extended kalman filter, (EKF), where an additional state is added for each parameter that is expected to change. Estimating all of the vehicle parameters simultaneously is not practical with the measurements available. The EKF was mentioned in Chapter 2 so we only give the relevant vectors to describe the filting algorithm below:

$$\mathbf{x} = \begin{bmatrix} \theta & \dot{\theta} & x & \dot{x} & m \end{bmatrix}^T \qquad \mathbf{u} = V \qquad \mathbf{y} = \begin{bmatrix} \theta & \dot{\theta} & x & \dot{x} \end{bmatrix}^T$$

The results of the closed loop system identification of the mass of the vehicle are shown in Figure 4.3. Due to the non-linearity of the EKF algorithm, it is essential for it to have reasonable starting estimates for the parameters, measurement noise, and

process noise. If the process noise is too high, then the model will gain no information from the data. If the measurement noise is too high, then the filter will not converge to the true value. Note that the velocity, $\dot{x}$, measurement is very noisy but the filter smooths the estimate out using the process model. The value of the pitch, θ, is also initially incorrect but converges after the mass parameter converges to the known value.

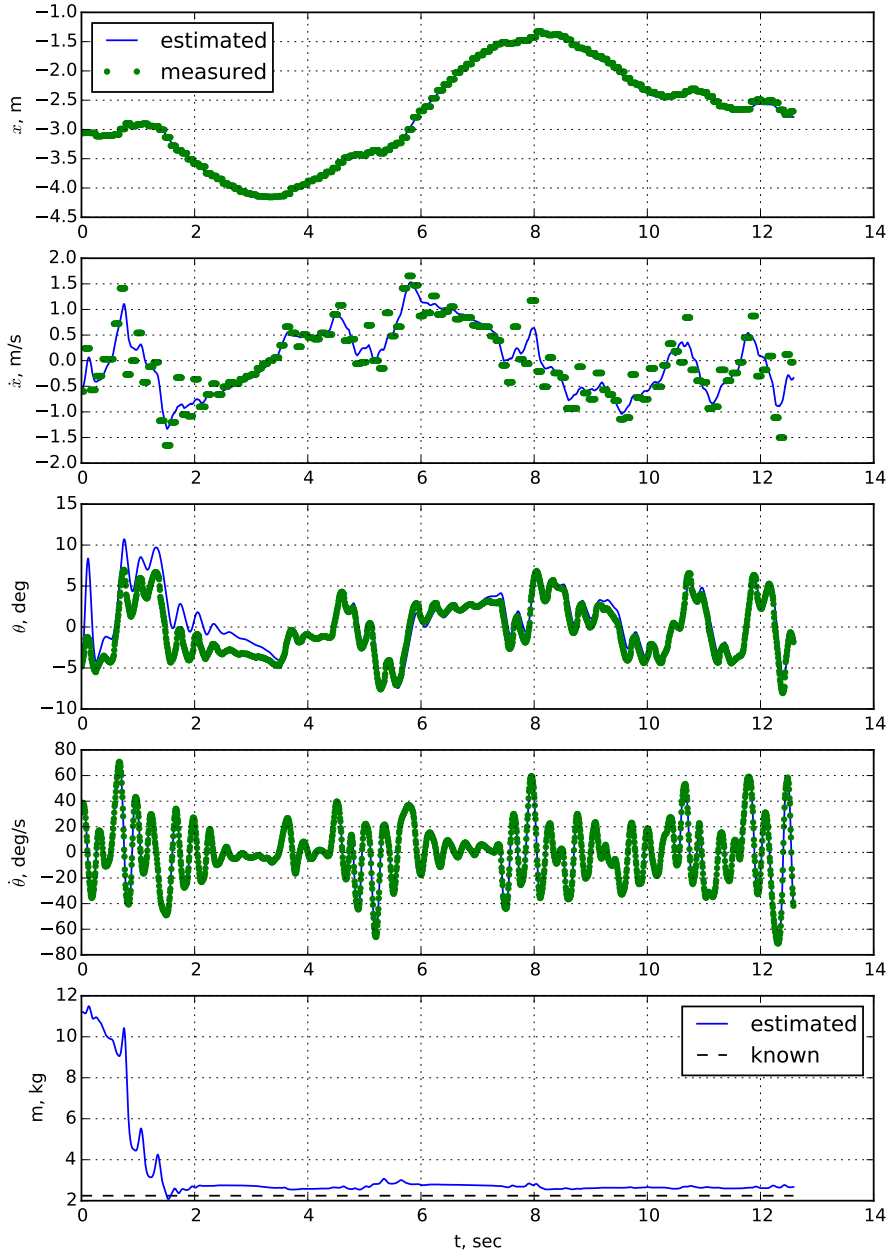
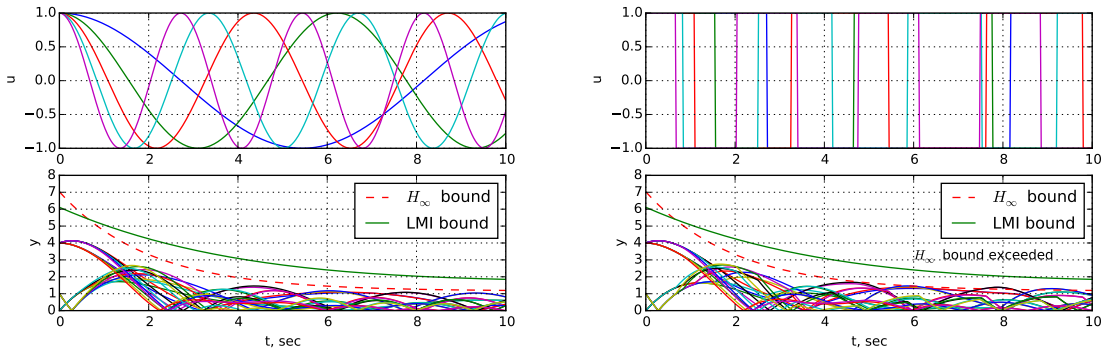


Fig. 4.5. Recursive system identification of robot mass using EKF algorithm.

4.4 Lyapunov based Approach to PIHA Model Checking

An alternative to the H_∞ approach for constructing a polyhedral invariant hybrid automaton, PIHA, is to use Lyapunov functions to construct invariant sets. The advantages of this method is that it can be applied to non-linear systems with bounded disturbances. The disadvantage is that it is often difficult to find a Lyapunov function for a given system and it is more computationally expensive. In Figure 4.6 it is clear



(a) sine input

(b) square input

Fig. 4.6. A comparison of H_∞ and LMI (Lyapunov) bounding for a linear system.

that the LMI based Lyapunov bound out performs the H_∞ bound. The assumptions of the H_∞ bound (steady state conditions, and sinusoidal input) are revealed in the results. The sinusoidal input is well bounded, but the bound is over-approximated initially. The H_∞ norm is not able to bound the square wave input.

4.5 Simplified Analysis of Inverted Pendulum Dynamics

In order to make the analysis more tractable, we will limit our analysis to the ($\mathbf{x} = \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^T$) subspace to demonstrate application of PIHA based model checking. This is also a reasonable approximation for slow speed. In addition, when dynamic inversion is applied, it will be possible to cancel the effects of the vehicle velocity on the pitch dynamics.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta} \\ \frac{1}{l^2 m} \left(glm \sin\left(\theta\right) + u \right) \end{bmatrix}$$

Since this is a mechanical system, the total energy is a Lyapunov function if we also consider energy storage in the controller:

$$V = glm \cos\left(\theta\right) + \frac{k_1}{2} \left(-\theta_0 + \theta \right)^2 + \frac{l^2 m}{2} \left(\dot{\theta} \right)^2$$

The regulated dynamics are stable for any $k_2 > 0$. Depending on the relative magnitude of k_1, the gravity torque, and the initial conditions, the equilibrium will either be at $\theta = 0$ or at $\theta = k\pi$, where k is an integer.

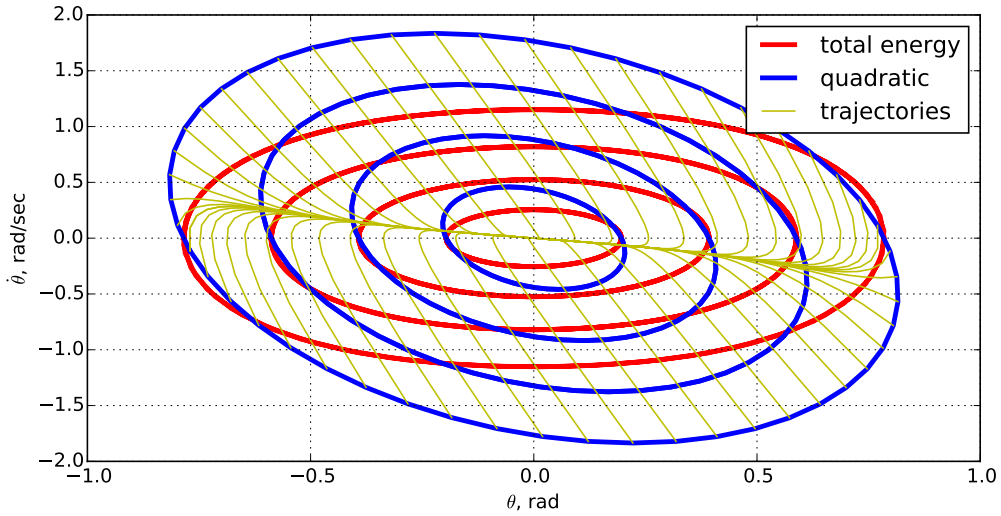
$$\dot{V} = \frac{\partial V}{\partial x} \frac{\partial x}{\partial t} = -k_2 \left(\dot{\theta} \right)^2$$

The system is linearized by adding a term that cancels the gravity torque to the input:

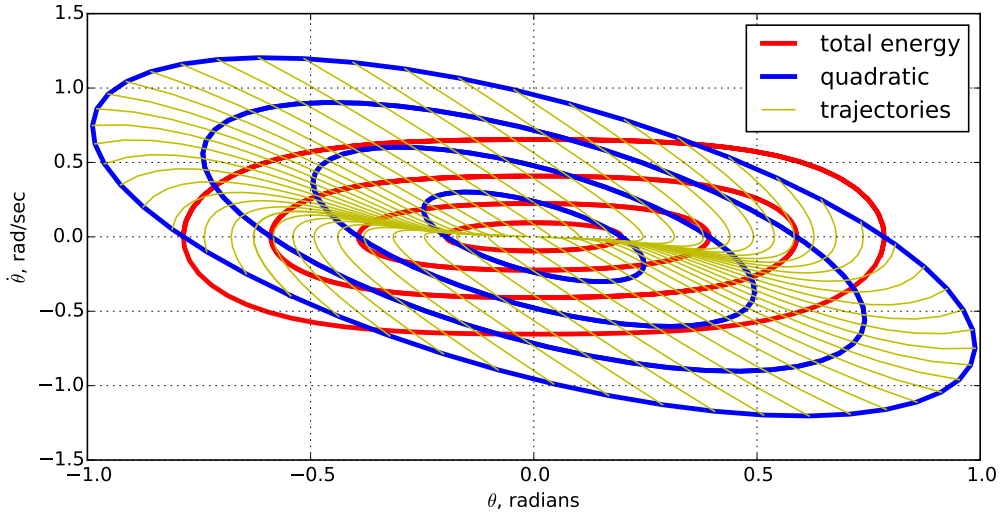
$$u = -glm \sin\left(\theta(t)\right) - k_1 \left(-\theta_0 + \theta(t) \right) - k_2 \frac{d}{dt} \theta(t)$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta} \\ -\frac{k_1}{l^2 m} (\theta(t) - \theta_0) - \frac{k_2}{l^2 m} \dot{\theta}(t) \end{bmatrix}$$

The resulting dynamics and Lyapunov levels set are shown in Figure 4.7. Note that for the LQR gains the quadratic approximation of the non-linear total energy Lyapunov function is closer than the case where the gains are limited.



(a) LQR controller k1=11.5, k2=5.7)



(b) Underpowered controller k1=10, k2=2)

Fig. 4.7. Lyapunov level sets for dynamic inversion regulated system.

4.6 Invariant Set Calculation via Lyapunov Method

The Lyapunov function can be used to find an attractive and invariant set. Once a state enters and invariant set it does not leave by definition. The invariant set can then be used to construct flow pipes and verify system safety as discussed in Chapter 3. The following theorem is from the AAE 666 class notes [50].

Theorem 4.6.1 *Consider a disturbed linear system:*

$$\dot{x} = Ax + Bw$$

$$z = Cx + Dw$$

where all eigenvalues of A have negative real part and w is a bounded input. Suppose there exists a positive real scalar α such that:

$$\begin{pmatrix} PA + A^T P + 2\alpha P & PB \\ B^T P & -2\alpha\mu_1 I \end{pmatrix} < 0 \tag{4.2}$$

$$\begin{pmatrix} C^T C - P & C^T D \\ D^T C & D^T D - \mu_2 I \end{pmatrix} < 0 \tag{4.3}$$

then

$$||y(t)||_\infty \leq \beta e^{-\alpha t} + \gamma ||u(t)||_\infty \leq \beta + \gamma ||u(t)||_\infty \tag{4.4}$$

where

$$\beta = x_0^T P x_0 \tag{4.5}$$

$$\gamma = \sqrt{\mu_1 + \mu_2} \tag{4.6}$$

Since α and P cannot both be a variable in the LMI in order for it to be linear, a line search must be performed to minimize γ by changing α. It is beneficial to

minimize γ since it is the bound at steady state while α represents the transient behavior.

The solution of the LMI problem was found using the PICOS python toolbox. This is a toolbox that interfaces to cvxopt. See Table 4.1 for the input and output to the program for the Lyapunov bounding problem..

Table 4.1.
Linear matrix inequality solution for the Lyapunov bounding problem using the PICOS solver with CVXOPT back-end.

```
line search
Optimization terminated successfully.
         Current function value: 1.687567
         Iterations: 19
         Function evaluations: 38
---------------------
optimization problem  (SDP):
5 variables, 0 affine constraints, 17 vars in 5 SD cones

P  : (2, 2), symmetric
mu_2  : (1, 1), continuous
mu_1  : (1, 1), continuous

minimize mu_1 + mu_2
such that
  [P*A + A.T*P + 2.0*alpha*P,P*B;B.T*P,( ( ( -2.0 )*alpha )*mu_1 )*I] <= |-1e-10|
  [C.T*C -P,C.T*D;D.T*C,D.T*D -mu_2*I] <= |-1e-10|
  P >= ( 1e-10 )*I
  mu_1 >= 1e-10
  mu_2 >= 1e-10
---------------------
optimal alpha:  [ 0.20525]
gamma:  1.68756728435
mu_1:  2.84788333291
mu_2:  6.30723574201e-09
P:  [ 1.21e+00  2.83e-01]
[ 2.83e-01  1.38e+00]
```

Now that we have tools to bound the output of linear systems with bounded inputs, we can develop a generic algorithm to calculate flow pipes for linear systems and create a PIHA to facilitate model checking for a system switching between different operating conditions. See Algorithm1 below.

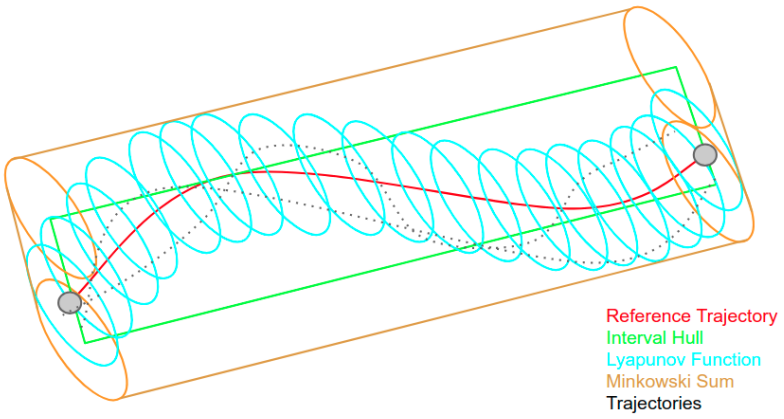


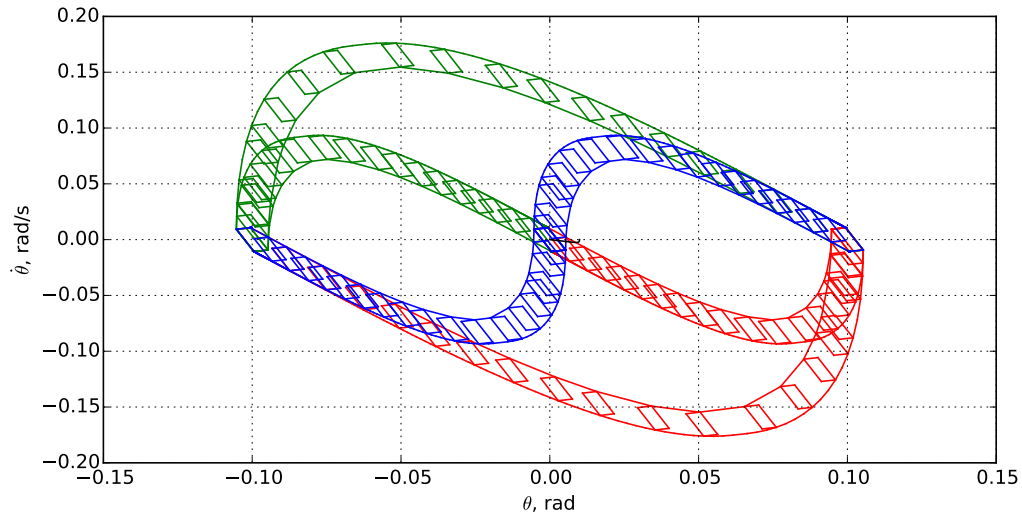
Fig. 4.8. Construction of Lyapunov based flow pipe

Algorithm 1 Lyapunov flow pipe computation

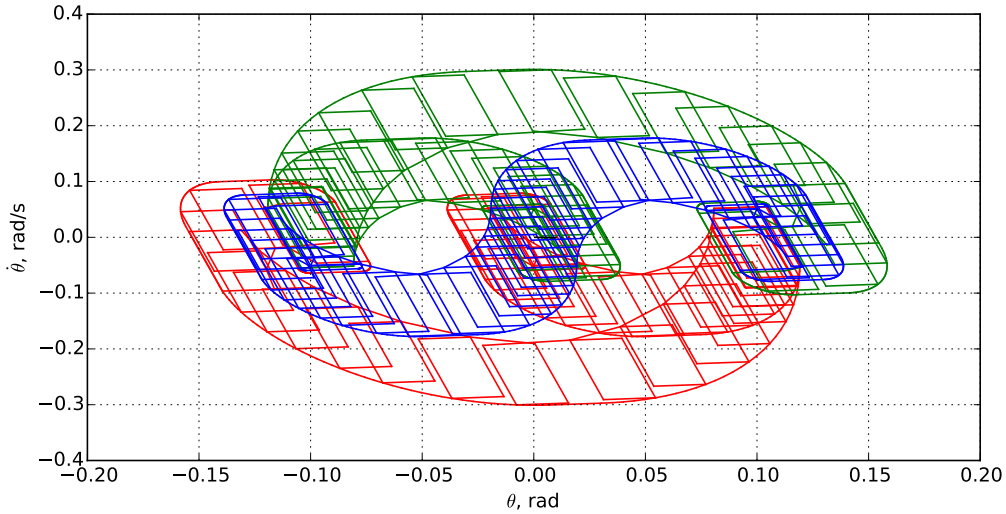
1. Find the invariant set for a linear system with bounded input using the LMI given in Theorem 4.6.1.
2. Propagate the reference trajectory for the system for a fixed time interval.
3. Compute the interval hull (smallest box enclosing the set) of the reference trajectory for the propagation duration.
4. Compute the convex hull for the flow pipe segment as the Minkowski sum of the interval hull and the invariant set.
5. Move to the next time interval and repeat the process.

The output of this algorithm for our simple system is shown in Figure 4.9. Note for the LQR gains the flow pipes are small. For the underpowered case, the flow pipes

swirl into the new operating point after the mode change. Also the invariant sets at each operating point are larger due to the increased ratio of disturbance to control input.



(a) LQR Gains K=(11.5, 5.7)



(b) Underpowered Controller K=(10,2)

Fig. 4.9. Bisimulations for pendulum dynamics switching between 3 operating points $\theta = (-0.1, 0, 0.1)$ rad.

One advantage of constructing the flow pipes as a series of convex hulls is that it is efficient to check if a point is inside or outside the flow pipe. This is accomplished by checking if the point is inside any of the convex hulls or not. In addition, the flow pipes can be efficiently expanded by changing the offset coefficient in the convex hulls. This allows expanding the convex hull to account for unknown disturbances. The system could then recheck the safety of the internal logic and dynamics with the new disturbance level and react accordingly.

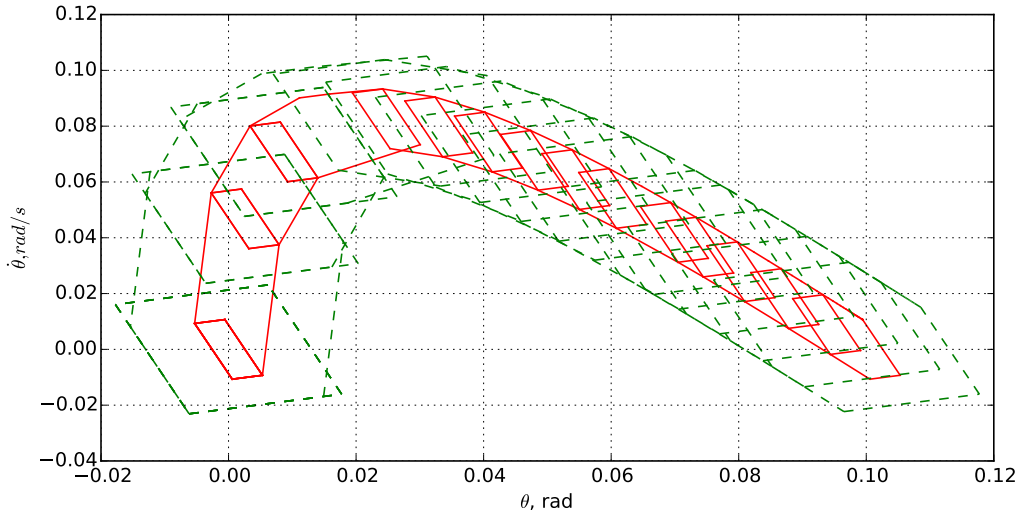


Fig. 4.10. Flow pipes are constructed from a series of convex hulls and can be efficiently expanded to account for unexpected disturbances.

4.7 Conclusion

In this chapter, we have developed the concept of Lyapunov based flow pipes to address the non-linearities in the two-wheel self-balancing robot. This method was used to create a PIHA based model of the system. In addition, an EKF online system identification algorithm was created that can be coupled with PIHA model checking as discussed in Chapter 3 for adaptive planning.

5. CASE STUDY: QUADROTOR WITH OPTICAL FLOW BASED VISUAL ODOMETRY

Fig. 5.1. Tarot Peeper quadrotor with custom 3D printed onboard computer canopy, prop guards, and optical flow sensor.

Many UASs have limited payload capacity due to their small size. Because of these payload restrictions, the sensing and computational payload must be lightweight, and sensors for such systems are generally limited to Inertial Measurement Units (IMUs) and GPS receivers, sometimes with the addition of altimeters and cameras. Increased UAS diversity and usage has seen the potential applications of UASs expand into varied tasks such as search and rescue, infrastructure inspection, and agriculture [51–53] . In many UAS use cases, increased accuracy for state estimation is desirable or required for autonomous or semi-autonomous operation. Some results of this chapter were published in the paper [54].

The standard set of sensors used to gather state information (IMU, GPS, magnetometer and barometer), is an effective combination and is accurate to within a few

centimeters when GPS is available. In many environments and situations, however, GPS signals can be unreliable or absent. In GPS-denied situations, the standard sets of sensors will accumulate error over time due mainly to drift in IMU that goes uncorrected by the ground truth position and velocity from the now-absent GPS. Because UASs operate in a range of different environments with varied GPS availability, it is sometimes necessary to have an accurate estimation of the state from an information source outside this usual set of sensors that can function when GPS is unavailable.

Optical flow measurements have been used for some situations where GPS is unavailable for a UAS, especially indoors. In a confined indoor environment, practically attainable speeds are generally much slower than in more open outdoor areas. This is acceptable because aircraft must fly closer to the ground indoors, and ground-facing optical flow sensors necessitate slower velocities the closer a UAS is to the ground. Ground-facing optical flow measurements require speed to be below a certain threshold depending on altitude so that features on the ground do not leave the field of view between successive images. Indoor flying also has the advantage of more uniform surfaces and less shadow in general than outdoor flying, so the error of optical flow measurements will generally be more consistent inside than outside.

The motivation for this work stems from a need for accurate navigation of a UAS in the absence of GPS measurements. This position is used to prevent errors in IMU acceleration measurements from accumulating over the course of the mission. Without such measurements, the UAS's estimate of its position diverges from the actual position, which can cause increasingly degraded navigation performance.

Optical flow based UAS navigation systems have not flown outdoors with sufficient accuracy to replace GPS over the duration of a typical mission (several minutes to tens of minutes). We improve optical flow based outdoor navigation performance by improving the estimation algorithms used in an optical flow based UAS navigation system, allowing the UAS to operate outdoors and at higher speeds because it can obtain a more accurate state estimate than similar systems in the absence of GPS data.

The contributions of this work are a robust, light-weight, and low cost optical flow based navigation solution that features an Invariant Extended Kalman Filter (IEKF) and enables flying outdoor missions without GPS. Our system can also be used as a backup layer in more complex systems with active visual odometry, given that the computational demand of this layer can be scaled for processors from the STM32F4 (as on the PX4Flow) up to high-end systems such as multi-core ARM processors.

We demonstrate the improved accuracy of the optical flow based navigation techniques through the simulation of two outdoor flight scenarios of a quadrotor equipped with an optical flow unit, laser altimeter, and updated software with IEKF, without using GPS data. We also compare the performance of the system running our estimation algorithms to the performance of the same system running our previous estimation algorithm that was able to first achieve optical flow based mission flight, LPE. LPE is a local position estimator that is coupled with a non-linear and independent quaternion based attitude estimator.

The rest of the chapter is organized as follows. Section 5.1 discusses past work related to the work we present. Section 5.2 gives details about the estimation and filtering techniques used in this work. Section 5.3 describes the experimental setup and provides a discussion of the experimental results. Section 5.4 analyzes the stability properties. Lastly, Section 5.5 presents conclusions and future work.

5.1 Related Work

This work combines IEKF estimation and optical flow based navigation techniques, and this section gives an overview of past work related to both of these areas.

An invariant extended Kalman filter takes advantage of the symmetries of a system so that the estimation error follows a stochastic differential equation independent of the system's trajectory as long as the system is group affine [55, 56]. Due to the bias states, our system is not group affine, but the small magnitude of the bias error make this approach useful in practice [57].

For a nonlinear system with symmetries, the IEKF uses a geometrically adapted correction term based on invariant output error instead of a linear correction term based on linear output error [58]. Similarly, the gain matrix is updated using an invariant state error rather than a linear state error. The gain of the covariance matrix converges to constant values in a much larger set of trajectories in the invariant case than in the case of the EKF, where convergence is guaranteed for the equilibrium points only. This larger domain of convergence results in more accurate estimation in general. The IEKF has been applied to an attitude estimation problem for a moving rigid body, which is an analog for a UAS, with GPS, inertial, and magnetic measurements [58].

Ground-facing optical flow navigation uses images from a ground-facing camera to track the relative motion of the UAS and the surrounding environment. Combined with rotation information and the distance to the ground, optical flow measurements can be used to calculate the velocity of the vehicle. Some of the earliest optical flow sensors took advantage of the sensors from optical computer mice [59], but more advanced hardware exists that can be implemented in an optical flow sensor to reduce the drift present in the early systems [60]. To deal with the high computational demands of optical flow computations for mobile robotic systems, specialized standalone optical flow systems have been developed and optimized for robotic navigation using improved sensors and processors [61, 62].

5.2 Estimation

The estimation algorithm we employ considers the kinematics, including the rotation and translation of the UAS, without considering the forces acting on it. The rotational kinematics are described by the first order differential equation for the time derivative of the quaternion, $\dot{\boldsymbol{q}}_{nb}$, representing the rotation from the body frame to the navigation frame. The translational kinematics are expressed in the navigation

frame, with the acceleration given as a function of the acceleration due to gravity, $-g\hat{n}_z$, and the acceleration in the body frame, $\boldsymbol{a}_b$.

$$\dot{\boldsymbol{q}}_{nb} = \frac{1}{2}\boldsymbol{q}_{nb} \otimes {}^{n}\boldsymbol{\omega}_{b}^{b} \tag{5.1}$$

$${}^{n}\dot{\boldsymbol{V}}_{n}^{P} = -g\hat{n}_z + \boldsymbol{q}_{nb} \otimes \boldsymbol{a}_b \otimes \boldsymbol{q}_{nb}^{-1} \tag{5.2}$$

$$\dot{\boldsymbol{R}}_{n}^{OP} = {}^{n}\boldsymbol{V}_{n}^{P} \tag{5.3}$$

The gyroscope measurement, $\boldsymbol{y}^g$, consists of the actual angular velocity of the body frame relative to the navigation frame, $\boldsymbol{y}^g$, plus a bias term, $\boldsymbol{b}^g$. The bias term is modeled as a first order differential equation where the time derivative of the bias is composed of a deterministic and a stochastic component. the stochastic component is assumed to be zero mean Gaussian, $\mathcal{N}(0, \sigma_g^2)$.

$$\boldsymbol{y}^g = {}^{n}\boldsymbol{\omega}^{b} + \boldsymbol{b}^g \tag{5.4}$$

$$\dot{\boldsymbol{b}}^g = -\tau_g \boldsymbol{b}^g + \boldsymbol{v}^g \tag{5.5}$$

$$\boldsymbol{v}^g \sim \mathcal{N}(0, \sigma_g^2) \tag{5.6}$$

The accelerometer is modeled in a similar manner to the gyroscope. The measurement, $\boldsymbol{y}^a$, is composed of the actual acceleration of the UAS relative to the navigation frame, $\boldsymbol{a}_b$, plus a bias term. The bias term is modeled as having its time derivative equal to the sum of a deterministic and stochastic term, where the stochastic component is assumed to be zero mean Gaussian, $\mathcal{N}(0, \sigma_a^2)$.

$$\boldsymbol{y}^a = \boldsymbol{a}_b + \boldsymbol{b}^a \tag{5.7}$$

$$\dot{\boldsymbol{b}}^a = -\tau_a \boldsymbol{b}^a + \boldsymbol{w}^a \tag{5.8}$$

$$\boldsymbol{w}^a \sim \mathcal{N}(0, \sigma_a^2) \tag{5.9}$$

The optical image is centered at a point distance d below the vehicle camera, aligned with the camera direction $\hat{b}_z$. If we compute the velocity of this vector in the

body frame we can derive an approximate equation for the optical flow of the image assuming a small image field of view, see Figure 5.2.

$$\boldsymbol{R}^{PG} = d\hat{b}_z \tag{5.10}$$

$${}^{n}\boldsymbol{V}^{PG} = {}^{b}\boldsymbol{V}^{PG} + {}^{n}\boldsymbol{\omega}^{b} \times \boldsymbol{R}^{PG} \tag{5.11}$$

$${}^{n}\boldsymbol{V}^{PG} = {}^{n}\boldsymbol{V}^{G} - {}^{n}\boldsymbol{V}^{P} \tag{5.12}$$

We note that ground velocity is zero in the navigation frame.

$${}^{n}\boldsymbol{V}^{G} = \boldsymbol{0} \tag{5.13}$$

This yields:

$${}^{b}\boldsymbol{V}^{PG} = -{}^{n}\boldsymbol{V}^{P} - {}^{n}\boldsymbol{\omega}^{b} \times d\hat{b}_z \tag{5.14}$$

which is used to compute the measurement equations for optical flow below where we take the dot product of the velocity with $\hat{b}_x$ and $\hat{b}_y$, assuming the camera is aligned with the body:

$$\boldsymbol{y}_b^f = \begin{bmatrix} {}^{b}\boldsymbol{V}_b^{PG} \circ \hat{b}_x \\ {}^{b}\boldsymbol{V}_b^{PG} \circ \hat{b}_y \end{bmatrix} \tag{5.15}$$

The state vector is composed of the quaternion describing the rotation from the body frame to the navigation frame (4 states), the velocity in the navigation frame (3 states), the position in the navigation frame (3 states), the gyroscope bias in the body frame (3 states), and the accelerometer bias in the body frame (3 states). The input vector consists of the accelerometer and gyroscope measurements in the body frame. These measurements are treated as input to the kinematic model and used for prediction. This is a typical approach in strap-down inertial navigation and is widely used since it doesn't require a dynamic model of the vehicle for prediction.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_{nb} & {}^{n}\boldsymbol{V}_n^P & \boldsymbol{R}_n^{OP} & \boldsymbol{b}_b^g & \boldsymbol{b}_b^a \end{bmatrix}^T \tag{5.16}$$

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{y}_b^a & \boldsymbol{y}_b^g \end{bmatrix}^T \tag{5.17}$$

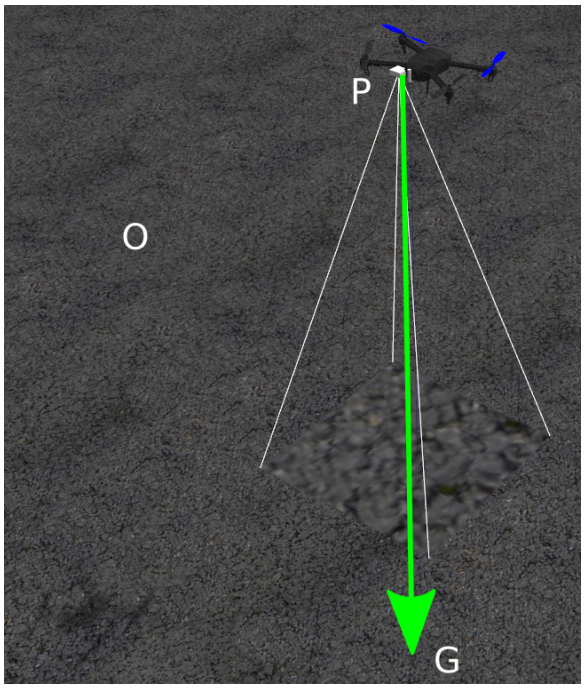


Fig. 5.2. A diagram of the vehicle, shown in the simple asphalt world.

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{w} \tag{5.18}$$

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \frac{1}{2}\hat{\boldsymbol{q}}_{nb} \otimes (\boldsymbol{y}_b^g - \boldsymbol{b}_b^g) \\ -g\hat{n}_z + \hat{\boldsymbol{q}}_{nb} \otimes (\boldsymbol{y}_b^a - \boldsymbol{b}_b^a) \otimes \hat{\boldsymbol{q}}_{nb}^{-1} \\ {}^n\hat{\boldsymbol{V}}_n^P \\ -\tau_g \boldsymbol{b}_b^g \\ -\tau_a \boldsymbol{b}_b^a \end{bmatrix} \tag{5.19}$$

$$\boldsymbol{w} \sim \mathcal{N}(0, Q) \tag{5.20}$$

The functions $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ and $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}))$ can be expressed in terms of an error state from a reference state. This simplifies the Jacobian matrices, allowing them to be constant along permanent trajectories if the frame of linearization is chosen wisely, as described in [55]. Note that $\hat{\boldsymbol{\gamma}}$ is incremental rotation angle error from the navigation frame. The benefit of this can be seen clearly in the difference between linearization in the body frame in Figure 5.3 and linearization in the navigation frame in Figure 5.4.

$$\boldsymbol{e} = \begin{bmatrix} \hat{\boldsymbol{\gamma}} - \boldsymbol{\gamma} \\ {}^n\hat{\boldsymbol{V}}_n^P - {}^n\boldsymbol{V}_n^P \\ \hat{\boldsymbol{R}}_n^{OP} - \boldsymbol{R}_n^{OP} \\ \hat{\boldsymbol{b}}_b^g - \boldsymbol{b}_b^g \\ \hat{\boldsymbol{b}}_b^a - \boldsymbol{b}_b^a \end{bmatrix}^T \tag{5.21}$$

The error dynamics are then linearized to produce the required matrices for the continuous prediction and discrete update Kalman filter. The continuous prediction step is implemented on a digital compute, so it is not truly continuous, but is considered as such as it is run at 250 Hz while most of the measurements arrive asynchronously and at a much lower rate. When the measurements arrive, z_k, the standard discrete time Kalman filter correction is used.

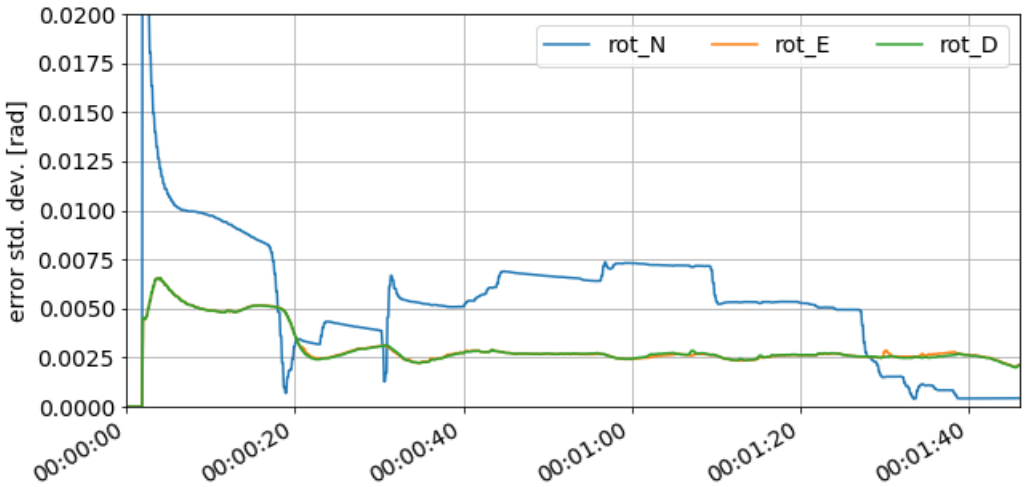


Fig. 5.3. Rotation uncertainty standard deviation, sample from PX4 EKF2 estimator (LPE cannot be compared since it had a separate attitude estimator).

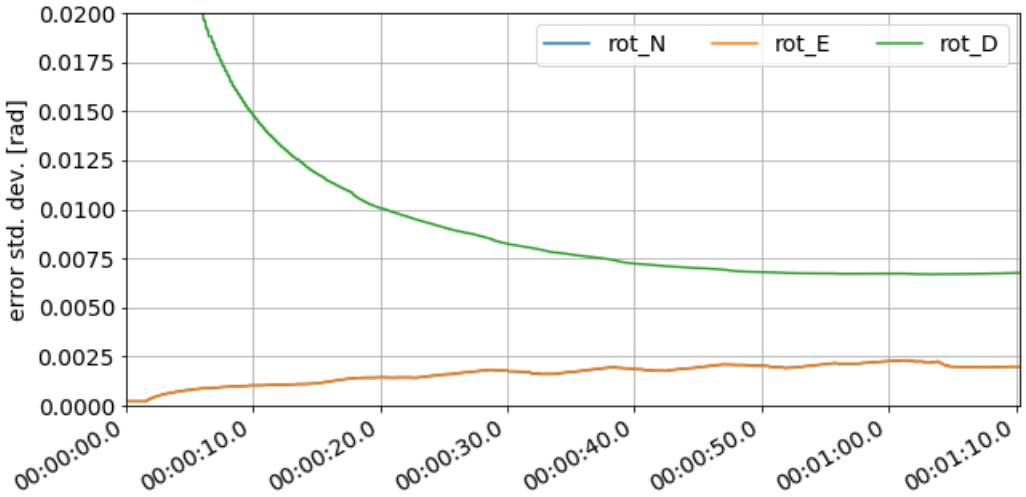


Fig. 5.4. Rotation uncertainty standard deviation for invariant (IEKF) filter. Note that the response is primarily a function of time and approaches a steady-state independent of the trajectory.

$$F = \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{e}}, \boldsymbol{u}))}{\partial \hat{\boldsymbol{x}}} \qquad B = \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{e}}, \boldsymbol{u}))}{\partial \boldsymbol{u}} \tag{5.22}$$

$$H = \frac{\partial \boldsymbol{g}(\hat{\boldsymbol{e}}, \boldsymbol{u}))}{\partial \hat{\boldsymbol{x}}} \qquad D = \frac{\partial \boldsymbol{g}(\hat{\boldsymbol{e}}, \boldsymbol{u}))}{\partial \hat{\boldsymbol{x}}} \tag{5.23}$$

$$\dot{\hat{\boldsymbol{x}}}(t) = \boldsymbol{f}(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) \tag{5.24}$$

$$\hat{\boldsymbol{x}}_{k|k-1} = \hat{\boldsymbol{x}}_{k-1|k-1} + \int_{t_1}^{t_2} \boldsymbol{f}(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t))dt \tag{5.25}$$

$$\dot{P}(t) = FP(t) + P(t)F^T + Q(t) \tag{5.26}$$

$$P_{k|k-1} = P_{k-1|k-1} + \int_{t_1}^{t_2} \dot{P}(t)dt \tag{5.27}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{5.28}$$

$$K_k = P_k H_k^T S_k^{-1} \tag{5.29}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{5.30}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + K_k(z_k - h(\hat{\boldsymbol{x}}_{k|k-1})) \tag{5.31}$$

The system is given as:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, +)M(\boldsymbol{x})w \tag{5.32}$$

$$y = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, +)N(\boldsymbol{x})v \tag{5.33}$$

We consider the following non-linear stochastic dynamic system:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + M(\boldsymbol{x})\boldsymbol{w}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) + N(\boldsymbol{x})\boldsymbol{v}$$

$$\boldsymbol{w} \sim \mathcal{N}(0, 1)$$

$$\boldsymbol{v} \sim \mathcal{N}(0, 1)$$

Consider a Lie group ($g \in G$) with local transformation groups $\varphi_g, \psi_g, \rho_g$ and identity element $\boldsymbol{I}_g$, such that $\boldsymbol{X} = \varphi_g(\boldsymbol{x})$, $\boldsymbol{U} = \psi_g(\boldsymbol{u})$, and $\boldsymbol{Y} = \rho_g(\boldsymbol{y})$. The non-linear stochastic system above is said to be invariant if it is unchanged by the transformation groups and the following transformed system of equations is valid.

$$\dot{\boldsymbol{X}} = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}) + M(\boldsymbol{X})\boldsymbol{w}$$

$$\boldsymbol{Y} = \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{U}) + N(\boldsymbol{X})\boldsymbol{v}$$

The continuous-discrete form of the IEKF prediction step is given below. Note that the covariance propagation is only dependent on the estimated invariant $\hat{\boldsymbol{J}}(\hat{\boldsymbol{x}}, \boldsymbol{u})$, which is the main reason that convergence properties are improved over a standard EKF.

$$\hat{\boldsymbol{x}}_{k|k-1} = \hat{\boldsymbol{x}}_{k-1|k-1} + \int_{t_{k-1}}^{t_k} f(\hat{\boldsymbol{x}}, \boldsymbol{u})dt$$

$$F(\hat{\boldsymbol{J}}) = \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}, \boldsymbol{u})}{\partial \hat{\boldsymbol{x}}}$$

$$P_{k|k-1} = P_{k-1|k-1} + \int_{t_{k-1}}^{t_k} \left(F(\hat{\boldsymbol{J}})P + PF^T(\hat{\boldsymbol{J}}) + M(\boldsymbol{I}_g)M(\boldsymbol{I}_g)^T \right) dt$$

The continuous-discrete form of the IEKF correction step is then given below. Note also that the correction step is only dependent on the estimated invariant $\hat{\boldsymbol{J}}(\hat{\boldsymbol{x}}, \boldsymbol{u})$ and the estimated invariant error $\hat{\boldsymbol{E}}$.

$$
\begin{aligned}
H_k(\hat{\boldsymbol{J}}) &= \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}, \boldsymbol{u})}{\partial \hat{\boldsymbol{x}}} \\
S_k &= H_k P_{k|k-1} H_k^T + N(\boldsymbol{I}_g) N^T(\boldsymbol{I}_g) \\
K_k &= P_{k|k-1} H_k^T S_k^{-1} \\
P_{k|k} &= (I - K_k H_k) P_{k|k-1} \\
\hat{\boldsymbol{E}} &= \rho_{\hat{\boldsymbol{x}}^{-1}}(\boldsymbol{y}) - \rho_{\hat{\boldsymbol{x}}^{-1}}(\boldsymbol{h}(\hat{\boldsymbol{x}}, \boldsymbol{u})) \\
\boldsymbol{x}_{k|k} &= \boldsymbol{x}_{k|k-1} + D\varphi_g(\boldsymbol{I}_g) K_k \hat{\boldsymbol{E}}
\end{aligned}
$$

$$
\begin{aligned}
\dot{\boldsymbol{q}}_{nb} &= \frac{1}{2} \boldsymbol{q}_{nb} \otimes {}^n\boldsymbol{\omega}_b^b \\
{}^n\dot{\boldsymbol{V}}_n^P &= -g\hat{n}_z + \boldsymbol{q}_{nb} \otimes \boldsymbol{a}_b \otimes \boldsymbol{q}_{nb}^{-1}
\end{aligned}
$$

$$
\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_{nb} & {}^n\boldsymbol{V}_n^P & \boldsymbol{b}_b^g & \boldsymbol{b}_b^a \end{bmatrix}^T \tag{5.34}
$$

$$
\boldsymbol{u} = \begin{bmatrix} \boldsymbol{y}_b^a & \boldsymbol{y}_b^g \end{bmatrix}^T \tag{5.35}
$$

The state space can be considered a group with the operator. Note that this can be considered at rotation in the earth frame, translation in the body frame, angular velocity in the body frame, and the acceleration in the body frame.

$$
\varphi_(q_0, V_0, \omega_0, a_0) \begin{pmatrix} q \\ V \\ \omega_b \\ a_b \end{pmatrix} = \begin{pmatrix} q \otimes q_0 \\ V + V_0 \\ q_0^{-1} \omega_b q_0 + \omega_0 \\ q_0^{-1} a_b q_0 + a_0 \end{pmatrix}
$$

The following transformation groups may also be defined. Note we define an accelerometer bias instead of accelerometer scale factor as this was more appropriate for our sensor error model.

$$\Psi_{(q_0,V_0,\omega_0,a_0)} \begin{pmatrix} \omega_m \\ a_m \end{pmatrix} = \begin{pmatrix} q_0^{-1} \otimes \omega_m \otimes q_0 + \omega_0 \\ q_0^{-1} \otimes a_m \otimes q_0 + a_0 \end{pmatrix}$$

$$\rho_{(q_0,V_0,\omega_0,a_0)} \begin{pmatrix} y_V \\ y_B \end{pmatrix} = \begin{pmatrix} y_V + V_0 \\ q_0^{-1} \otimes y_B \otimes q_0 \end{pmatrix}$$

The invariants are given by $\Psi_{\hat{\boldsymbol{x}}^{-1}}(u)$, where $\hat{\boldsymbol{x}}^{-1} = (\hat{\boldsymbol{q}}_{nb}^{-1}, -{}^n\hat{\boldsymbol{V}}_n^P, -{}^n\hat{\boldsymbol{\omega}}_n^b, -\hat{\boldsymbol{q}}_{nb}^{-1} \otimes \hat{\boldsymbol{b}}_b^g \otimes \hat{\boldsymbol{q}}_{nb}, -\hat{\boldsymbol{b}}_b^a)$

$$\begin{pmatrix} \tilde{J}_w \\ \tilde{J}_a \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{q}}_{nb} \otimes (\boldsymbol{y}_b^g - \hat{\boldsymbol{b}}_b^g) \otimes \hat{\boldsymbol{q}}_{nb}^{-1} \\ \hat{\boldsymbol{q}}_{nb} \otimes (\boldsymbol{y}_b^a - \hat{\boldsymbol{b}}_b^a) \otimes \hat{\boldsymbol{q}}_{nb}^{-1} \end{pmatrix}$$

The invariant output error is given by:

$$\boldsymbol{E} = \rho_{\hat{\boldsymbol{x}}^{-1}} \begin{pmatrix} \hat{\boldsymbol{y}}_b^f \\ \hat{\boldsymbol{y}}_b^m \end{pmatrix} - \rho_{\hat{\boldsymbol{x}}^{-1}} \begin{pmatrix} \boldsymbol{y}_b^f \\ \boldsymbol{y}_b^m \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{y}}_b^f - \boldsymbol{y}_b^f \\ B_n - \hat{\boldsymbol{q}}_{nb} \otimes \boldsymbol{y}_b^m \otimes \hat{\boldsymbol{q}}_{nb} \end{pmatrix}$$

The invariant state error $\boldsymbol{\eta} = \boldsymbol{x}^{-1}\hat{\boldsymbol{x}}$ is given by:

$$\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\nu} \\ \boldsymbol{\beta} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{q}}_{nb} \otimes \boldsymbol{q}_{nb}^{-1} \\ {}^n\hat{\boldsymbol{V}}_b^P - {}^n\boldsymbol{V}_b^P \\ \boldsymbol{q}_{nb} \otimes (\hat{\boldsymbol{b}}_b^g - \boldsymbol{b}_b^g) \otimes \boldsymbol{q}_{nb}^{-1} \\ \boldsymbol{q}_{nb} \otimes (\hat{\boldsymbol{b}}_b^a - \boldsymbol{b}_b^a) \otimes \boldsymbol{q}_{nb}^{-1} \end{pmatrix}$$

When linearized about the group identity, (1, 0, 0, 1) and dropping all quadratic terms, we obtain the following state space representation:

$$F = \begin{pmatrix} 0_{33} & 0_{33} & -\frac{1}{2}I_3 & 0_{33} \\ -2\tilde{J}_a\times & 0_{33} & 0_{33} & -\frac{1}{2}I_3 \\ 0_{33} & 0_{33} & \tilde{J}_\omega\times & 0_{33} \\ 0_{33} & 0_{33} & 0_{33} & \tilde{J}_\omega\times \end{pmatrix}$$

$$H = \begin{pmatrix} 0_{33} & I_{33} & 0_{33} & 0_{33} \\ 2B_n\times & 0_{33} & 0_{33} & 0_{33} \end{pmatrix}$$

$$M = Diag(M_q, M_V, M_\omega, M_a)$$

$$N = Diag(N_f, N_m)$$

$$K = -(K_q, K_V, K_\omega, K_a)^T$$

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_{nb} & {}^n\boldsymbol{V}_n^P & \boldsymbol{b}_b^g & \boldsymbol{b}_b^a \end{bmatrix}^T \tag{5.36}$$

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{y}_b^a & \boldsymbol{y}_b^g \end{bmatrix}^T \tag{5.37}$$

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) + M(\boldsymbol{x})\boldsymbol{w} \tag{5.38}$$

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \frac{1}{2}\hat{\boldsymbol{q}}_{nb} \otimes (\boldsymbol{y}_b^g - \boldsymbol{b}_b^g) \\ -g\hat{n}_z + \hat{\boldsymbol{q}}_{nb} \otimes (\boldsymbol{y}_b^a - \boldsymbol{b}_b^a) \otimes \hat{\boldsymbol{q}}_{nb}^{-1} \\ 0 \\ 0 \end{bmatrix} \tag{5.39}$$

$$\boldsymbol{w} \sim \mathcal{N}(0, 1) \tag{5.40}$$

$$\tag{5.41}$$

5.3 Simulation

In this section we investigate the performance of the IEKF in two simulation scenarios: flying over asphalt and flying over a forest. First, we give an overview of the UAS used in this simulation and its various components. The airframe is the IRIS+ quadcopter from 3D Robotics modified with a PX4flow optical flow unit and SF10-A laser altimeter, both mounted on the bottom of the airframe facing the ground. The IRIS+ has a Pixhawk autopilot running the PX4 autopilot software, and a GPS receiver. The open source autopilot software is modified to include our implementation of the IEKF.

The Pixhawk's integrated sensors include a 3-axis combination accelerometer and gyroscope, an additional gyroscope, a combination accelerometer and magnetometer, and a barometer. For ground-facing optical flow measurements we use the PX4Flow [62], an open source software and hardware optical flow camera and computation system. The sensor combines a sonar range finder, a machine vision CMOS sensor, and a gyroscope. The optical flow processing is performed on a 4x4 binned image at 400 Hz. The increased light sensitivity aids in outdoor performance as well as performance in lower light indoor environments.

Using data from the gyro to account for rotation and data from the altimeter to give distance from the ground, the PX4Flow performs optical flow calculations which result in velocities in the forward and rightward directions. Specifically, the PX4Flow calculates rotation compensated ground velocities, non rotation compensated flow measurements, the altitude measurement and a measure of quality of the measurements.

For accurate altitude measurements we use the Lightware SF10-A laser altimeter, which works by emitting a laser pulse and measuring the time it takes the pulse to reach the ground and bounce back. It is effective from 0 to 25 meters with a resolution of 1 cm at 32 Hz. It can be directly integrated into the Pixhawk via serial connection.

The software in the loop (SITL) simulation environment reproduces this configuration with high fidelity, incorporating realistic aircraft and sensor models and running the same PX4 firmware that runs on the actual Pixhawk hardware.

In both scenarios we compare the results of the default PX4 estimator to the IEKF described in this chapter. The PX4's default estimator is called the Local Position Estimator (LPE), which is an implementation of the Extended Kalman filter.

5.3.1 Asphalt World

The asphalt world consists of a flat, level, square piece of ground with an asphalt-like texture. Figure 5.2 shows the quad rotor model flying above the asphalt plane with a gray background. The scenario for this environment is a flight through five waypoints, which is depicted in Figures 5.5 and 5.6 for the LPE estimator and IEFK estimator, respectively. We note that from comparing the figures, the IEKF follows the desired trajectory more closely than LPE. Table 5.1 compares the errors and standard deviations of the state estimates form the LPE and IEKF, showing the IEKF has superior performance to the LPE overall.

5.3.2 Forest World

The forest world consists of a randomly generated lightly-hilled terrain with patches of dirt and grass, and several types of trees. Figure 5.7 shows a top-down view of the forest simulation environment, and Figure 5.8 shows a view of the same environment from closer to the ground. Similarly to the asphalt scenario, the scenario for the forest environment is a flight through five waypoints, depicted in Figures 5.9 and 5.10 for the LPE estimator and IEFK estimator, respectively. As in the asphalt case, we note that from comparing the figures the IEKF performs much more closely to the desired trajectory than the LPE. Table 5.1 compares the errors and standard deviations of the state estimates form the LPE and IEKF, showing the IEKF has superior performance to the LPE overall.

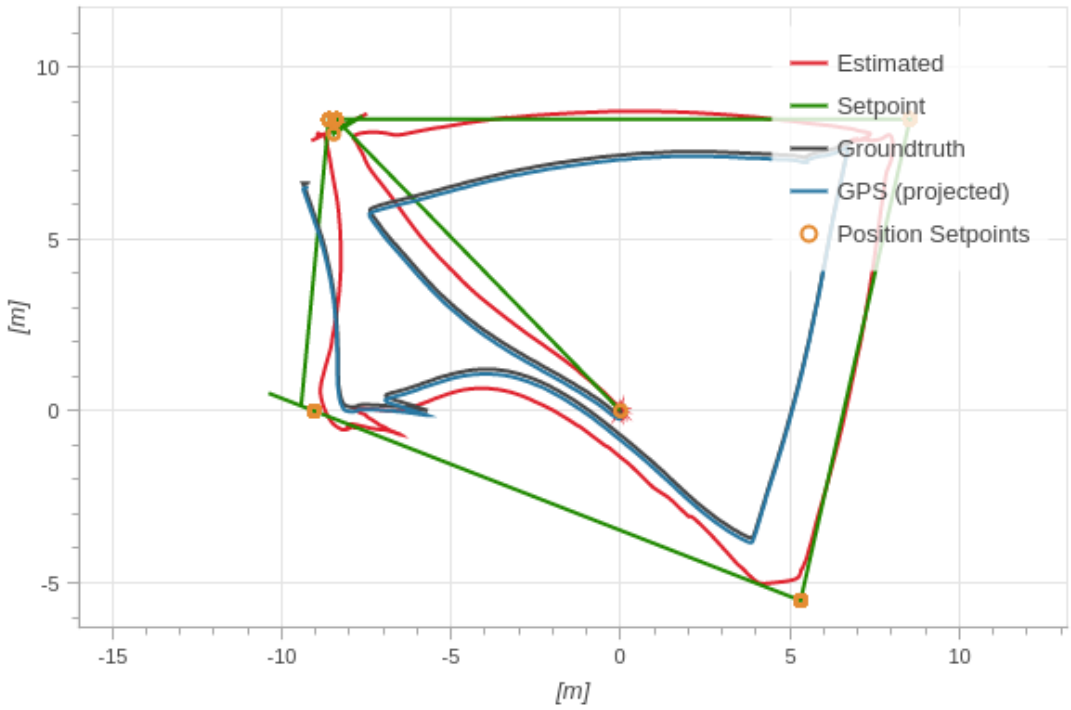


Fig. 5.5. Position estimation performance of LPE in asphalt world.

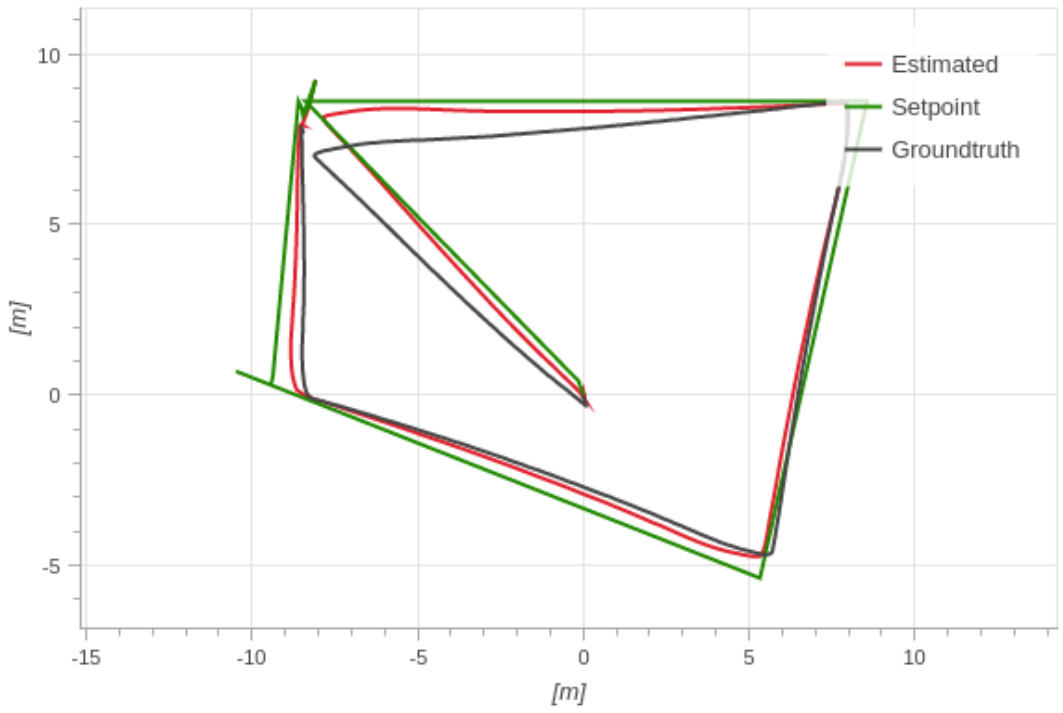


Fig. 5.6. Position estimation performance of IEKF in asphalt world.

Fig. 5.7. The randomly generated forest environment as seen from above.

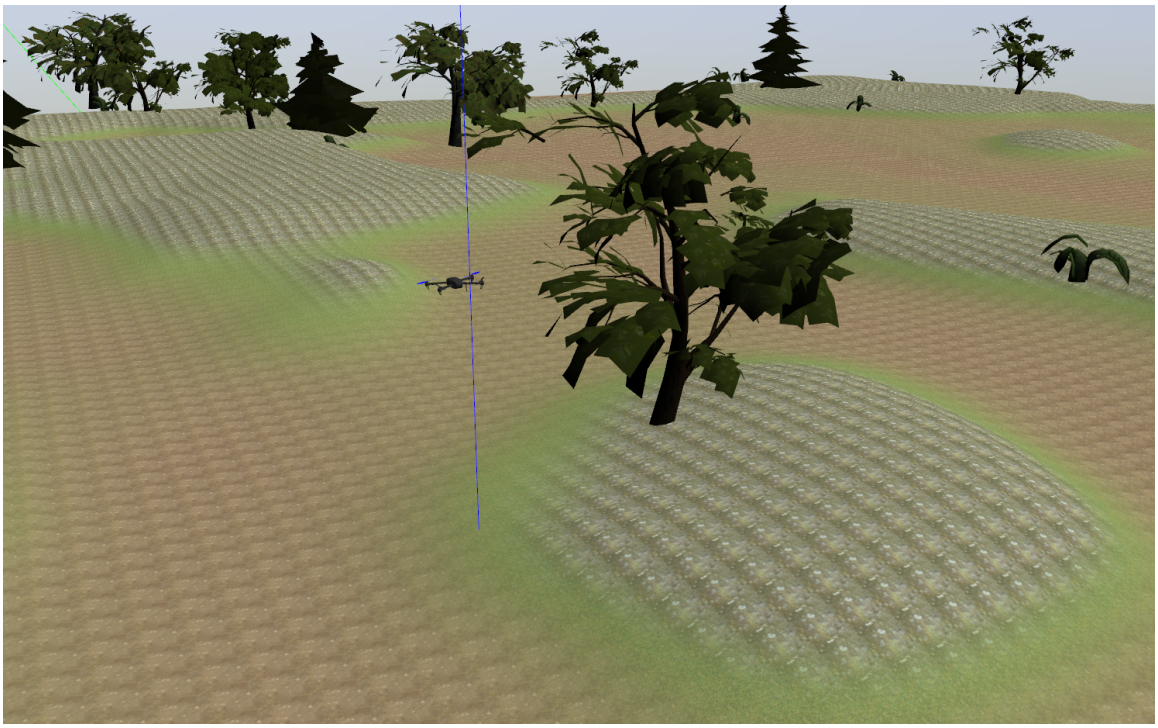


Fig. 5.8. The randomly generated forest environment as seen near ground. (see video)

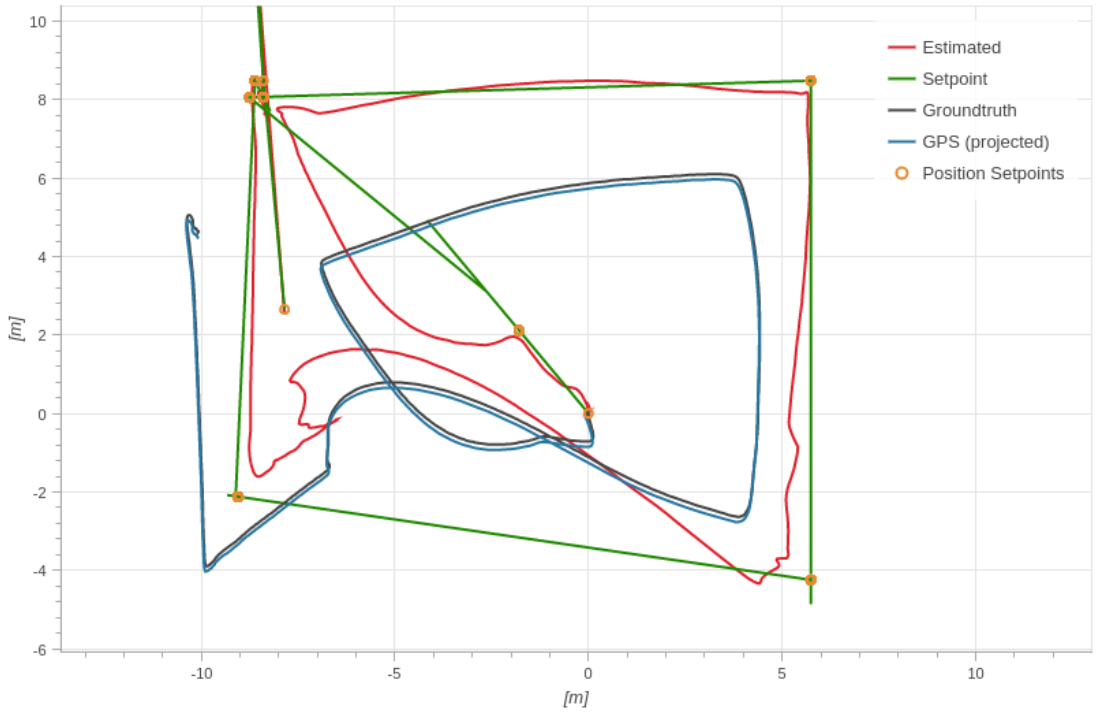


Fig. 5.9. Position estimation performance of LPE in forest world.

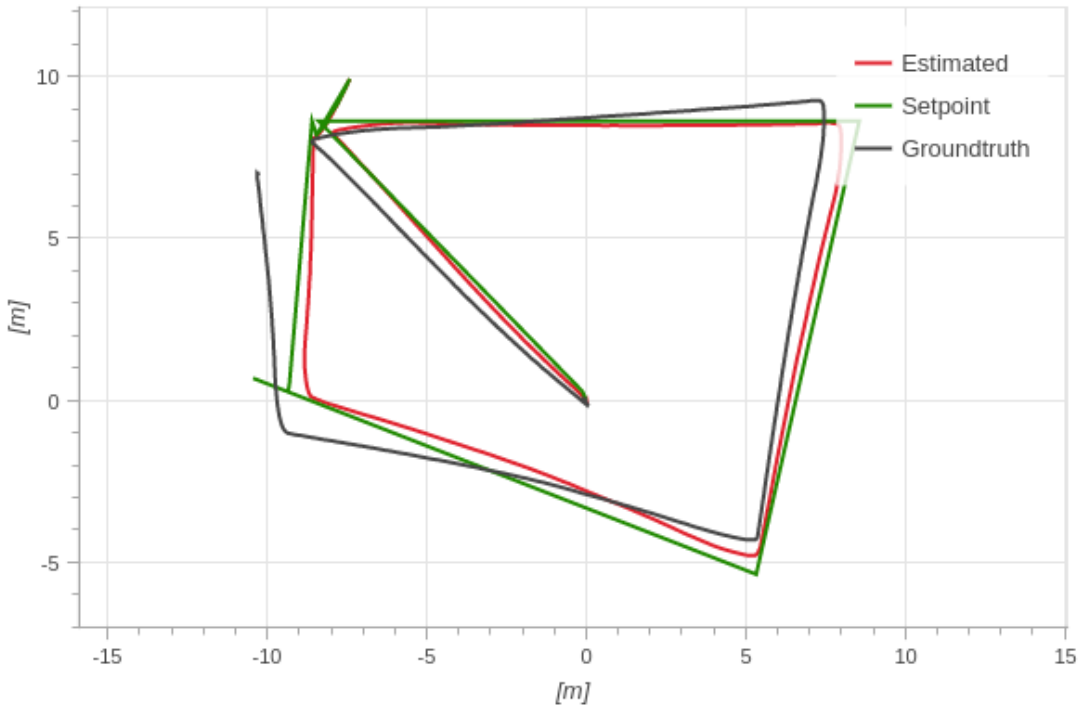


Fig. 5.10. Position estimation performance of IEKF in forest world.

Table 5.1.
Estimator performance summary.

	iekf-forest	iekf_asphalt	lpe-asphalt	lpe-forest
roll_error_mean (deg)	-0.579716	-0.804876	-0.666432	-0.858474
roll_error_std (deg)	0.948943	1.350552	1.557641	1.294216
pitch_error_mean (deg)	-1.574314	-1.063777	-1.023135	-0.415455
pitch_error_std (deg)	1.521558	1.676162	1.441003	2.294992
yaw_error_mean (deg)	4.218197	2.935080	4.132194	4.546224
yaw_error_std (deg)	1.974698	1.053764	3.179310	2.493628
vx_error_mean (m/s)	0.029789	0.033832	0.076343	0.089883
vx_error_std (m/s)	0.108827	0.119753	0.361049	0.861364
vy_error_mean (m/s)	0.035905	-0.000781	-0.080879	-0.113024
vy_error_std (m/s)	0.104390	0.073591	0.447757	0.504294
vz_error_mean (m/s)	-0.014053	-0.052192	-0.068068	-0.082464
vz_error_std (m/s)	0.064945	0.038756	0.076836	0.139088
x_error_mean (m)	0.284757	0.278491	0.215694	2.256762
x_error_std (m)	0.790395	0.414855	0.791383	3.217910
y_error_mean (m)	0.579137	-0.008913	0.297288	0.709045
y_error_std (m)	0.789139	0.189227	0.683715	1.068405
z_error_mean (m)	0.481207	0.072513	0.054236	0.623590
z_error_std (m)	0.088114	0.019158	0.029134	0.288893

A summary table of the performance of the two estimators in both environments in shown in Table 5.1. Note that in general the IEKF estimator outperforms the LPE estimator. Also, the forest environment is more challenging for both algorithms due to the uneven terrain, multiple textures, and varying commanded height above ground, but the IEKF is more robust to these problems and still performs well.

5.4 Continuous Kalman Filter Invariant Set via LMI

We selected the IEKF due to the local asymptotic stability for group affine systems proven in [56]. Unfortunately, this proof requires the system to be group affine. A group affine system's dynamics are log-linear, meaning the system evolution can be represented exactly by a differential equation on the Lie algebra. The Lie exponential map can then be used to transform the Lie algebra back to the Lie group representing the state. The sensor bias in our estimator prevents the system from being group affine. In practice, it has been noted that the bias errors are small and the system possesses good convergence properties [57].

We can approximate the attitude estimation of the UAS as a steady-state continuous Kalman filter on the lie group with bounded disturbances, and a bounded model error representing the effect of the bias. The high sampling rate of the IMU makes the continuous approximation feasible. If the measurement and process noise matrices are constant, the covariance matrix will converge in steady-state, regardless of the measurements.

For the continuous time steady-state Kalman filter, we construct a linear matrix inequality (LMI) which can be used to calculate an invariant set for the error in the Kalman filter when the Gaussian measurement and process noise are replaced with bounded disturbances. This is a practical assumption as truncated normal distributions are more representative of the true system. In addition, we may increase the disturbance levels to represent cyberattacks or environmental conditions such as extreme weather. Bounded modeling errors such as bias, can be accounted for by incorporation into the disturbance magnitude or using polytopic Lyapunov functions to bound errors in the A matrix [50].

We consider the linear observer dynamics with matrices M and N to shape the disturbance. This allows us to simplify the LMI by settings $||w||_\infty < 1$ and $||v||_\infty < 1$ to represent the bounded disturbances while their magnitude may be embedded in M and N.

$$
\begin{aligned}
\dot{x} &= Ax + Mw \\
y &= Cx + Nv \\
\hat{y} &= C\hat{x} \\
\dot{\hat{x}} &= A\hat{x} + L(y - \hat{y}) \\
&= A\hat{x} + LCe + LNv \\
e &:= x - \hat{x} \\
\dot{e} &:= (A - LC)e + Mw - LNv
\end{aligned}
$$

The time derivative of the Lyapunov function can be found using the system dynamics and the product rule.

$$
\begin{aligned}
V &:= e^T P e \\
A_o &:= A - LC \\
\dot{V} &= \dot{e}^T P e + e^T P \dot{e} \\
&= (e^T A_o^T + w^T M^T - v^T N^T L^T) P e + e^T P (A_o e + M w - L N v) \\
&= e^T (A_o^T P + P A_o) e + w^T M^T P e - v^T N^T L^T P e + e^T P M w - e^T P L N v
\end{aligned}
$$

The inequality we wish to represent is:

$$\dot{V} + V - \gamma_v ||v||^2 + \gamma_w ||w||^2 < 0 \tag{5.42}$$

If (5.42) is true, then if $V > \gamma_w ||w||^2 + \gamma_v ||v||^2$, then $\dot{V} < 0$. This is the condition for an attractive invariant set, given by $V \leq \gamma_w ||w||^2 + \gamma_v ||v||^2$.

We consider a simple yet illustrative example:

$$
A = \begin{pmatrix} 0 & 1 \\ -2 & 1 \end{pmatrix} \qquad C = \begin{pmatrix} 0 & 1 \end{pmatrix}
$$

$$
M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad N = \begin{pmatrix} 1 \end{pmatrix} \qquad L = \begin{pmatrix} -0.41421356 \\ 0.91229032 \end{pmatrix}
$$

The LMI can be constructed as follows:

minimize: $\gamma_v + \gamma_w$

subject to the constraints:

$$
\begin{pmatrix} A_o^T P + P A_o + P & PM & -PLN \\ M^T P & -\gamma_w I & 0 \\ -N^T L^T P & 0 & -\gamma_v I \end{pmatrix} < 0 \tag{5.43}
$$

$$P > 0 \tag{5.44}$$

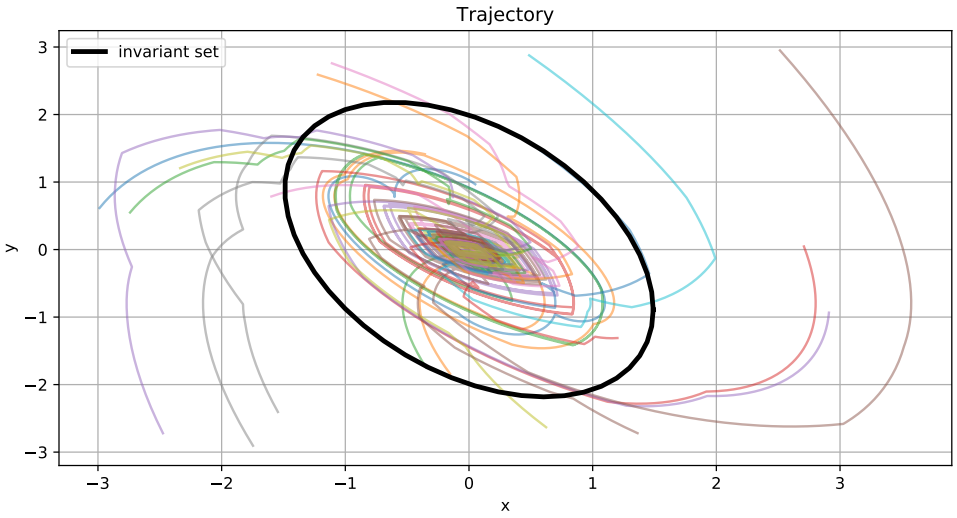


Fig. 5.11. Kalman Filter Monte Carlo Simulation - Error Trajectories

In Figure 5.11 it is clear that all trajectories are attracted to the invariant set in the monte carlo simulation and eventually they remain within the invariant set. Figure 5.12 shows this more clearly by plotting the evolution of the Lyapunov function found by the LMI for each trajectory. It is clear that the Lyapunov function eventually converges below gamma, representing the value of the Lyapunov function at the boundary of the invariant set. Note that since the invariant sets exist in the Lie

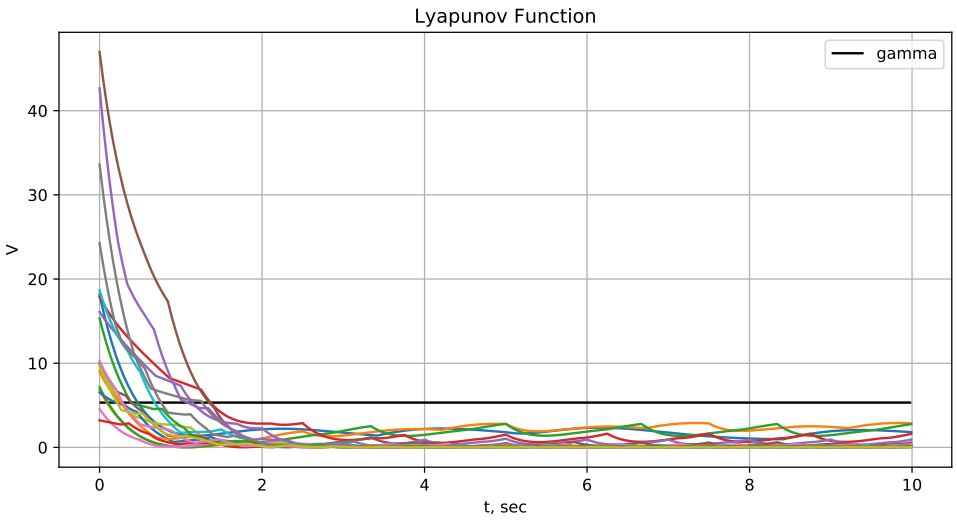


Fig. 5.12. Kalman Filter Monte Carlo Simulation - Error Lyapunov Function

algebra, the exponential map must be used to transform them back to the Lie group before creation of a flow pipe.

5.5 Conclusion

In this chapter we have derived an IEKF based estimation algorithm for a quadrotor with ground-facing optical flow measurements which improves position estimation over the EKF. We compared the results for simulated flights over asphalt and forest environments, demonstrating that the IEKF's performance was superior to the

EKF's. This increased performance allows for outdoor autonomous missions to be flown in the absence of GPS data. Finally, we have constructed a linear matrix inequality (LMI) to compute an invariant set of the IEKF for application of PIHA based model checking as discussed in Chapter 3.

6. COUNTER UNMANNED AERIAL SYSTEM

In the last few years, several high profile events have been interrupted by small scale UASs, commonly referred to as drones. In September of 2013, a man flew a UAS in front of the German chancellor during a campaign event [63]. Two years later, in January 2015, a hobbyist accidentally piloted his DJI Phantom over the highly restricted airspace near the White House and crashed it on the ground [64]. Fortunately, there were no personal injuries or damage in either of those cases. However, as a direct result of those incidents and more, world governments realized that there are a lack of security protocols in place for such small, yet potentially dangerous, threats. In both of the aforementioned cases, the UAS operators had no ill-intent. Yet, as UASs become more innovative, so too might the threats. The fear that UASs will be used for more nefarious purposes, causing severe risk to life and property, is legitimate. This chapter is based on a conference paper [65].

Most of the threat of UASs being used for malicious purposes stems from the fact that these small systems are very difficult to detect. Small UASs are nearly impossible to see with a naked eye from long distances. Anti-aircraft systems that are built specifically for detecting objects in the sky are typically specialized to detect much larger objects or smaller objects moving at much faster speeds. These detection systems, similar to the ones used by the White House, cannot detect small, slow moving UASs [66]. Smaller detection systems have difficulty distinguishing UASs from small birds or other small, slow moving objects in the sky. Furthermore, small UASs are typically flown at much lower altitudes than standard aircraft. Anti-aircraft weaponry pointed at such low altitudes is not only ineffective at countering very small UASs, but also becomes an immediate danger to any people in the area. Similarly, if the UAS has a volatile payload, any destructive counter-measure may cause the UAS and its payload to drop into an undesirable area. An ideal solution to the threat of

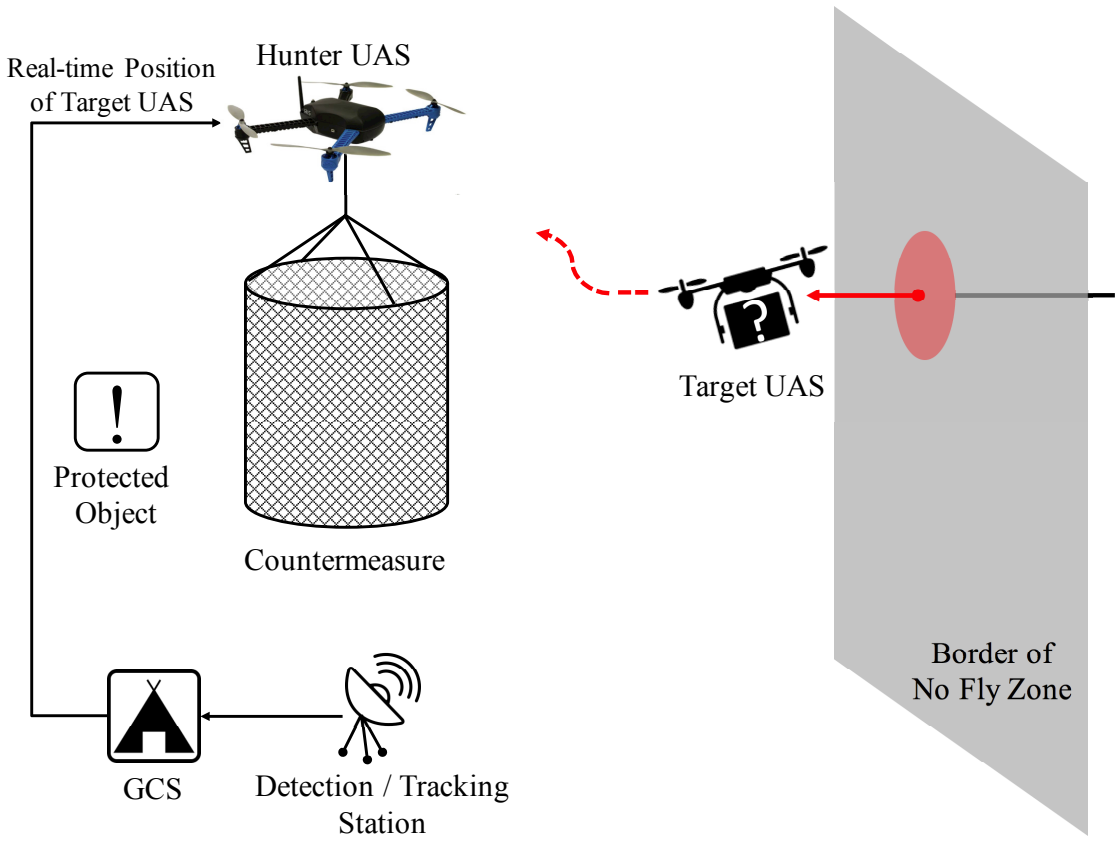


Fig. 6.1. Overview of the CUAS

UASs must be low-regret, meaning that as few people as possible are alarmed or put in harm's way when taking down the threat.

6.1 Controller Design

For estimation of the target UAS state, $\mathbf{x}$, we assume a constant velocity model, as defined below, and employ a hybrid Kalman filter with a continuous prediction step and a discrete update step. RADAR measurements, $\mathbf{z}_k$, are used to correct the estimate and are received at 10 Hz.

$$\mathbf{x} = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z \end{bmatrix}^T \tag{6.1}$$

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} v_x & v_y & v_z & 0 & 0 & 0 \end{bmatrix}^T \tag{6.2}$$

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t)) + \mathbf{w}(t) \tag{6.3}$$

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T \tag{6.4}$$

$$\mathbf{z}_k = \mathbf{g}(\mathbf{x}_k) + \mathbf{v}_k \tag{6.5}$$

where p_i are the position components in the north-east-down frame, v_i are the velocity components in the north-east-down frame, the state at step k is defined as $\mathbf{x}_k = \mathbf{x}(t_k)$, the RADAR measurement at step k is defined as $\mathbf{z}_k = \mathbf{z}(t_k)$, the process noise at time t, $\mathbf{w}(t)$, is normally distributed with mean $\mathbf{0}$ and covariance matrix Q, $\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, Q)$, and the RADAR measurement noise at step k, v_k, is normally distributed with mean $\mathbf{0}$ and covariance matrix R, $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, R)$.

In order to increase the rate of successful engagement, we command the hunter position set-point $\mathbf{x}_{sp}$ to lead the trajectory of the target by $\Delta_l = 0.6$ seconds. This accounts for both the net trailing the hunter UAS, due to drag at high velocities, as well as latencies in the system. This prediction is achieved using the constant velocity model defined below.

$$\mathbf{x}_{sp} = \hat{\mathbf{x}}(t) + \mathbf{f}(\hat{\mathbf{x}}(t))\Delta_l \tag{6.6}$$

A PID controller is then used to regulate the error ($\mathbf{x_e} = \mathbf{x}_{sp} - \hat{\mathbf{x}}$) in the system and send a velocity command to the lower level control laws. This velocity command must be saturated using the norm of the velocity, instead of component-wise for the x and y components, to ensure that the heading to the target is preserved. Since the target vehicle is approaching from a distance, the velocity command will be saturated during a significant portion of the flight. It is therefore important that the saturated velocity vector point toward the target. This simple approach proved effective. If the hunter overshoots the target on the first pass, it turns around and then continues

trying to intercept while following where the relative speed is reduced and the tracking is more accurate.

If the commanded velocity in the x-y plane exceeds the maximum, V_{xy}, it is saturated by:

$$sat_{xy}(v_x) = V_{xy}\left(v_x/\sqrt{v_x^2 + v_y^2}\right) \tag{6.7}$$

$$sat_{xy}(v_y) = V_{xy}\left(v_y/\sqrt{v_x^2 + v_y^2}\right) \tag{6.8}$$

Similarly, if the commanded z velocity exceeds the maximum, V_z, it is saturated by:

$$sat_z(v_z) = V_z\left(v_z/\sqrt{v_z^2}\right) \tag{6.9}$$

The original PX4 firmware did not employ saturation for the local position setpoint command sent via MAVLink [67], so a modification to the firmware was required. The hunter UAS reached speeds of 20 m/s without velocity saturation, which reached the target quickly, but lead to excessive overshoot.

6.2 Experiments and Results

Field testing was conducted over a five day period in June 2015, due to limited availability of the RADAR equipment, to evaluate the performance of the prototype CUAS. The experiments revealed issues with the control of the system, as discussed previously, but finally concluded with a successful engagement of the target UAS by the hunter. As a result of our limited test time, we were only able to conduct five test flights with the final software configuration, and we were successful on the 5^{th} attempt. A photo of the engagement is shown in Fig. 6.2. After the successful engagement with the RADAR-based system, we repeated the experiment with LIDAR in place of RADAR. LIDAR was less successful since the UAS had to be close before it was visible.

Fig. 6.2. Engagement of target by hunter

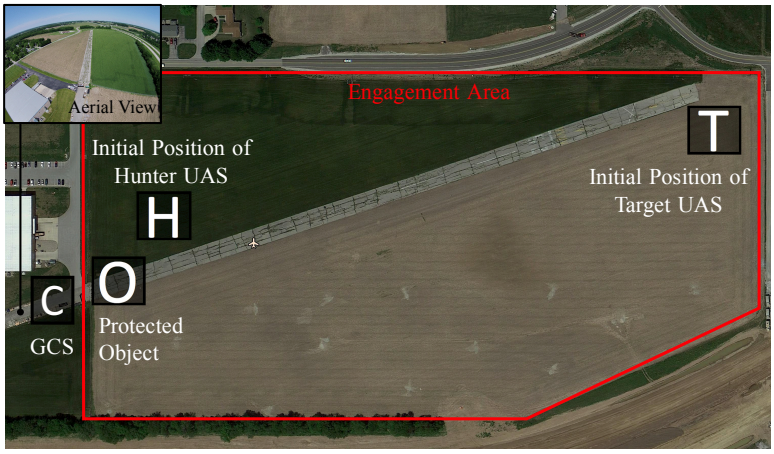


Fig. 6.3. Aerial view of the test site

6.2.1 Test Setup

The tests were performed at an abandoned airport with a 500 meter asphalt runway. The aerial view of the airport is shown in Fig. 6.3. To make the test as realistic as possible, we used a DJI Phantom 2 as the target UAS, due to its popularity in the market. The target UAS flew in a straight line from the end of the runway toward the protected area at 4 m/s.

6.2.2 Results

The overall results are plotted in Fig. 6.4. Notice that when RADAR data is not present, the prediction of the target UAS continues in a straight line. The hunter UAS approaches the target and makes the first pass around 100 meters from the protected location. The hunter then continues engagement attempts over the next

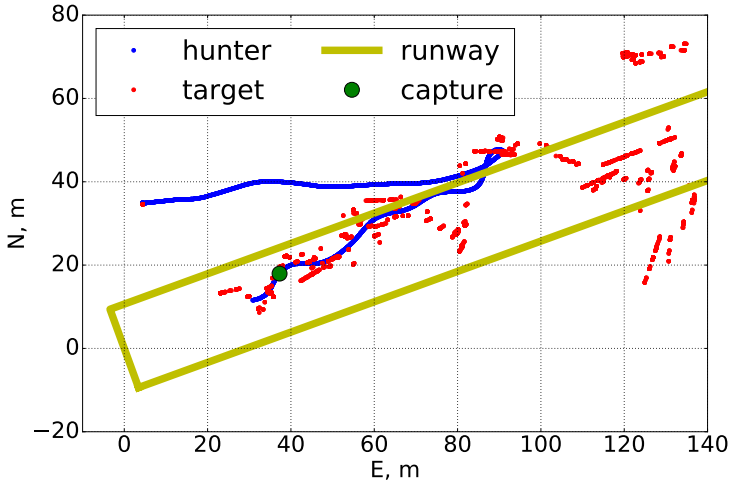


Fig. 6.4. Interception of target by hunter with RADAR based tracking

13 seconds and 52 meters. The final engagement occurs at approximately 40 meters from the protected location.

The time history of the target and hunter trajectories can be seen in Fig. 6.5. The initial pass of the target by the hunter occurs at 15 seconds. As the measurements become more accurate, the hunter continues making engagement attempts with the net until it captures the target UAS at 28 seconds.

The altitude time history is shown in Fig. 6.6. The RADAR elevation angle data is degraded when compared to the azimuth, so we compensated by having the hunter vehicle fly at a set height of 17 meters and increasing the length of the net to account for the uncertainty in the target altitude. The hunter altitude tracking performance error remains below 1 meter from the estimated altitude during the entire flight.

The engagement is visible on the plot of the measured IMU data at 28 seconds. There is a 1 g spike in the z acceleration as the vehicle stops after the engagement, but in this case, the target UAS was a DJI Phantom 2 and the props stopped after hitting the carbon fiber supports on the net, which caused it to fall to the ground

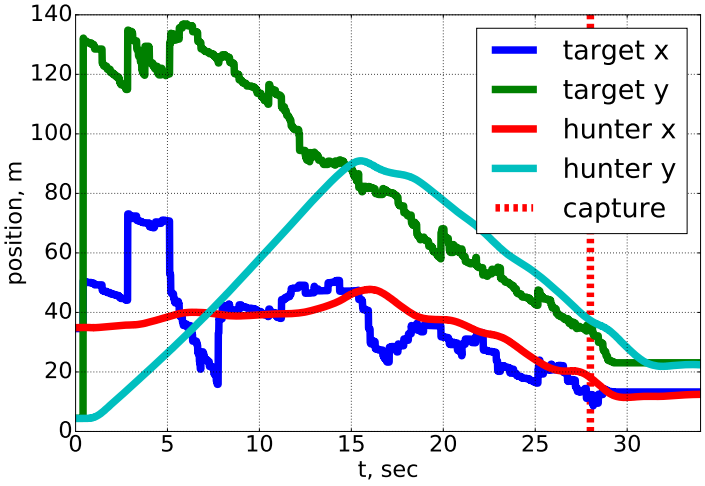


Fig. 6.5. Position tracking

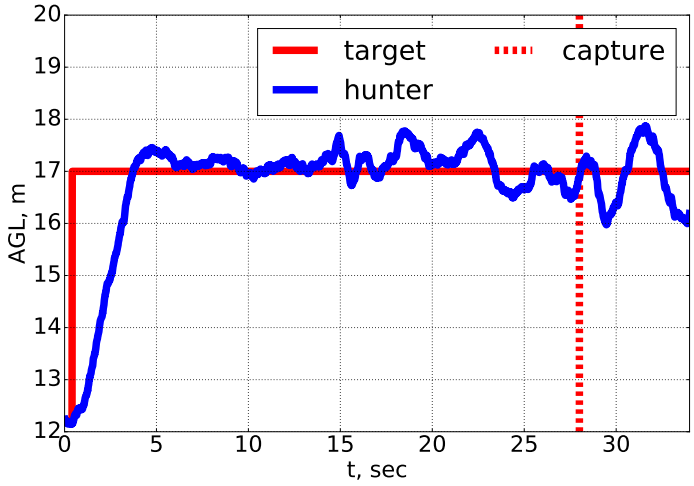


Fig. 6.6. Altitude tracking

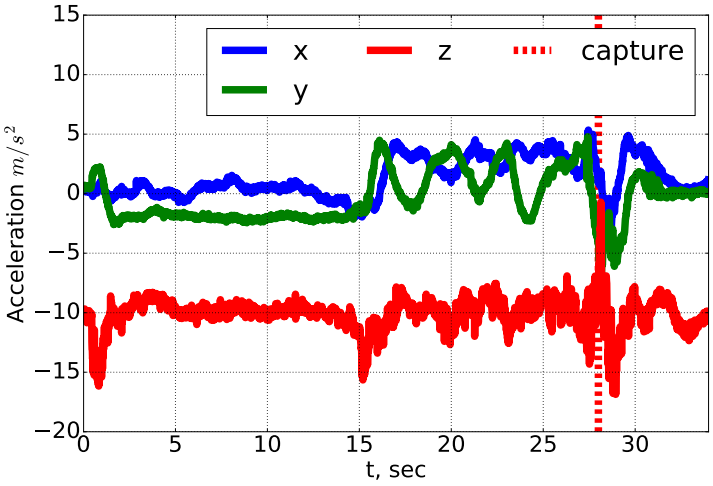


Fig. 6.7. Measured IMU accelerations of the hunter

instead of becoming tangled. This caused little disturbance to the hunter as shown in Fig. 6.7.

Fig. 6.8 shows the two phases of flight where the hunter approaches the target UAS and reaches the commanded velocity of $7m/s$. After missing the first high velocity pass, the hunter turns around to pursue the target with the net and engages it at 28 seconds. The ability of this system to continue engagement attempts makes it more robust than single-fire systems, such as net guns, where you cannot retry after the first attempt.

6.3 Conclusion

Our experiment has shown that RADAR-based, fully autonomous, air-to-air Counter Unmanned Aerial Systems (CUASs) are possible with existing technologies. The major challenges were distinguishing between the target and hunter UASs when they were near each other and the altitude accuracy of the RADAR measurements. The cylindrical net design that enabled a single hunter UAS to make repeated engagement

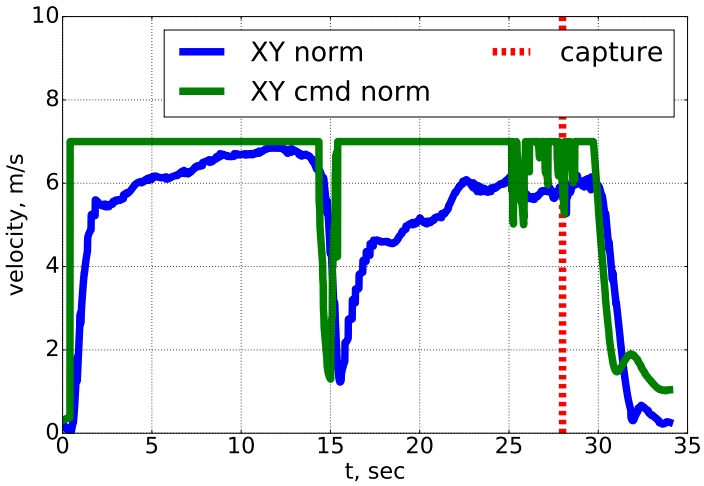


Fig. 6.8. Velocity magnitude (xy components) of the hunter

attempts at various angles and with full autonomy greatly improved the reliability of the system. To improve the reliability of the system, the size of both the net and the hunter UAS can be increased. Also, the commanded approach trajectory could employ game theory, instead of saturated PID control.

7. SUMMARY

In this dissertation, we have discussed the vulnerabilities inherent in current unmanned vehicles to both malicious attacks and system logic errors. We have then stepped through a series of progressively complex case studies and developed algorithms for verification of each vehicle. We have focused on creation of flow pipes for each case which are the fundamental building block of Polyhedral Invariant Hybrid Automatons (PIHAs). We have demonstrated application of PIHA based model checking by hand in Chatper 3. This work can be used for reachable set analysis, or model checking when combined with a finite state machine representation of the control system. Finally to address the threat of a UAS under hostile control, we have developed and tested a control system for a hunter UAS coupled with ground based RADAR.

7.1 Acknowledgments

We would like to thank Sypris Electronics for supporting the work in Chapter 2. Chapter 3 was supported by the National Science Foundation under Grant Numbers CNS-1239196, CNS-1239171, and CNS-1239229. Additional support was provided by the AFRL SFFP program.

REFERENCES

REFERENCES

[1] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta, "Ensuring Safety, Security, and Sustainability of Mission-Critical Cyber–Physical Systems," *Proceedings of the IEEE*, vol. PP, no. 99, pp. 1–17, 2011.

[2] D. K. Nilsson and U. E. Larson, "A Defense-in-Depth Approach to Securing the Wireless Vehicle Infrastructure," *Journal of Networks*, vol. 4, no. 7, pp. 552–564, sep 2009. [Online]. Available: http://www.academypublisher.com/jnw/vol04/no07/jnw0407552564.pdf

[3] DefenseWeb, "Flying operations of remotely piloted aircraft unaffected by malware - USAF — defenceWeb," 2011. [Online]. Available: http://www.defenceweb.co.za/index.php?option=com{_}content{&}view=article{&}id=20084

[4] J. Wolf, "China key suspect in U.S. satellite hacks: commission — Reuters," 2011. [Online]. Available: https://www.reuters.com/article/us-china-usa-satellite-idUSTRE79R4O320111028

[5] J. S. Warner and R. G. Johnston, "GPS spoofing countermeasures," Los Alamos National Laboratory, Tech. Rep., 2003. [Online]. Available: http://library.lanl.gov/cgi-bin/getfile?00852243.pdf

[6] J. Krozel and D. Andrisani II, "Independent ADS-B verification and validation," in *AIAA 5th ATIO*, sep 2005. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/6.2005-7351

[7] J. Goppert, N. Wenyao, and I. Hwang, "ArduPilotOne: Extending the capabilities of the open-source ArduPilotMega autopilot system," in *AIAA Infotech at Aerospace Conference and Exhibit 2012*, 2012.

[8] G. Tétrault-Farber, "Russian postal drone program hits wall in debut," 2018. [Online]. Available: https://www.reuters.com/article/us-russia-post-drone-crash/russian-postal-drone-program-hits-wall-in-debut-idUSKCN1H91B4

[9] S. Gorman, Y. J. Dreazen, and A. Cole, "Insurgents Hack U.S. Drones - WSJ," 2009. [Online]. Available: https://www.wsj.com/articles/SB126102247889095011

[10] R. Mitchell and I.-R. Chen, "Survivability analysis of mobile cyber physical systems with voting-based intrusion detection," in *Proc. 7th Int. Wireless Communications and Mobile Computing Conf. (IWCMC)*, 2011, pp. 2256–2261.

[11] Q. Zhu, C. Rieger, and T. Basar, "A hierarchical security architecture for cyber-physical systems," in *Proc. 4th Int Resilient Control Systems (ISRCS) Symp*, 2011, pp. 15–20.

[12] A. Chutinan and B. H. Krogh, "Computational techniques for hybrid system verification," *Automatic Control, IEEE Transactions on*, vol. 48, no. 1, pp. 64–75, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=1166525

[13] ——, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations," in *Hybrid Systems: Computation and Control*. Springer, 1999, pp. 76–90. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-48983-5{_}10

[14] B. I. Silva, K. Richeson, B. Krogh, and A. Chutinan, "Modeling and verifying hybrid dynamic systems using CheckMate," in *Proceedings of 4th International Conference on Automation of Mixed Processes*, 2000, pp. 323–328.

[15] E. Asarin, T. Dang, and O. Maler, "The d/dt tool for verification of hybrid systems," in *Computer Aided Verification*. Springer, 2002, pp. 365–370. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-45657-0{_}30

[16] B. I. Silva, O. Stursberg, B. H. Krogh, and S. Engell, "An assessment of the current status of algorithmic approaches to the verification of hybrid systems," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 3. IEEE, 2001, pp. 2867–2874. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=980711

[17] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems," in *Computer aided verification*. Springer, 1997, pp. 460–463. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-63166-6{_}48

[18] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical computer science*, vol. 138, no. 1, pp. 3–34, 1995.

[19] J. Goppert, A. Shull, N. Sathyamoorthy, W. Liu, I. Hwang, and H. Aldridge, "Software/hardware-in-the-loop analysis of cyberattacks on unmanned aerial systems," *Journal of Aerospace Information Systems*, vol. 11, no. 5, 2014.

[20] G. Looye, A. Varga, S. Bennani, D. Moormann, and G. Grubel, "Robustness Analysis Applied to Autopilot Design Part 1: mu-Analysis of Design Entries to a Robust Flight Control Benchmark," 1998.

[21] O. Kosut, L. Jia, R. J. Thomas, and L. Tong, "Malicious data attacks on smart grid state estimation: Attack strategies and countermeasures," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE, 2010, pp. 220–225.

[22] Y. Liu, P. Ning, and M. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security*, vol. 14, no. 1, pp. 13:1—-13:33, may 2011. [Online]. Available: http://discovery.csc.ncsu.edu/pubs/ccs09-PowerGrids.pdf

[23] W. Liu, C. Kwon, I. Aljanabi, and I. Hwang, "Cyber Security Analysis for State Estimators in Air Traffic Control Systems," in *Guidance, Navigation, and Control and Co-located Conferences*. American Institute of Aeronautics and Astronautics, aug 2012. [Online]. Available: http://dx.doi.org/10.2514/6.2012-4929

[24] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*. IEEE, 2009, pp. 911–918.

[25] ——, "False data injection attacks in control systems," in *First Workshop on Secure Control Systems*, 2010.

[26] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli, "False data injection attacks against state estimation in wireless sensor networks," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5967–5972.

[27] A. Teixeira, S. Amin, H. Sandberg, K. H. Johansson, and S. Sastry, "Cyber security analysis of state estimators in electric power systems," in *49th IEEE Conference on Decision and Control*, 2010, pp. 5991–5998.

[28] H. Li, L. Lai, and W. Zhang, "Communication Requirement for Reliable and Secure State Estimation and Control in Smart Grid," *Smart Grid, IEEE Transactions on*, vol. 2, no. 3, pp. 476–486, 2011.

[29] T. Baumeister, "Literature review on smart grid cyber security," University of Hawaii, Tech. Rep., 2010. [Online]. Available: https://csdl-techreports.googlecode.com/svn/trunk/techreports/2010/10-11/10-11.pdf

[30] S. Amin, A. A. Cárdenas, and S. S. Sastry, "Safe and Secure Networked Control Systems under Denial-of-Service Attacks," in *Proceedings of the 12th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 31–45. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00602-9{_}3

[31] S. Amin, "Security Interdependencies for Networked Control Systems with Identical Agents," nov 2010. [Online]. Available: http://www.truststc.org/pubs/756.html

[32] D. Kundur, X. Feng, S. Liu, T. Zourntos, and K. L. Butler-Purry, "Towards a Framework for Cyber Attack Impact Analysis of the Electric Smart Grid," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, pp. 244–249.

[33] R. D. Colgren and T. L. Johnson, "Flight mishap prevention for UAVs," in *Proc. Aerospace Conf. IEEE*, vol. 2, 2001.

[34] J. M. Wilson and M. E. Peters, "Automatic flight envelope protection for light general aviation aircraft," in *Proc. IEEE/AIAA 28th Digital Avionics Systems Conf. DASC '09*, 2009.

[35] H. Shin, Y. Kim, E. T. Kim, and K. J. Seong, "Flight envelope protection controller using dynamic trim algorithm," in *Proc. ICCAS-SICE*, 2009, pp. 3228–3232.

[36] J. E. Williams and S. R. Vukelich, *The USAF Stability and Control Digital DATCOM Users Manual*, USAF, nov 1979.

[37] W. B. Blake, "Prediction of Fighter Aircraft Dynamic Derivatives Using Digital Datcom," in *AIAA 3rd Applied Aerodynamics Conference*, 1985.

[38] B. Stevens and F. Lewis, *Aircraft control and simulation.* New York, NY: Wiley, 2003. [Online]. Available: http://books.google.com/books/about/Aircraft{_}control{_}and{_}simulation.html?id=T0Ux6av4btIC

[39] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology*, 2, Ed. The Institute of Electrical Engineers, 2004.

[40] S. Julier and J. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, mar 2004. [Online]. Available: http://ieeexplore.ieee.org/document/1271397/

[41] D. Du, L. Liu, and X. Du, "A low-cost attitude estimation system for UAV application," in *Proc. Chinese Control and Decision Conf. (CCDC)*, 2010, pp. 4489–4492.

[42] A. Willsky and H. Jones, "A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems," *Automatic Control, IEEE Transactions on*, vol. 21, no. 1, pp. 108–112, feb 1976.

[43] S. X. Ding, *Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms and Tools.* Springer, 2008. [Online]. Available: http://www.amazon.com/Model-based-Fault-Diagnosis-Techniques-ebook/dp/B001BTLVQ6

[44] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using the relative velocity paradigm," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, may 1993, pp. 560 –565 vol.1.

[45] H. V. N. Botha, "A Closed Loop Research Platform That Enables Dynamic Control Of Wing Gait Patterns In A Vertically Constrained Flapping Wing - Micro Air Vehicle," 2016. [Online]. Available: https://etd.ohiolink.edu/pg{_}10?0::NO:10:P10{_}ACCESSION{_}NUM:wright1462801627

[46] J. Goppert, J. Gallagher, I. Hwang, and E. Matson, "Model Checking of a Flapping-Wing Mirco-Air-Vehicle Trajectory Tracking Controller Subject to Disturbances," in *Advances in Intelligent Systems and Computing*, vol. 274. Springer, Cham, 2014, pp. 531–543. [Online]. Available: http://link.springer.com/10.1007/978-3-319-05582-4{_}46

[47] D. B. Doman, M. W. Oppenheimer, and D. O. Sigthorsson, "Wingbeat shape modulation for flapping-wing micro-air-vehicle control during hover," *Journal of guidance, control, and dynamics*, vol. 33, no. 3, pp. 724–739, 2010. [Online]. Available: http://arc.aiaa.org/doi/pdf/10.2514/1.47146

[48] C. Baier, J.-P. Katoen, and Others, *Principles of model checking.* MIT press Cambridge, 2008, vol. 26202649.

[49] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "Nusmv 2: An opensource tool for symbolic model checking," in *Computer Aided Verification.* Springer, 2002, pp. 359–364. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-45657-0{_}29

[50] M. Corless, "AAE 666 Notes," Purdue University, Tech. Rep., 2013.

[51] H. Xiang and L. Tian, "Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (UAV)," *Biosystems Engineering*, vol. 108, no. 2, pp. 174–190, feb 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1537511010002436

[52] C. A. F. Ezequiel, M. Cua, N. C. Libatique, G. L. Tangonan, R. Alampay, R. T. Labuguen, C. M. Favila, J. L. E. Honrado, V. Canos, C. Devaney, A. B. Loreto, J. Bacusmo, and B. Palma, "UAV aerial imaging applications for post-disaster assessment, environmental management and infrastructure development," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, may 2014, pp. 274–283. [Online]. Available: http://ieeexplore.ieee.org/document/6842266/

[53] P. Doherty and P. Rudol, "A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization," in *AI 2007: Advances in Artificial Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–13. [Online]. Available: http://link.springer.com/10.1007/978-3-540-76928-6{_}1

[54] J. Goppert, S. Yantek, and I. Hwang, "Invariant Kalman filter application to optical flow based visual odometry for UAVs," in *International Conference on Ubiquitous and Future Networks, ICUFN*, 2017.

[55] S. Bonnabel, "Left-invariant extended {Kalman} filter and attitude estimation," in *Decision and {Control}, 2007 46th {IEEE} {Conference} on*. IEEE, 2007, pp. 1027–1032. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/4434662/

[56] A. Barrau and S. Bonnabel, "The invariant extended Kalman filter as a stable observer," *IEEE Transactions on Automatic Control*, 2017.

[57] ——, "Invariant Kalman Filtering," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 237–257, 2017.

[58] S. Bonnabel, P. Martin, and E. Salaun, "Invariant Extended Kalman Filter: theory and application to a velocity-aided attitude estimation problem," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, dec 2009, pp. 1297–1304. [Online]. Available: http://ieeexplore.ieee.org/document/5400372/

[59] M. Dille, B. Grocholsky, and S. Singh, "Outdoor Downward-Facing Optical Flow Odometry with Commodity Sensors," in *Field and Service Robotics*. Springer, Berlin, Heidelberg, 2010, pp. 183–193. [Online]. Available: http://link.springer.com/10.1007/978-3-642-13408-1{_}17

[60] J. Campbell, R. Sukthankar, and I. Nourbakhsh, "Techniques for evaluating optical flow for visual odometry in extreme terrain," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 4. IEEE, 2004, pp. 3704–3711. [Online]. Available: http://ieeexplore.ieee.org/document/1389991/

[61] D. Honegger, P. Greisen, L. Meier, and P. Tanskanen, "Real-time velocity estimation based on optical flow and disparity matching." in *IROS*, 2012. [Online]. Available: https://www.researchgate.net/profile/Marc{_}Pollefeys/publication/261353635{_}Real-time{_}velocity{_}estimation{_}based{_}on{_}

optical{_}flow{_}and{_}disparity{_}matching/links/5458a6f40cf2cf5164828b0d/Real-time-velocity-estimation-based-on-optical-flow-and-disparity-matching.

[62] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, may 2013, pp. 1736–1741. [Online]. Available: http://ieeexplore.ieee.org/document/6630805/

[63] S. Gallagher, "German chancellor's drone attack shows the threat of weaponized UAVs," 2013. [Online]. Available: http://arstechnica.com/information-technology/2013/09/german-chancellors-drone-attack-shows-the-threat-of{%}5C{%}5C-weaponized-uavs/

[64] H. Abdullah, "Man Detained for Flying Drone Near White House," 2015. [Online]. Available: http://www.nbcnews.com/news/us-news/man-detained-trying-fly-drone-near-white-house-n359011

[65] J. M. Goppert, A. R. Wagoner, D. K. Schrader, S. Ghose, Y. Kim, S. Park, M. Gomez, E. T. Matson, and M. J. Hopmeier, "Realization of an autonomous, air-to-air Counter Unmanned Aerial System (CUAS)," in *Proceedings - 2017 1st IEEE International Conference on Robotic Computing, IRC 2017*. IEEE, apr 2017, pp. 235–240. [Online]. Available: http://ieeexplore.ieee.org/document/7926544/

[66] M. S. Schmidt and M. D. Shear, "A Drone, Too Small for Radar to Detect, Rattles the White House," 2015. [Online]. Available: http://www.nytimes.com/2015/01/27/us/white-house-drone.html

[67] L. Meier, J. Camacho, B. Godbolt, J. Goppert, L. Heng, M. Lizarraga, Others, and MAVLink, "Mavlink: Micro air vehicle communication protocol," jan 2013. [Online]. Available: http://qgroundcontrol.org/mavlink/start

VITA

VITA

James Goppert received his Bachelor and Master's Degree in Aerospace Engineering from Purdue University. He has contributed to many open source aerospace software projects including the PX4 autopilot, the APM autopilot, and the JSBSim flight simulator. He is also CEO of a robotics consulting company specializing in autopilot software and computer vision.