ABSTRACT


Title of Thesis: **RESILIENT STATE ESTIMATION FOR MICRO AIR VEHICLES UNDER SENSOR ATTACKS**

*Akshay Prasad, Master of Science, 2017*

Thesis Directed By: Dr. Nikhil Chopra (Department of Mechanical Engineering)

This thesis proposes a solution to the problem of resilient state estimation and sensor fusion in an autonomous micro air vehicle. The setup comprises of redundant sensors that measure the same physical signal. An adversary may spoof a subset of these sensors and send falsified readings to the controller, potentially compromising performance and safety of the system. This work integrates Brooks-Iyengar Sensor fusion algorithm with a generic state estimator as a method to thwart sensor attacks. The algorithm outputs a point estimate and a fusion interval based on an assumed set of faulty sensors. Finally, the thesis illustrates the usefulness of the resilient state estimator with a case study on a MAV flight dataset.

# RESILIENT STATE ESTIMATION FOR MICRO AIR VEHICLES UNDER SENSOR ATTACKS

by

Akshay Prasad

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2017

Advisory Committee:
Professor Nikhil Chopra, Chair
Professor Jeffrey Herrmann
Professor Shapour Azarm

To my parents, family and friends.

# Acknowledgements

Firstly, I would like to thank Dr. Nikhil Chopra for making this thesis possible and for his continuous support throughout my tenure at the University of Maryland, College Park. He offered me the chance to pursue my thesis under his supervision and encouraged me towards my thesis topic. I am highly grateful to him for his time and effort throughout the last two years. Dr. Chopra ensured that I felt comfortable discussing any problems I encountered. He set a lot of directions for me to delve deep into the field of micro air vehicles (MAVs), something I had very little prior experience. He ensured I had access to all possible resources for the successful completion of my thesis.

Further, I would like to thank Dr. John MacCarthy for his continuous support regarding all matters including academic, administrative, and personal. He always ensured that I was on track with my courses and thesis. He was also instrumental in helping me sort out any issues that I faced during the span of my studentship at the University of Maryland, College Park. I would also like to thank Dr. Jeffrey Herrmann and Dr. Shapour Azarm for serving on my thesis committee and for reviewing my work. I deeply appreciate their time and academic expertise.

I would additionally like to thank Nirupam Gupta and Gurtajbir Herr for their help and support. Especially in matters related to getting my workspace set up and helping me place orders for my hardware equipment.

Likewise, I would like to thank my peers at the University of Maryland for a wonderful experience. Finally, I want to express deep gratitude to my family for their relentless financial and emotional support. They played a key role in keeping me motivated,

which has brought me to the point where I conclude my thesis. This work was partially supported by Naval Air Warfare Center Aircraft Division - Pax River, MD under Contract No. N00421132M022.

# Table of Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **ADC** | Analog to Digital Converter |
| **CAN** | Controller Area Network |
| **CPS** | Cyber Physical System |
| **DOF** | Degree of Freedom |
| **DOS** | Denial of Service |
| **EKF** | Extended Kalman Filter |
| **ESC** | Electronic Speed Controller |
| **GCS** | Ground Control Station |
| **GPS** | Global Positioning System |
| **I2C** | Inter - IC Bus |
| **IMU** | Inertial Measurement Unit |
| **IOT** | Internet of Things |
| **LIDAR** | Light Detection and Ranging |
| **LTE** | Long Term Evolution 4G Wireless Communication Standard |
| **MAV** | Micro Air Vehicles |
| **MEMS** | Microelectromechanical Systems |
| **PPM** | Pulse Position Modulation |
| **RC** | Radio Control |
| **SONAR** | Sound Navigation and Ranging |
| **SPI** | Serial Peripheral Interface |
| **UART** | Universal Asynchronous Receiver/Transmitter |
| **UAV** | Unmanned Air Vehicles |
| **UGV** | Unmanned Ground Vehicles |

# 1. Introduction

Cybersecurity incidents have been on the rise over the past years. No industry, gadget, or tool is safe from threat and menace of cyber-attacks. The next generation of cyber physical systems (CPS) is becoming complex in design and encompass a diverse set of components and elements. Security concerns and considerations can no longer be an after-thought. The design process needs to incorporate these vectors and ensure necessary protection.

Modern vehicles have numerous embedded elements communicating internally as well as externally using different technologies. Moreover, with the recent interest in Internet of Things (IOT), vehicle system designs have moved from insulated control systems to open and connected architectures with functionalities such as remote diagnostic information, inter-device communication, and online updates. An ever-increasing set of functionalities, network connectivity, and design complexity introduces security susceptibilities that are exploitable. Often the security guarantees of these systems are based on the security of external communication links and authentication protocols. Consequently, an effective attack that compromises the gateway, or physical attacks on components connected on the internal network may be leveraged to completely handicap the system. A similar strategy was used in [1] by the authors to disrupt the operation of a car and take complete control over it.

An attack on a CPS may be carried out by intruding the computational nodes or communication channels and altering the physical environment. Information security approaches can be used to safeguard the CPS, but noninvasive attacks such as altering the

physical environment may still be possible [2], [3] and [4]. In [2], [3] and [4] the authors review how an attack signal may be introduced into the control system loop by altering the sensor measurements. Orthodox cybersecurity techniques such as secure communication protocols for internal networks cannot defend the system against attacks on physical components. Acquiring access to the internal network would allow the attacker to completely compromise the controller, actuator, and all elements on board. Such attacks may be restricted by using cryptography tools. However, these tools may add additional processing delay and require additional resources, which might be scarce in some CPS domains. Researchers in [5] identified that it is necessary to address this security challenge in the control design phase because attacks may be disguised as malicious signals to the controller. Vaguely attacks may be classified into:

- The attacker takes over a sensor to provide wrong readings.
- Disturb actuation.

The thesis primarily discusses the scenario when an adversary attacks the sensor measurements of a micro air vehicle (MAV) (a subclass of UAVs, see Table 1). Unmanned Air Vehicles (UAVs) are vehicles that are either controlled remotely by a Ground Control Station, radio remote controller, or autonomously programmed prior to the mission flight. UAVs are used for military missions as well as for commercial usage. MAVs hold the promise of enabling online retail, survey, emergency services, etc. They have been gaining recognition over the previous few years. With companies such as Amazon Prime [6], UPS, Google [7], DJI, GoPro investing heavily in these air vehicles, MAVs are gradually becoming a common place. This is also supplemented by an increasing affordability of

such MAVs making it more accessible to hobbyists and enthusiasts. One can construct a MAV in as less as $250 by procuring parts from multiple online vendors.

Numerous companies working on different applications for such MAVs including and not limited to cinematography, rescue missions, agricultural chemical deployment, ecological surveys, emergency response, 3-D Mapping, drone delivery, etc. is increasing incessantly. However, their exposure to cyber-attacks makes them a potential tool for espionage, terrorism, vandalism, etc.

Recently, commercial UAVs were authorized to fly in the US national air space by the FAA [8]. It is anticipated that more than 7 million small drones will occupy the US airspace by 2020 [9]. Consequently, the enticement for cyber-attacks on UAVs is only going to surge.

With an increasing level of autonomy, the mandate for secure and hardened systems is going to surge exponentially. Additional studies and analysis, studying the susceptibility of MAVs to cyber-attacks and investment in security of MAVs at each layer of abstraction is quintessential. The infamous attack on a UAV system [10] highlighted this need; when members of a terrorist group intercepted and recorded a UAV video feed using a $26 software SkyGrabber. SkyGrabber was designed to capture free satellite-entertainment channels [10]. Investigations revealed that this was enabled due to an unencrypted video channel. It was revealed later [11], that the flaw was known to the US government since the 1990s. In 2012, Iranian forces stated that they captured an RQ-170 [12], [13]. Subsequently they landed the UAV and obtained mission data. It was postulated that a lack of security measures of the UAV sensor system was used to attack the GPS subsystem [12]. In 2012, a research team at UT Austin [14], in a demonstration, showcased their ability to

hijack a military UAV by spoofing the GPS signal and thereby taking complete control of the UAV. They showcased a proof-of-concept by using $1,000 worth of equipment. These above cases demonstrate a necessity for investment in research for cybersecurity for UAVs of all classes for military and public use.

Moreover, the GPS-spoofing notion emphasizes the need to include unusual elements (e.g. sensors, input channels) while developing risk assessment models of UAVs and for other autonomous systems. Autonomous systems such as UAVs, Unmanned Ground Vehicles (UGVs), etc. are dependent on their sensor systems to operate optimally. Ensuring that all the logical ports are hardened is vital to building a secure system.

The motivation for this work surfaced from our shared conviction that there is an increasing threat diversity of attack threats to UAVs. A UAV that can fly at high speeds over people and property is a weapon. Just like hacking is prevalent online, a UAV is susceptible to the same risks with far greater consequences.

It has been established by investigations that cheap consumer MAVs are not secure. As presented by Rodday (2015) [15], even professional grade MAVs previously presumed to be secure and hack-proof; used by government agencies, have been found to be susceptible to naïve attacks such as man-in-the-middle attacks. The security of such professional grade MAVs is crucial because of the sensitivity of their missions. Hijacking such drones may lead to loss of property and mission failure amongst other effects.

Due to the lack of adequate depth in cybersecurity of UAVs till recent past, most of the existing autopilot software and their underlying control architecture, state estimation techniques were built without taking security constraints into account. Since UAVs rely heavily on the autopilot software and flight controller to function optimally, it is important

to develop autopilot systems that are robust and secure against adversaries. We hope that our work through this thesis has a contribution towards this goal.

Quadcopters were identified as the system of interest because of their simplicity. The platform is easy to implement for testing. There are only four control variables, which enable 3 degrees of freedom. Furthermore, there are growing applications of quadrotors and other such MAVs in varied industries.

This work examines the problem of resilient state estimation in MAV against sensor spoofing. Resilient state estimation is the study of estimating the system states when sensor measurements are compromised by attackers. It overviews the problem of attack resilient state estimation in autonomous systems where multiple sensors measure the same physical signal. In the scenario, where a malicious attacker may corrupt a subset of these sensors, falsified sensor measurements are passed to the controller, potentially compromising the safety of the system. The range of sensor attacks is evaluated in section 2.3.2.4.

## 1.1. Research Goals

This following research goals were identified at the beginning of the thesis.

RG-1. Identify security vulnerabilities that exist in the system design of MAVs.

RG-2. Review the state estimator used in our system of interest. Study the impact of sensor attacks on the state estimation algorithm. Identify the most critical sensor to the state estimation process.

RG-3. Identify a resilient sensor fusion technique to improve the state estimation process. Integrate resilient sensor fusion technique with the generic state estimator.

RG-4. Verify improvement in resilience to sensor spoofing attacks with the proposed state estimator through a case study on a MAV.

## 1.2. Organization of the Thesis

The thesis is divided into 6 chapters. In chapter 1 the need for a study for cybersecurity consideration for MAVs is introduced, it identifies the need for the investigation and the research goals. Chapter 2 summarizes the numerous security vulnerabilities and threat vectors that exist in the current MAV models available to the public sector. Following which, a synopsis of the state estimation techniques used in the system of interest is covered in chapter 3. This background on state estimation is important to understand the critical sensors in a MAV since it highlights the impact of sensor attacks on the system. Chapter 4 outlines Brooks-Iyengar Fusion as a resilient sensor fusion technique to defend against sensor fusion attacks. The thesis proposes integrating Brooks-Iyengar Fusion with a generic state estimator. In Chapter 5, the performance of the proposed state estimator is showcased through results of a software in the loop simulations in the presence of sensor spoofing attacks. The improvement in the state estimation process with the proposed state estimator is evident. In chapter 6, the thesis is concluded with a summary of research goals covered and a direction of future research. Figure 1, describes how the scope of this research was narrowed.

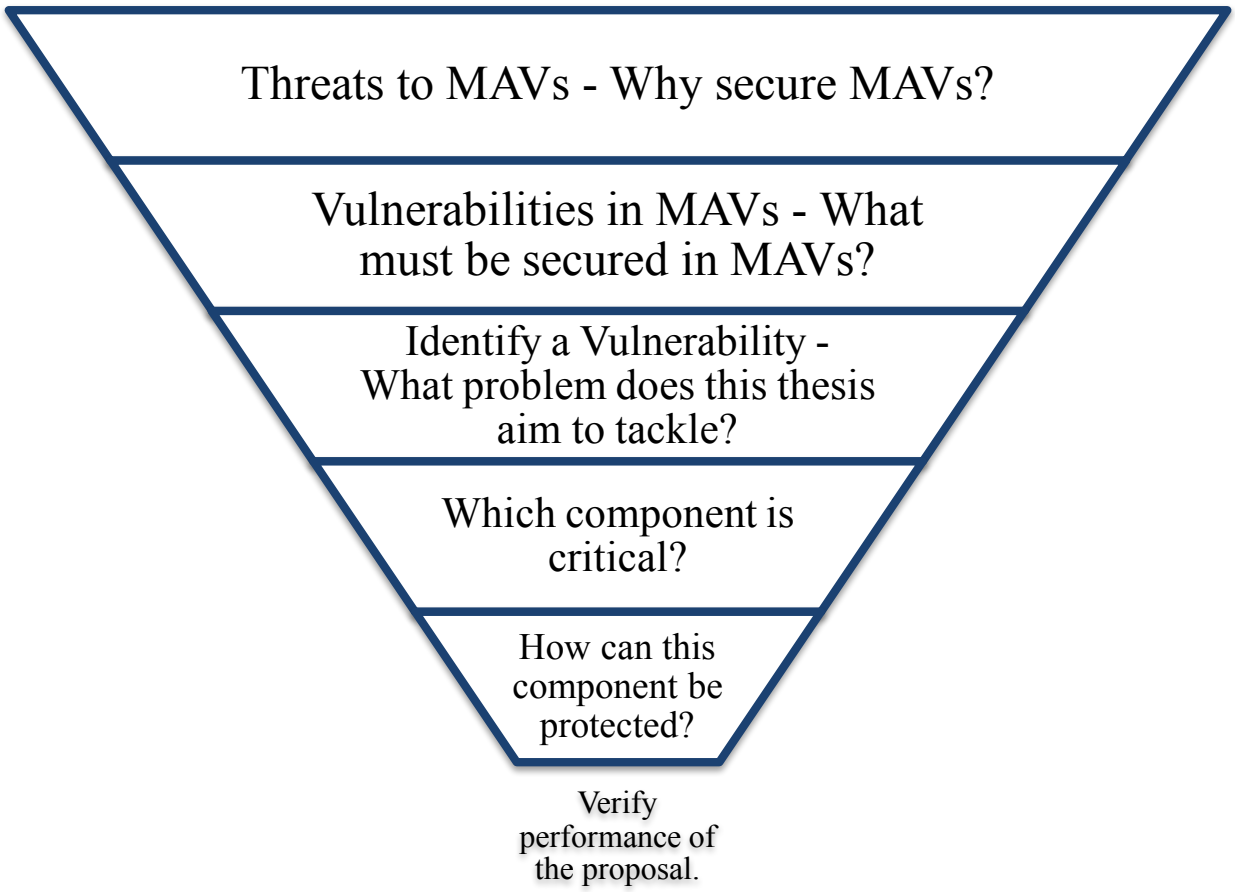


Figure 1: Outline of the Thesis

# 2. Vulnerability Analysis of Unmanned Air Vehicles

In this chapter, our system of interest is described and the different kind of threat vectors to a MAV are reviewed. Following which a case highlighting the need for securing against sensor attacks in a MAV is presented and the security problem that this thesis aims to tackle is identified. As defined by the Department of Defense [16] :

*"A powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or non-lethal payload. Ballistic or semi-ballistic vehicles, cruise missiles, and artillery projectiles are not considered unmanned aerial vehicles."*

The primary difference between a UAV and normal air vehicle is that no crew is required to be on board the vehicle. The vehicle may be remotely controlled or programmed for autonomous operation under supervision.

## 2.1. Types of UAVs

The following classification of UAVs is based on Austin [17] . Austin classifies the UAVs based on the size and capability of the air vehicle to carry out a given mission. However, there may be scenarios where a user or a system employs the air vehicle for different types of missions. The focus of this work is primarily resilient state estimation problem for MAVs. However, the study can be extended to other autonomous vehicles after suitable modifications.

| Type | Purpose |
|---|---|
| HALE (High altitude long endurance) | These are operated by Air Forces from fixed bases. They can fly at high altitudes of 15,000m and can operate for more than 24 hours. They are generally armed and are used for long range (trans global) reconnaissance and surveillance. |
| MALE (Medium altitude long endurance) | They operate at comparatively shorter ranges (greater than 500 km). They can fly at altitudes of 5,000m to 1,500m, but from fixed bases. |
| TUAV (Tactical UAV) | These vehicles are smaller and have simpler than HALE and MALE. They are operated by land and naval forces and have a range of up to 500km. |
| Close-Range UAVs | They operate at ranges of 100 km and are used in fields, including roles such as reconnaissance, target designation, NBC monitoring ship-to-shore surveillance, airfield security, power-line inspection, crop-spraying, etc. |
| MUAV or Mini UAV | These UAV's of below a specific mass generally below 20 kg. They operate at a range of about30 km. They are primarily used by mobile battle groups and for numerous civilian purposes. |
| Micro UAV or MAV | The MAVs were initially described as a UAV having a wing-span of 150 mm or less. This condition has been relaxed now. They are mostly used for urban operations and for commercial purposes. |

Table 1: Classification of UAVs. Austin [17]

## 2.2. System of Interest:

The system of interest is a quadrotor that uses a Pixhawk flight controller. The flight controller can be programmed using a ground control station or controlled using a radio

control as indicated in Figure 3. The MAV may be set for autonomous navigation using an on-board computer.

Pixhawk is an open-hardware project that was conceptualized and developed in consequence to the PIXHAWK Project at the Autonomous Systems Lab at ETH Zurich. It was primarily made to provide sophisticated autopilot hardware and software to academic, hobby, and industrial communities at low costs. This platform was identified based on its outreach, capabilities, and its market share. It is an industry standard and designed by the team in conjunction with 3D Robotics and the ArduPilot Group. The technical specifications as specified are [18]:

- 168 MHz Cortex M4F CPU (256 KB RAM, 2 MB Flash)
- Sensors: 3D Accelerometer / Gyroscope / Magnetometer / Barometer
- Integrated backup, override and failsafe processor with mixing
- microSD card slot, 5 UARTs, CAN, I2C, SPI, ADC, etc.

The main components in our system as indicated in Figure 1 include:

1. PIXHAWK Flight Controller
2. GPS/3DOF Measurement Sensor
3. RC Receiver
4. Ground Control Station
5. Radio Controller
6. ESC and Motors.


Figure 2: Different components in our system of interest.

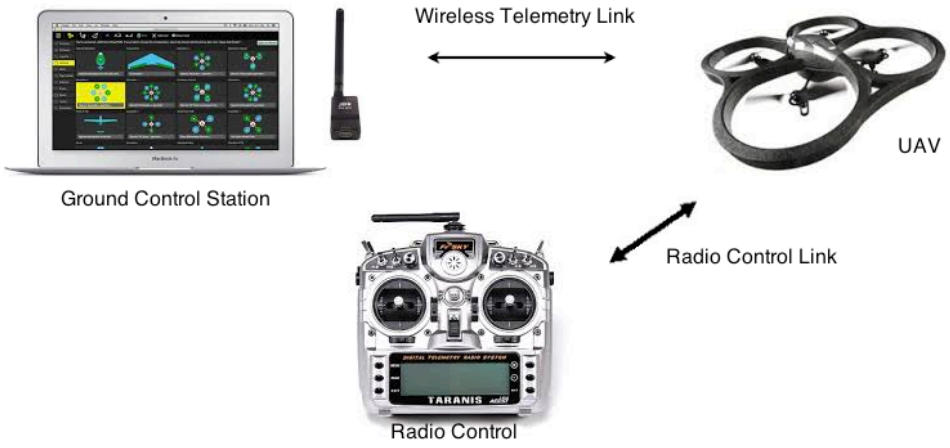


Figure 3: Control Channels for our System of Interest

## 2.3. Threat Analysis:

The type of attacks that may be mounted on a given MAV being operated autonomously, or being controlled by the operator is overviewed in this section. The intentions of the attacker may be unknown. As discussed in [19], vulnerabilities can be classified broadly into application logic attacks where the attacker manipulates the sensor readings and other inputs into the control system and control system attacks that damage the normal behavior of the control system.

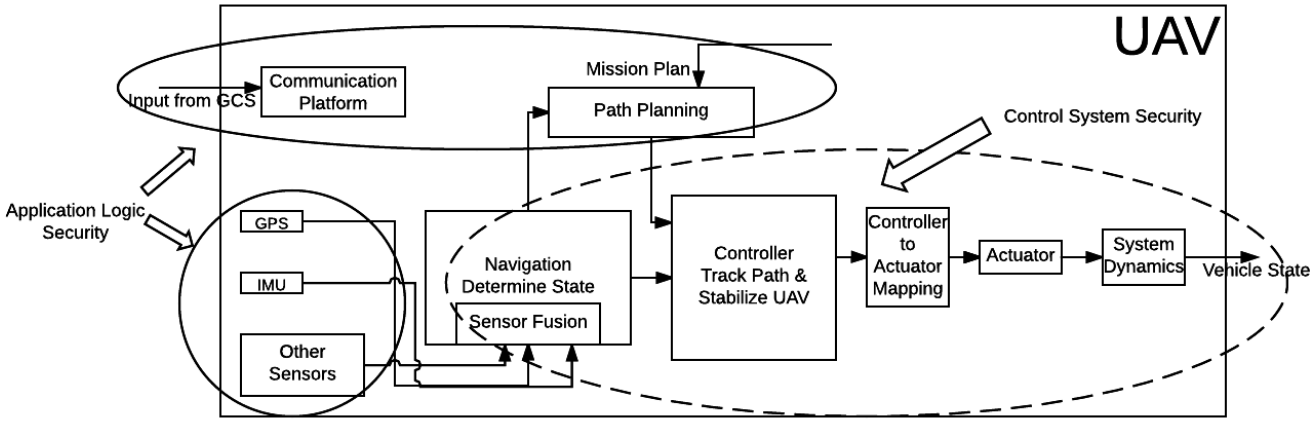


Figure 4: Threats to a UAV

### 2.3.1. Physical Attacks

Physical attacks could include physically targeting the MAV with a tool or a weapon. It could be a missile launcher or a gun targeting the MAV. However, the cost for mounting such an attack would far outweigh the incentive. Other attacks include flying another drone into the target drone. One may also use other tools such as high-powered magnetic beams to damage the electronic circuits on the drone. One may throw a net on the MAV to stop the rotors and bring it down.

### 2.3.2. Logical Attacks

A brief description of the system structure and the exchange of control signals is described in Figure 4. There are multiple control channels and several technologies enabling these channels to control flight parameters such as position, pose, altitude, waypoints, etc. Since these MAVs are autonomous and are generally not controlled actively, a compromised channel may severely damage the mission goals. If an attacker changes high level control signals such as way points or even change parameters such as calibration parameters of the sensors, MAV may be lost or damaged. Total control of the MAV may be lost without any scope of recovery if system parameters are changed or a malware software is installed on board.

The control channels may use different technologies such as 4G LTE, Radio, XBee, etc. Many of the low cost and low power control channel technologies do not have secure protocols and are susceptible to cyber-attacks. Consequently, a lot of MAV manufacturers are limited from installing security protocols and authentication modules because of the limited processing capabilities. Also, using regular encryption and authentication techniques adds delay corresponding to encryption and initial key exchange.

In [20], Villasenor points out the growing concern of counterfeit electronics. He argues that the electronics supply chain could be intentionally compromised during design. If these vulnerabilities are placed into the design prior to manufacture with sufficient skill, they would be extremely difficult to detect. These backdoors could be exploited years later to intentionally compromise the system containing the chip.

#### *2.3.2.1. Control Channel Attacks*

As observed in Figure 5, there are multiple control signals that may be used to send forged data to the MAV. This is possible because of the absence of application layer encryption in the MAV - GCS communication channel or the lack of authentication protocols. Without a method to verify if the data received through the communication channel is legitimate or not, the flight controller ought to assume that the data received is correct. This permits forged signals to be processed as any other correct signals from an authorized operator. This is a critical flaw that may be used to mount attacks such as changing the mission plan, changing way points, etc. and may lead to loss of control of the MAV permanently. Sometimes proprietary protocols are used without proper encryption and authentication techniques. Even though these techniques may provide some level of security, these protocols can be reverse engineered or an attacker might just mount a replay attack leading to unexpected behavior.

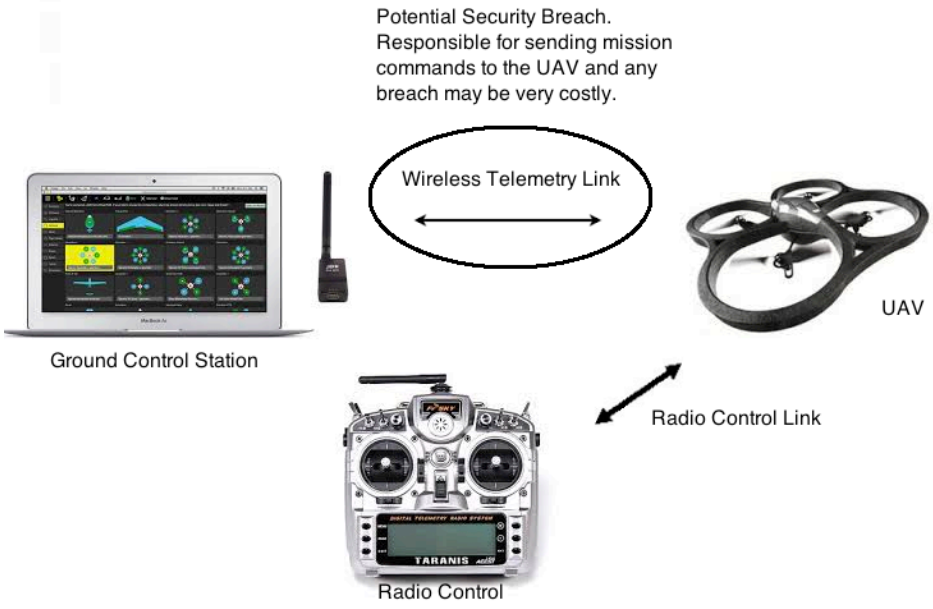


Figure 5: Control Channel Attack

#### *2.3.2.2. Denial of Service Attacks (DoS):*

An attacker may use a simple Denial of Service attack on any of the control channels which use wireless communication. Anyone close to either the transmitter or the receiver can receive signals and analyze the packet information, making it vulnerable. Both the receiver and the transmitter use the same channel for communication; the adversary may flood this channel with bogus signals and thereby prevent legitimate messages and signals from being sent and received. A hobby MAV, the AR.Drone operated using smartphones through a local WIFI network was de-authenticated using such DoS attacks [21] .

#### *2.3.2.3. Replay Attacks*

As suggested in section 2.3.2.2, in case of proprietary protocols used as security measures, one may use replay attacks to disrupt the normal operation of a MAV. One may even use the same type of replay attacks even if the messages were encrypted. This would include recording the signals and replaying them at a later period. This attack is only feasible when there are no authentication modules to verify the origin of these messages.

#### *2.3.2.4. Sensor Attacks*

Sensor attacks can be mounted by installing malicious software on the flight controller or the processor responsible for processing the sensor information. The software modifies the sensor information before it is passed to the flight controller. The Stuxnet malware is a famous example of such an attack [22]. Often, MAVs may rely on networked infrastructure for sharing sensor information. This may be for swarm robotics, VICON systems, or WIFI localization systems. An attacker may degrade sensor measurements by manipulating the data packets being exchanged between different modules, elements or subsystems of the system. Adversaries may also spoof sensors by tampering the sensor hardware externally

or modifying the sensor environment. This may lead the sensor to pass false information of the value of physical signal it attempts to measure.

An attempt to externally manipulate the sensor measurements in a MAV was described in [23]. In [23], Son et al. presented their work on "Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors". They described a method using consumer grade electronics to introduce noise in gyroscope sensor measurements. They exploited the fact that resonant frequencies of many MEMS based gyroscopes are found in the audible frequency band. They also present the effect of resonant/attacked output of the gyroscopes on the flight control of the drone. They describe the attacking technique and establish the consequence of such attacks, refer to Figures 6 and 7. In figure 6 and figure 7, region A and region C correspond to the gyroscope operating under normal conditions, whereas region B corresponds to the region where the gyroscope is under attack (Fig 6.a.). The operator increases the throttle of the quadrotor to increase its altitude (Fig 6.b.); as soon as the attack starts, the altitude drops and the quadrotor crashes (Fig 7.b.).

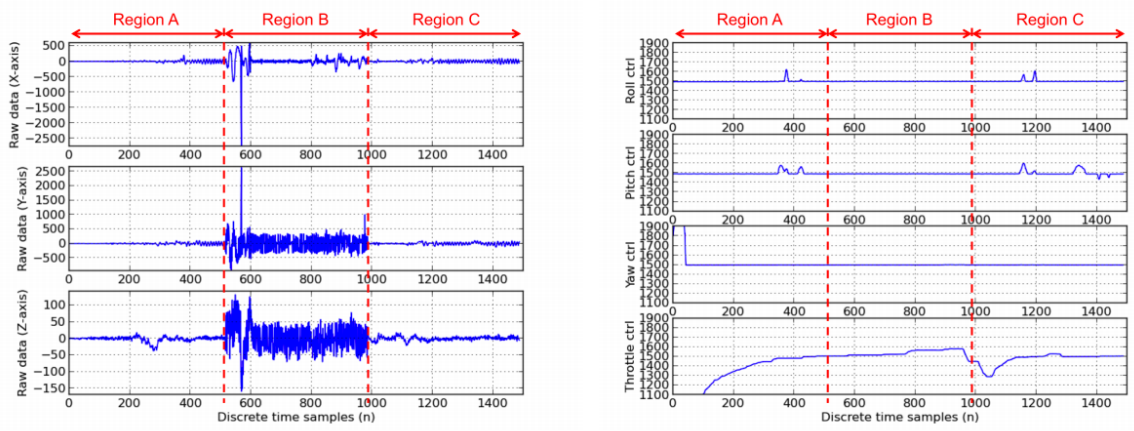


Figure 6: a) Gyroscope Sensor Data. b) Operator Control Signals [23]

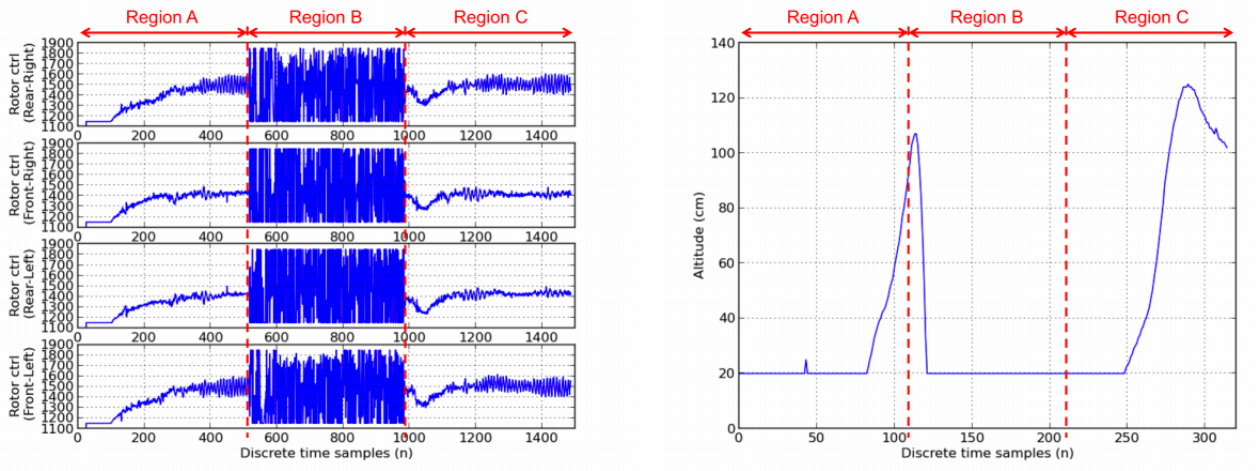


Figure 7: a) Rotor Control Data. b) Altitude of the System [23]

2.3.2.4.1. Global Positioning System (GPS)

DoS attacks like the one mentioned in section 2.3.2.2 can be mounted on GPS modules by flooding garbage signals on the target GPS frequencies. This prevents correct signals from passing through and disrupts localization and state estimation. There are devices available that enable such jamming with different levels of sophistication. GPS jamming could be used to disrupt the mission by preventing the MAV to reach its destination.

GPS spoofing is also another plausible threat vector. One may spoof GPS signals and introduce them to the channel. The under-attack MAV may use the forged signals in its state estimation calculations. Accordingly, an attacker may convince the MAV to redirect its path because of a falsely identified location.

### *2.3.2.5. Software Attacks*

There are two components to the software: the software on the ground control station and the on-board software for flight control for conducting the autonomous mission. The ground control station is used for programming flight paths and setting flight parameters.

Prior to the mission (or during the mission) the GCS programs the MAV with the mission data including waypoints and other mission information. In most configurations, the GCS is connected to the flight computer or flight controller constantly through a wireless channel. The attacker may use malware onto the flight controller or on the ground control station. This would allow the attacker to reprogram the flight mission and control the flight as suited. The following scenarios are possible:

- The ground control station is connected to the Internet and the attacker uses the flight planner in real time to obtain live control of the MAV.
- The attacker may introduce malware on the ground control station, which may preprogram the flight as per the intentions of the attacker.
- A lot of hobby drones use generic ground control station software with an unsecure communication link. One might use this vulnerability to force pair their computer using a radio link.

## 2.4. Summary

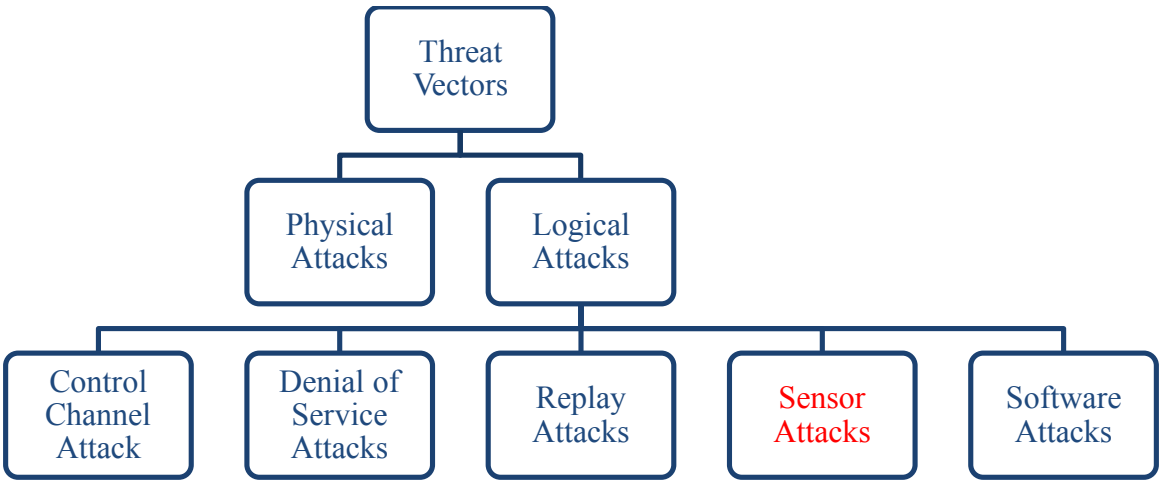


Figure 8: Different threat vectors for a MAV and the focus of this thesis

In this chapter, the threat vectors for a MAV were identified. The threat vectors can be broadly classified into physical attacks and logical attacks. Logical attacks were further classified into control channel attacks, denial of service attacks, replay attacks, man in the middle attacks, sensor attacks, and software attacks. Most of the logical attacks are application layer security problems and must be tackled using encryption, authentication and other application logic security techniques. However, sensor attacks, actuator attacks and controller attacks are control system security problems. The work of Son et al. [23] is presented to highlight the impact of sensor attacks on a MAV. The fact that the MAV crashes as soon as there is an attack on it is an important observation. This highlights the importance of studying the problem of resilient state estimation. The work in this thesis discusses the problem of defending against sensor attacks for MAVs. Next, in Chapter 3, the state estimation process in a current state of the art flight controller is reviewed to identify the sensor that must be safeguarded.

# 3. Background on State Estimation in MAVs

This chapter discusses the state estimation technique and a modular approach to state estimation implemented in a flight controller. A controller in a MAV relies heavily on information on the current state of the system. The controllers need information on the states (position, attitude) at a high rate with minimum delay because of the fast dynamics of the system. Inertial Measurement Unit (IMU) form the core of the state estimation algorithm. IMUs consist of a gyroscope to measure angular velocity and an accelerometer to measure the linear acceleration. Most MAVs can carry an IMU because of its lightweight, inexpensiveness, and low energy requirements. In addition to noisy sensor readings, the sensors are affected by a continuous time-varying bias; thereby making a continuous accurate estimation of the pose in aggressive flights difficult and error prone. Hence, IMUs are used in conjunction with other sensors such as cameras, Sound Navigation and Ranging (SONAR), Light Radar (LIDAR), Global Positioning System (GPS), etc. to support an accurate estimate.

IMUs have a high update frequency but with drifting measurements leading to error accumulation. Sensors such as cameras, GPS, and laser systems have slowly drifting measurements but at a much slower update rate. As suggested by Weiss (2012) [24], sensors can be classified based on their drift speed and frame rate as in Figure 1. IMUs have a high update rate but with accumulating errors, whereas GPS and Laser Trackers provide information at low drift rates but with lowest update rates. The ideal sensor would

lie at the top left corner of the graph in Figure 7. Consequently, it's necessary to have sensors in addition to the IMU to enable good state estimation. The analysis in this chapter is based on the work of Weiss (2012) [24] and Achtelik (2013) [25]. The discussion analyzes a 6 Degree of Freedom (DOF) sensor (a camera) along with a 3DOF position sensor (GPS) in addition to the on-board IMU. The discussion follows an Extended Kalman Filter Framework to establish an algorithm to address the issue of drifting measurements and low update rates.

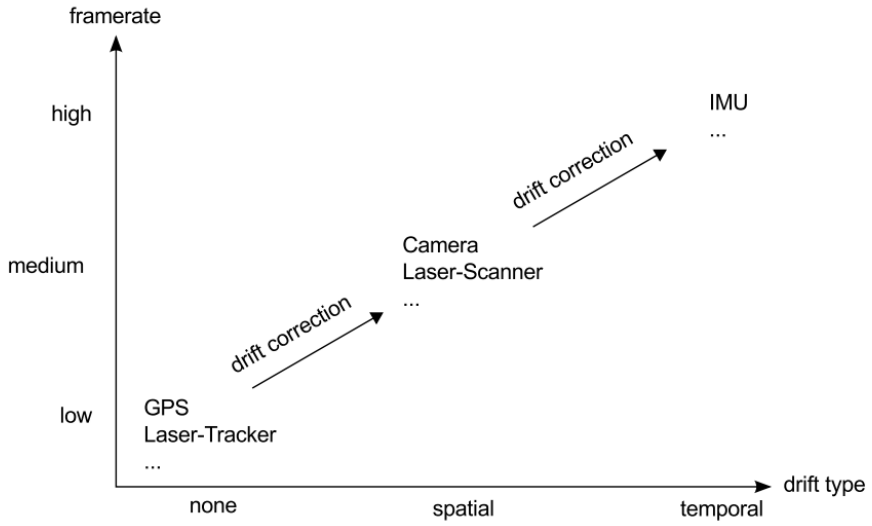


Figure 9: Classification of sensors based on drift type and update rate [24]

## 3.1. Extended Kalman Filter Framework

The Extended Kalman Filter (EKF) is useful in MAVs because it is recursive and does not need a history of state estimates to establish the current state estimate. It does so by minimizing the error of the observed estimates. Within a EKF, there is a prediction step followed by a correction step. In the prediction step, the framework predicts the states of

the system and the uncertainty of the process (covariance propagation) using the model of the system. In the correction step, a sensor measurement is applied to the sensor model, which corrects the state of the system and is used to update the uncertainty of the estimate. The step is referred to as correction and covariance update.

Section 3.1.1 explores the EKF framework for an MAV as described in [24] and [25]. This chapter aims to identify the most critical sensor with respect to state estimation process.

### 3.1.1. MAV Process Model vs Direct IMU Input

There are two techniques/models for state prediction.

| MAV Model | IMU Model |
|---|---|
| When, a MAV Model is used for the prediction process. It uses the torques and thrusts originating from the rotors as inputs to the process Model. | IMU measurements use linear acceleration and angular velocity as inputs to the process model. Acceleration is integrated to calculate velocity, which is integrated to calculate the position. |

Table 2: Comparing the MAV and IMU model

The IMU measurement model is advantageous because of lower computational requirements. This is better exemplified when one analyzes the state size and weighs the cost of IMU updates in the range of 1kHz. The following comparison is expressed in [24].

1. State Size: For the MAV model, the input to the process model will consist of angular momentum. This process model will require additional vector states. Update and covariance predictions steps run at high frequencies and require dense matrix multiplications making it computationally inefficient.

2. Measurement update: As discussed in section 3.1.1, MAV process model has angular momentum from rotor speeds as its inputs. IMU readings are used for EKF measurement updates. Consequently, EKF updates must be computed for each IMU measurement. At high measurement rates tending to 1kHz it gets computationally expensive. However, when one use an IMU based model, IMU measurements appear only in the EKF prediction step, which needs only a few calculations despite the frequency. There are also benefits when analyzing the computational complexity.
3. Model uncertainty: The system model may not be accurate for all platforms. The MAV model is not constant and may change based on different payloads and different positioning of payloads. The moment of inertia, mass, rotor constants may change based on the platform used. These model parameters can be estimated using our EKF framework at a cost of a few additional states, again adding to computational complexity. However, in the case of IMU based models, one must measure the current linear acceleration and angular velocity to evaluate the position and orientation.
4. Un-modeled disturbances: There may be un-modeled disturbances such as wind gusts. The IMU model is advantageous because it rejects noisy IMU measurements. IMUs measure disturbances accurately but take a few time updates and corrections before the MAV realizes it was a real disturbance. Modeling such disturbances for the MAV model is a difficult task. Researchers have tried to model wind speeds and wind gusts. Modeling wind gusts requires many assumptions and is a complex analysis. One would have to incorporate these disturbances in his MAV model and for accurate state estimation. However, in the case of an IMU model, disturbances are measured directly and can be incorporated in the prediction step.

### 3.1.2. Mathematical Background – Quaternion Algebra

This section is based on the work done by Trawny and Roumeliotis (2015) [26]. Below is a brief discussion on quaternion algebra.

Any quaternion $\bar{q}$ is defined as:

$$\bar{q} = q_4 + q_1 i + q_2 j + q_3 k$$

where $q_4$ is the real part and $q_1 i + q_2 j + q_3 k$ is the imaginary part. Here $q_1, q_2, q_3$ are real numbers, and $i, j, k$ are hyperimaginary numbers. The quaternion can also be written as a 4-dimenstional-column matrix:

$$\bar{q} = [q_1 \ q_2 \ q_3 \ q_4]^T$$

Let $\bar{q}$ and $\bar{p}$ be two quaternions, then quaternion multiplication can be defined as:

$$\bar{q} \otimes \bar{p} = (q_4 + q_1 i + q_2 j + q_3 k)(p_4 + p_1 i + p_2 j + p_3 k)$$

$$\bar{q} \otimes \bar{p} = \begin{bmatrix} \bar{q} & -\bar{q}^T \\ \bar{q} & \bar{q} I_3 + \lfloor \bar{q} \times \rfloor \end{bmatrix} . \bar{p} = \begin{bmatrix} \bar{p} & -\bar{p}^T \\ \bar{p} & \bar{p} I_3 + \lfloor \bar{p} \times \rfloor \end{bmatrix} . \bar{q}$$

The skew-symmetric matrix operator represented as $\lfloor \bar{q} \times \rfloor$ is defined as:

$$\lfloor \bar{q} \times \rfloor = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}$$

The cross product of two quaternions can also be written as: $q \times p = \lfloor q \times \rfloor p$

## 3.2. Extended Kalman Filter Setup

For the EKF setup, the IMU process model is used based on the discussion in section 3.1.1. The prediction and covariance propagation steps for the core states is discussed in the following sections. Other sensors may add additional or different states, however the formulation of the core set of states obtained from the IMU does not change, because IMU measurements are always available to the filter.

### 3.2.1. Inertial Sensor Model

The IMU measurements are perturbed by a bias $b$ and additive white Gaussian noise $n$. Thus, the real angular velocity $\omega$ and the real linear acceleration $a$ in the IMU frame can be modeled as:

$$\omega = \omega_m - b_\omega - n_\omega$$

$$a = a_m - b_a - n_a$$

$$E[n_\omega / n_a] = 0_{3\times1}$$

where $\omega$ is the angular velocity, $a$ is the acceleration, subscript $m$ refers to the measured value, $b$ refers to the bias and $n$ is the additive white Gaussian noise. Here $E[n_\omega / n_a]$ is the expectation of the noise signals $n_\omega, n_a$. Additive refers to noise that is added to any noise that may be intrinsic to the system. The bias dynamics is modeled as a random walk process with zero mean white Gaussian noise as its time derivative. Thus, we model the bias as:

$$\dot{b}_\omega = n_{b_\omega}$$

$$\dot{b}_a = n_{b_a}$$

$$E\left[n_{b_\omega}\right] = 0_{3\times1}$$

$$E\left[n_{b_a}\right] = 0_{3\times1}$$

### 3.2.2. State Representation

The state of the filter is described as:

$$x_c = \left[{p_\omega^i}^T \quad {v_\omega^i}^T \quad {\bar{q}_\omega^i}^T \quad b_\omega^T \quad b_a^T\right]^T$$

where $p_\omega^i$ is the position of the IMU in the world frame, $v_\omega^i$ is the velocity of the IMU in the world frame, $\bar{q}_\omega^i$ is the quaternion describing rotation from the world frame to the IMU frame, $b_a$ and $b_\omega$ are the respective biases. Also, the hat operator placed over state variable

notations refer to the estimate of that variable, for example $\hat{x}_c$ is the estimate of the core set of states for our state estimator.

The equations that govern the abovementioned states are:

$$\dot{p}_\omega^i = v_\omega^i$$

$$\dot{v}_\omega^i = C_\omega^i.(a_m - b_a - n_a) - g$$

$$\dot{\bar{q}}_\omega^i = \frac{1}{2}.\bar{q}_\omega^i \otimes \begin{bmatrix} 0 \\ \omega_m - b_m - n_m \end{bmatrix} \text{ as } \dot{\bar{q}}_\omega^i = \frac{1}{2}\bar{q} \otimes \bar{\omega}$$

$$\dot{b}_\omega = n_{b_\omega}\ \dot{b}_a = n_{b_a}$$

Please note that g is the gravity vector, and $C_\omega^i$ is the rotation matrix computed using $\bar{q}_\omega^i$. It is essential to include the biases as states, since these drifts accumulate over time and must be updated online. Quaternions are used to represent the attitude of the MAV.

Next, calculate the error vectors for each of the states. Note that, as suggested in [26], the error in the quaternion value is calculated as $\delta\bar{q}$, instead of the difference between $\bar{q}\ and\ \hat{\bar{q}}$ (the actual value of the quaternion and the quaternion estimate respectively). Hence, using the small angle approximation:

$$\bar{q} = \ \hat{\bar{q}} \otimes \delta\bar{q} \Leftrightarrow \delta\bar{q} = \hat{\bar{q}}^* \otimes \bar{q}$$

If the rotation corresponding with the error quaternion $\delta\bar{q}$ is very small, we can use the small angle approximation and calculate the error angle vector $\delta\theta$ as:

$$\delta\bar{q} = \begin{bmatrix} \delta q \\ \delta q_4 \end{bmatrix} \approx \delta\bar{q} = \begin{bmatrix} 1 & \frac{1}{2}\delta\theta^T \end{bmatrix}^T$$

where $\delta\theta$ is the angle vector. Similarly, in the case of rotation matrices, as suggested in [25],

$$C = \hat{C}.\Delta C \Leftrightarrow \Delta C = \hat{C}^T.C$$

$$\Delta C \approx I_3 + \lfloor \delta\theta_\times \rfloor$$

For other states, use the arithmetic difference to calculate the error, i.e., $\tilde{x} = x - \hat{x}$. Hence, as deduced in [26] and [24], the error in the state variables can be written as:

$$\tilde{x}_c = \left[ {\Delta p_\omega^i}^T \;\; {\Delta v_\omega^i}^T \;\; {\delta\theta_\omega^i}^T \;\; \Delta b_\omega^T \;\; \Delta b_a^T \right]$$

$$\Delta\dot{p}_\omega^i = \Delta v_\omega^i$$

$$\Delta\dot{v}_\omega^i = -\hat{C}_\omega^i . \left[ \left( a_m - \hat{b}_a \right)_\times \right] . \delta\theta^i - \hat{C}_\omega^i . \Delta b_a - \hat{C}_\omega^i n_a$$

$$\delta\dot{\theta}_\omega^i = \left[ \left( \omega_m - \hat{b}_\omega \right)_\times \right] . \delta\theta_\omega^i - \Delta b_\omega - n_\omega$$

$$\Delta\dot{b}_\omega = n_{b_\omega}$$

$$\Delta\dot{b}_a = n_{b_a}$$

The above equations can be summarized as $\dot{\tilde{x}}_c = F_c^c . \tilde{x}_c + G_c^c . n$. Linearizing above equations using noise vector $n_c = \left[ n_a^T \;\; n_{b_a}^T \;\; n_\omega^T \;\; n_{b_\omega}^T \right]$ around $\hat{x}_c$. Here, the superscript c used in the notation refers to the continuous time domain. As indicated in [25] and [24], see that:

$$\dot{\tilde{x}}_c = \left. \frac{\partial \dot{\tilde{x}}_c}{\partial \tilde{x}_c} \right|_{\hat{x}_c} . \tilde{x}_c - \left. \frac{\partial \dot{\tilde{x}}_c}{\partial n} \right|_{\hat{x}_c} . n = F_c^c . \tilde{x}_c + G_c^c . n$$

Calculate the noise covariance matrix for the discrete case as described in [24]:

$Q_c^d = \int_{\Delta t} F_c^d(\tau) . G_c^c . Q_c^c . G_c^{c,T} . F_c^d(\tau)^T d(\tau)$, where

$$F_c^d = \begin{bmatrix} I_{d_3} & \Delta t & A & B & -C_{\hat{q}_\omega^i}^T \Delta t^2 & 0_{13\times3} \\ 0_3 & I_{d_3} & C & D & -C_{\hat{q}_\omega^i}^T \Delta t & 0_{13\times3} \\ 0_3 & 0_3 & E & F & 0_3 & 0_{13\times3} \\ 0_3 & 0_3 & 0_3 & I_{d_3} & 0_3 & 0_{13\times3} \\ 0_3 & 0_3 & 0_3 & 0_3 & I_{d_3} & 0_{13\times3} \\ 0_{13\times3} & 0_{13\times3} & 0_{13\times3} & 0_{13\times3} & 0_{13\times3} & I_{d_{13}} \end{bmatrix}$$

$$A = -C_{\hat{q}_\omega^i}^T \lfloor \hat{a}_\times \rfloor \left( \frac{\Delta t^2}{2} - \frac{\Delta t^3}{3!} \lfloor \omega_\times \rfloor + \frac{\Delta t^4}{4!} \lfloor \omega_\times \rfloor^2 \right)$$

$$B = -C_{\hat{q}_\omega^i}^T \lfloor \hat{a}_\times \rfloor \left( \frac{-\Delta t^3}{3!} + \frac{\Delta t^4}{4!} \lfloor \omega_\times \rfloor - \frac{\Delta t^5}{5!} \lfloor \omega_\times \rfloor^2 \right)$$

$$C = -C_{\hat{q}_\omega^i}^T \lfloor \hat{a}_\times \rfloor \left( \Delta t - \frac{\Delta t^2}{2!} \lfloor \omega_\times \rfloor + \frac{\Delta t^3}{3!} \lfloor \omega_\times \rfloor^2 \right)$$

$$D = -A$$

$$E = I_d - \Delta t \lfloor \omega_\times \rfloor + \frac{\Delta t^2}{2!} \lfloor \omega_\times \rfloor^2$$

$$F = -\Delta t + \frac{\Delta t^2}{2!} \lfloor \omega_\times \rfloor - \frac{\Delta t^3}{3!} \lfloor \omega_\times \rfloor^2$$

$$Q_d = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 \\ -C_{\hat{q}_\omega^i}^T & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & -I_{d_3} & 0_3 \\ 0_3 & 0_3 & 0_3 & I_{d_3} \\ 0_3 & I_{d_3} & 0_3 & 0_3 \\ 0_{13\times3} & 0_{13\times3} & 0_{13\times3} & 0_{13\times3} \end{bmatrix}$$

where $Q_c^c$ is the noise covariance matrix for the continuous case, $Q_c^c = diag(\sigma_{n_a}^2\ \sigma_{n_{b_a}}^2\ \sigma_{n_\omega}^2\ \sigma_{n_{b_\omega}}^2)$. The superscript $c$ in $F_c^c$ refers to the continuous space, whereas the superscript $d$ in $Q_c^d$ refers to the discrete space, $I_{d_3}$ is a 3x3 identity matrix.

### 3.2.3. Propagation Steps:

The propagation steps for the state variables is as follows:

1. Using the state variable differential equations propagate the state variables. For the quaternion, use the quaternion integration method described in [26]:

$$\bar{q}(t_{k+1}) = \left( \exp\left( \frac{1}{2} \Omega(\bar{\omega}) \Delta t \right) + \frac{1}{48} \left( \Omega\big(\omega(t_{k+1})\big)\Omega\big(\omega(t_k)\big) - \Omega\big(\omega(t_{k+1})\big)\Omega\big(\omega(t_{k+1})\big) \right) \Delta t^2 \right) \bar{q}(t_k)$$

2. $F_c^d\ and\ Q_c^d$ are calculated based on the methods described in [24].

3. State covariance matrix is calculated using $P_{k+1|k} = F^d P_{k|k} F^{d,T} + Q^d$ [25].

### 3.2.4. Measurement Models:

As noted earlier in the introduction of chapter 3, for robust state estimation, additional sensors are needed around the IMU. These sensors help in correcting drift from integration of IMU sensors. Note that the origin of the additional sensors does not coincide with that of the IMU. Hence, additional states must be added to the core state $x_c$. These states include the displacement $p_i^s$ of the sensor from the IMU for the 6DOF and 3DOF sensors, the rotation $q_i^s$ of the sensor from the IMU frame, and a scaling factor $\lambda$. Note that these states do not have any associated dynamics and are not updated during the propagation step:

$$\dot{p}_i^s = 0$$

$$\dot{\lambda} = 0$$

$$\dot{\vec{q}}_i^s = 0$$

#### 3.2.4.1. 6DOF Sensor:

As mentioned in section 3.2.4, one needs additional states for a 6DOF sensor. The new state is defined as in [24]:

$$x = \begin{bmatrix} x_c & \lambda & p_i^{s^T} & q_i^{s^T} \end{bmatrix}$$

$$z_p = p_\omega^s = \left(p_i^s + C_\omega^i . p_i^s\right). \lambda + n_p$$

where $z_p$ is the position measurement from the sensor and $n_p$ is the additive white Gaussian noise. The position error is defined as:

$$\tilde{z}_p = z_p - \hat{z}_p$$

$$\tilde{z}_p = \left(p_i^s + C_\omega^i . p_i^s\right). \lambda + n_p - \left(\hat{p}_i^s + \hat{C}_\omega^i . \hat{p}_i^s\right). \hat{\lambda}$$

where $\hat{z}_p$ is the position estimate.

Also $z_p$ can be expressed using:

$$p = \Delta p + \hat{p}$$

$$C = \hat{C}.(I_3 + \lfloor d\theta \times \rfloor)$$

As indicated in [24], solving for $\tilde{z}_p$ using the above equations yields:

$$\tilde{z}_p = \Delta p_\omega^i \hat{\lambda} + \hat{C}_\omega^i . \Delta p_i^s . \hat{\lambda} - \hat{C}_\omega^i \lfloor \hat{p}_i^s \times \rfloor . \delta\theta . \hat{\lambda} + \hat{p}_\omega^i . \Delta\lambda + C_\omega^i . \hat{p}_i^s . \Delta\lambda$$

The rotation measurement is

$$z_q = \bar{q}_\omega^s \otimes \bar{q}_s^i \otimes \delta\bar{q}_n$$

Error rotation calculations are used to evaluate the following, like in [24].

$$\bar{q} = \hat{\bar{q}} \otimes \delta\bar{q} \Leftrightarrow \delta\bar{q} = \hat{\bar{q}}^* \otimes \bar{q} \; or \; C = \hat{C}.\Delta C \Leftrightarrow \Delta C = \hat{C}^T.C$$

Rotation error is formulated as in [24]:

$$\Delta C = \hat{C}^T . C . \Delta C_n$$

$$\Delta C = \hat{C}_i^{s^T} . \hat{C}_\omega^{i^T} . C_\omega^i . C_i^s . \Delta C_n \; with \; \Delta C = (I_3 + \lfloor \delta\theta \times \rfloor)$$

$$\lfloor \delta\theta \times \rfloor = \left( I_3 + \hat{C}_i^{s^T} . \lfloor \delta\theta_\omega^i \times \rfloor . \hat{C}_i^s \right) . (I_3 + \lfloor \delta\theta_i^s \times \rfloor) . (I_3 + \lfloor \delta\theta_n \times \rfloor) - I_3$$

Higher order terms such as $\delta\theta \times \delta\theta$ are omitted. These terms only cause errors at the point of linearization where the expected value of these terms is zero. Also note that:

$$\mathrm{C}.\lfloor x_\times \rfloor . C^T = \lfloor C . x_\times \rfloor$$

Hence following the discussion in [24], the estimation error can be calculated as:

$$\lfloor d\theta \times \rfloor \approx \hat{C}_i^{s^T} . \lfloor d\theta_\omega^i \times \rfloor . \hat{C}_i^s + \lfloor d\theta_i^s \times \rfloor + \lfloor d\theta_n \times \rfloor$$

$$\Rightarrow \tilde{z}_q = \delta\theta = \hat{C}_i^{s^T} . d\theta_\omega^i + d\theta_i^s + \delta\theta_n$$

Thereafter, one can calculate the values for the Jacobians using the expressions obtained for $\tilde{z}_q$ and $\tilde{z}_p$ namely. This includes $H_p = \frac{\partial \tilde{z}_p}{\partial \tilde{x}}, H_q = \frac{\partial \tilde{z}_q}{\partial \tilde{x}}, V_p = \frac{\partial \tilde{z}_p}{\partial n_p}, V_q = \frac{\partial \tilde{z}_q}{\partial \delta\theta_n}$. Please

note that $H$ is the observability matrix obtained by stacking $H_p$ and $H_q$, $P$ is the state covariance matrix and $V$ is obtained by stacking $V_p$ and $V_q$.

Following which, proceed with the EKF correction steps as formulated in [24]:

1. Calculate residual, i.e., the error in the estimate: $\tilde{z} = \left[\tilde{z}_p^T \ \tilde{z}_q^T\right]^T$
2. Calculate Innovation: $S = H.P.H^T + V.R.V^{\square}$
3. Calculate Kalman Gain: $K = P.H.S^{-1}$
4. Calculate correction: $\hat{\tilde{x}} = K.\tilde{z}$
5. Correct the states using the following equations:
    a. $\delta\bar{q} = \hat{\bar{q}}^* \otimes \bar{q}; \delta\bar{q} = \left[q \quad \frac{1}{2}\delta\theta^T\right]^T$
    b. $\tilde{x} = x - \hat{x}$ for the remaining states
6. State covariance is updated as:

$$P_{k+1|k+1} = (I + K.H).P_{k+1|k}.(I + K.H)^T + K.V.R.V^T.K^T$$

3.2.4.2. 3DOF Sensor (GPS):

In this section, we analyze inclusion of a 3DoF Sensor such as GPS or a laser tracker, which yields the position of the vehicle in the world frame. The rotation vector is irrelevant and is not included as the state $q_s^i$. Nonetheless, keep the scaling factor and the translational calibration state for the position vector from the IMU to the sensor as suggested in [25]. The state is described as:

$$x = \left[x_c \ \lambda \ p_i^{s^T}\right]$$

where

$$z_p = p_\omega^s = \left(p_i^s + C_\omega^i.p_i^s\right).\lambda + n_p$$

The Jacobian calculations and the EKF update steps remain unchanged [25], [24]:

$$H_p = \frac{\partial \tilde{z}_p}{\partial \tilde{x}}, H_q = \frac{\partial \tilde{z}_q}{\partial \tilde{x}}, V_p = \frac{\partial \tilde{z}_p}{\partial n_p}, V_q = \frac{\partial \tilde{z}_q}{\partial \delta\theta_n}.$$

Following which, proceed with the EKF correction steps as formulated in [24]:

1. Calculate residual: $\tilde{z} = \left[\tilde{z}_p^T\right]^T$
2. Calculate Innovation: $S = H.P.H^T + V.R.V^T$
3. Calculate Kalman Gain: $K = P.H.S^{-1}$
4. Calculate correction: $\hat{\tilde{x}} = K.\tilde{z}$
5. Correct the states using the following equations:
    a. $\delta\bar{q} = \hat{\bar{q}}^* \otimes \bar{q}; \delta\bar{q} = \begin{bmatrix} q & \frac{1}{2}\delta\theta^T \end{bmatrix}^T$
    b. $\tilde{x} = x - \hat{x}$ for the remaining states
6. State covariance is updated as:

$$P_{k+1|k+1} = (I + K.H).P_{k+1|k}.(I + K.H)^T + K.V.R.V^T.K^T$$

## 3.3. Vulnerability of the State Estimator

In section 3.1 and section 3.2, a background on EKF based state estimator frame work was provided to highlight the importance of the IMU at the core of the state estimator. The IMU measurements form the core of the EKF framework and are explicitly used in the "Time Update – Prediction Step" and implicitly affect the calculations of the state covariance and noise covariance matrix. Hence any attack on the IMU compromises the state estimator. Figure 10 depicts the normal operation of the state estimator and Figure 11 illustrates the impact of an attack on the IMU on the state estimator.

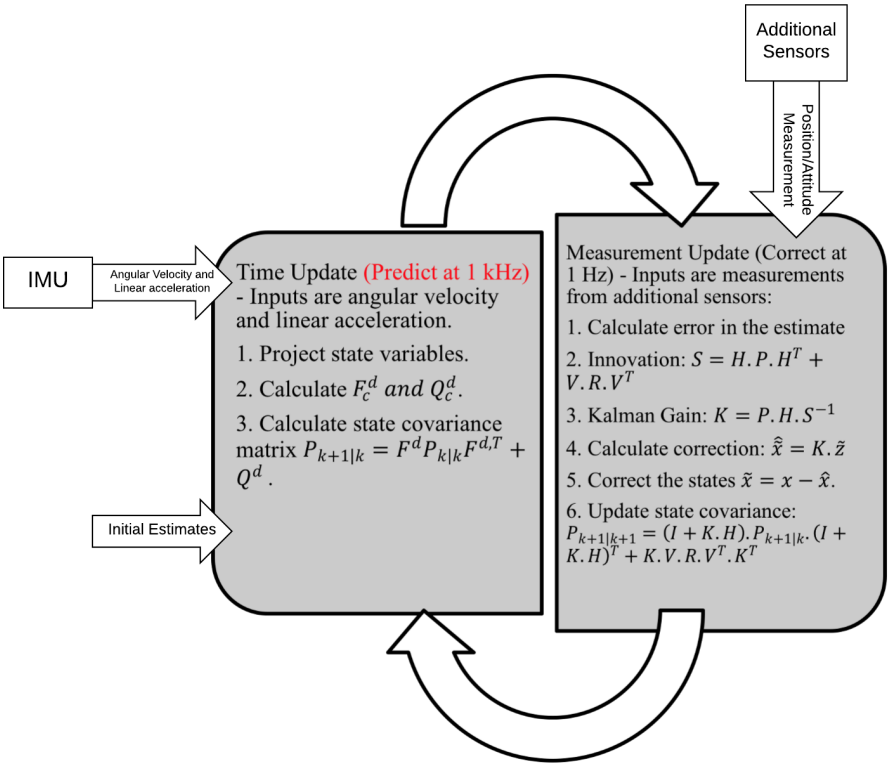


Figure 10: Extended Kalman Filter Framework

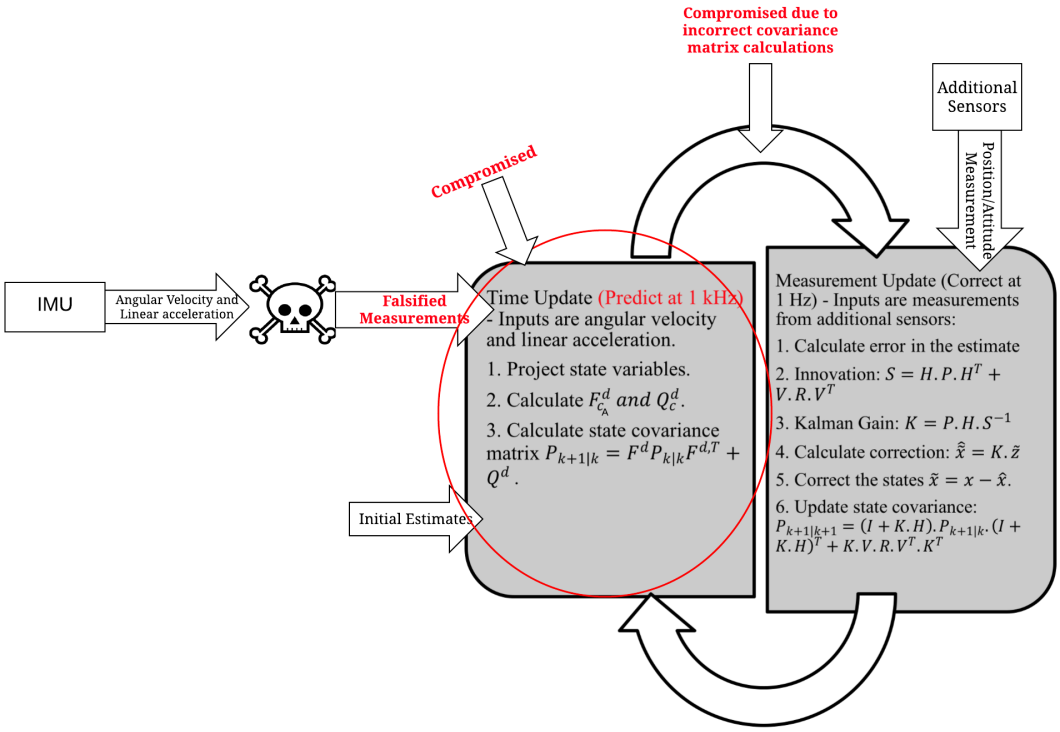


Figure 11: State Estimator in the presence of an adversary

## 3.4. Summary

This chapter presents a background on the essential components of state estimation in autonomous MAVs. Many essential aspects of state estimation, enabling autonomous flights with a MAV were covered. First, the different models used for the state estimation in MAVs were compared. The IMU model was identified as a better option for the state estimator. The IMU measurements received at rates of 1kHz were used for the prediction process of the state estimator and form an integral part of the state covariance matrix, noise covariance matrix calculations; hence, implicitly impacting the performance of the correction process of the state estimator as well.

Thus, there is a necessity to secure the IMU from sensor attacks. The IMU measurements form the core of the state estimation process and must be safeguarded against any malicious intents. In the next chapter, a method of integrating a resilient sensor fusion technique while using redundant sensors to counteract the effect of attacks on the IMU is proposed. The background review on the state estimation process is vital to appreciate the need for protecting the IMU in a MAV.

# 4. Resilient State Estimation for MAVs

Most cyber-physical systems and specifically MAVs are systems that continuously interact with dynamic and stochastic environments. Sensors are fundamental to transforming physical signals into logical signals, thereby providing an irreplaceable bridge between the real world and control systems. Sensors are undeniably one of the most important parts of automated control systems like MAVs. In this chapter, the problem of state estimation in the presence of compromised sensors is confronted.

## 4.1. Related Work

This section examines the current state of the art in resilient state estimation and secure control techniques against faults, failures, and attacks. It is a result of an exhaustive search of recent publications on our topics of interest.

First, the latest research on attack models and attack techniques on cyber physical systems and control systems is reviewed in this chapter. In [4], Teixeira et al. present an adversarial model for networked control system architecture. Using a typical control architecture for a networked control system, they analyze and present attack models for replay attacks, bias injection attacks, and zero dynamics. This work is useful for modeling attack spaces and threat vectors. In [27], Smith discussed a method to covertly modify sensor and actuator signals. He suggested that it can be accomplished by intruding the network or through physically modifying the sensors. He also proposes a parametrized feedback structure to gain control of a Linear time-invariant plant. The controller assumes that any disturbance or fault entering the system is comparable to an uncompromised setting. A malicious agent can construct such an attack using just the design of the plant;

knowledge of the controller is not needed. The above work advocates the case for safeguarding the sensors and the state estimation process.

There have been numerous investigations into various approaches to secure general cyber-physical systems, based on different assumptions and conditions. Previous work done in the field of robust control [28] is not extendable to our problem of security since attacks cannot be modeled as bounded disturbances and can be random in nature.

In [29], LeBlanc, Zhang, Koutsoukos, Sundaram attempt to tackle the problem of intra-network consensus in the presence of faulty nodes. The strategy is based on local information and is resilient to breaches, with an assumption that the compromised nodes have knowledge of the other nodes on the network. The authors provide necessary and sufficient conditions for the functional nodes to reach consensus in the presence of adversarial nodes. They argue that connectivity is not adequate and instead suggest a unique graph property called network robustness. Mitra and Sundaram [30], describe the problem of distributed state estimation in Linear time-invariant systems using a network of sensors, wherein some sensors are attacked to report faulty measurements. They developed a secure estimation strategy given a bounded Byzantine attack model, where compromised nodes have knowledge of the system dynamics and deviate from normal operation. They present sufficient guarantees for their estimation strategy. The relationship between the dynamics of the system, the graph structure, and the measurement structure of the nodes is discussed. Zhang and Sundaram [31] developed a resilient median-based consensus algorithm in the presence of faulty nodes. They claim that the consensus algorithm is computationally lightweight and is efficacious for multiple fault models. In [32], Sundaram and Hadjicostis present a distributed consensus algorithm that enables the system to

calculate an arbitrary state of the system in the presence of malicious/faulty nodes. They also calculate the maximum number of malicious nodes that the consensus algorithm can tolerate.

Chabukswar, Mo, and Sinopoli [33], present a model based technique for detecting integrity attacks on sensors of a control system. They discuss the effect of an attack on a control system in steady state by replaying the sensor information. The paper then suggests a control algorithm that addresses this vulnerability by augmenting the controller using a zero-mean Gaussian authentication signal to the Linear Quadratic Gaussian optimal control. They also show that the authentication signal helps in detecting the replay attack with a side-effect of degraded control performance. They also structure a method to design the covariance of the authentication signal to reduce performance loss with detection guarantees. In [34], Mo, Weerakkody, Sinopoli suggest a method called physical watermarking as a method to ensure the correct operation of a control system. They identify that cryptography tools are ineffective against physical attacks on the system or internal or through malicious insiders. They suggest physical watermarking to physically authenticate cyber-physical components; given a noisy input sensor reading, the effect of the noisy measurement can be observed in the true output based on the system dynamics. An attacker cannot emulate the watermark because he cannot construct the output measurement that should correspond to the faulty sensor measurement.

In [35], Pasqualetti et al. present a framework monitor to detect faults in linear systems and specifically in a power network caused by an adversary. They provide algorithms to design fault-monitoring filters for intruder detection. However, the number of monitor filters needed is combinatorially dependent on the number of sensors, leading

to concerns of scalability. Numerous researchers have attempted to model the attack and defense model within a game theory framework [36], [37], [38]. The attacker (modeled to maximize said cost) and the controller (modeled to minimize the same) are modeled as competitors in a game.

In [39], Fawzi, Tabuada, and Diggavi present a secure state estimation method for arbitrary attacks on sensors and actuators with the assumption that the profile of the attacked sensors does not change. They show that one can design an output-feedback stabilization law using a state estimator resilient to attacks with a standard feedback law. In [40], Hu et al. present a secure state estimator that protects UAV against arbitrary and unbounded attacks, where the attacked sensors may change over time. The authors couple their secure state estimator with a standard Kalman filter (as a pre-filter) and identify better results. Shoukry et al. [41] present a secure state estimator using a satisfiability modulo theory approach. They leverage formal methods over real numbers to identify a secure state estimator that is sound, complete, and computationally efficient. They also present an upper bound on the runtime performance. Their analysis is made without assumptions about the attack model, on a multi-dimensional system with multiple sensors.

Other attempts to solve the combinatorial state estimation problem have been done using brute force or convex relaxations. Chong et al. [42] present two different algorithms for secure state estimation. The algorithm uses the observability gramian and a consistency condition to select the correct estimate amongst multiple state estimates. They also present an observer that asymptotically converges to the right estimate based on the values of its past inputs and outputs. These approaches can be identified as brute force search techniques. In [43], Pajic et al. present a resilient state estimator that can be used in systems

with modeling errors. They also present a bound on the state estimation error. The authors model the state estimation problem as a $l_0$minimization problem which is then relaxed into a convex $l_1/l_r$ problem which can be solved in polynomial time. However, this relaxation may lead to incorrect estimates.

The authors of [5], building on the work of [43], [44], and [45] present some of the novel work in the field of attack resilient cyber physical systems with a focus on attack resilient state estimators. [46], [47], [48], [49], and [50] present some more work on resilient state estimators for different attack models and assumptions. Pajic, Weimer, Bezzo, Sokolosky, Pappas, and Lee [5] describe their work on the development of an attack resilient cyber physical system and conclude with a case study on the cruise controller for an Autonomous Ground Vehicle. Following the work in [43], they address attack-resilient state estimation and provide respective robustness guarantees. They conclude that the maximal performance loss by a smart malicious agent exploiting the difference between the physical dynamic model of the system and the model used for state estimation is bounded and linear. The article also presents a technique to map latency, jitter, and synchronization faults to parameters of the state estimator. Thus, one may map control performance to real time specifications. Lastly, they discuss a technique to construct an assurance case which includes the model of the state estimator and the physical environment, along with the software structure of the controller. The modeling, robustness guarantees, and assumptions made throughout the study for the system of interest (model used for the study) are like a lot of other CPS control and estimation problems.

## 4.2. Resilient Sensor Fusion

It is important to consider using redundant sensors to prevent a system breakdown due to sensor failure. Nonetheless, adding redundant sensors poses a new problem of fusing sensor measurements with varying and unpredictable error profiles. The system must identify faulty or error infused sensor readings from correct sensor readings.

Sensor fusion discusses methods to combine data from independent sensors into one sensor reading while ensuring robustness, precision, accuracy, and reliability. Distributed sensing enables identifying the number of node failures the system can handle while maintaining correctness and reliability. This section examines and structures the problem of attack resilient sensor fusion for resilient state estimation in the presence of adversaries. Each sensor measurement is structured as an interval and the width of the interval is based on the noise profile of the sensor; reflecting the underlying accuracy of a sensor. The sensor fusion technique makes no assumptions about the system dynamics. Hence, a similar approach can be used for other comparable systems to structure resilient state estimators.

The work integrates Brooks – Iyengar fusion, which outputs a fusion interval for a presumed set of compromised or spoofed sensors with the generic state estimator (Chapter 3), to construct a resilient state estimator. A compromised/attacked sensor is any sensor that is under the effect of an adversary is defined.

Sensor fusion is discussed in detail to highlight how a control system can be certain to make correct decisions in presence of compromised nodes or nodes under adversarial attacks. A summary of the Byzantine generals problem identified by Lamport et al. [51] is presented followed by a discussion on Brooks-Iyengar Fusion as the underlying sensor

fusion architecture to address the Byzantine generals problem. Other sensor fusion algorithms that improve the precision and accuracy of the measurements taken by multiple sensor networks include Approximate Consensus [52], In-exact Consensus [53], Byzantine Vector Consensus [54], and Multidimensional Agreement [55]. But, as concluded in [56], refer to Table 3, Brooks-Iyengar Fusion is a superior algorithm when compared in terms of maximum faulty nodes, convergence rate, accuracy, and precision of each round; against the other algorithms.

| **Algorithm** | **Approximate agreement** | **FCA** | **Approximate BVC** | **Marzullo sensor fusion** | **Brooks-Iyengar algorithm** |
|---|---|---|---|---|---|
| Input | scalar | scalar | vector | interval | interval/hybrid |
| Faulty PEs tolerated | $< N/2$ | $< N/3$ | $\leq (N-1)/(d+2)$ | $< N/3$ | $< N/3$ |
| Maximum faulty PEs | $< N/3$ | $< 2N/3$ | $\leq (N-1)/(d+2)$ | $< N/2$ | $< N/2$ |
| Convergence rate [21] | $1/(1+[N-2\tau-1])$ | $2\tau/N$ | $(1-\gamma)$ | $2 * accuracy$ | $2\tau/N$ |
| Accuracy | $\delta(U)$ | $\kappa + \delta\tau/N$ | in the convex hull | $[a_{\omega=N-\tau}, b_{\omega=N-\tau}]$ | $[a_{\omega=N-\tau}, b_{\omega=N-\tau}]$ |
| Precision of each round | $\delta(U)/2$ | $2\tau\delta/N$ | $(1-\gamma)(\Omega_l[t-1]-\mu_l[t-1])$ | $\lvert b_{\omega=N-2\tau} - a_{\omega=N-2\tau}\rvert$ | $\lvert b_{\omega=N-2\tau} - a_{\omega=N-2\tau}\rvert/(1+\alpha)$ |

Table 3: Comparison of different sensor fusion techniques [56]

The variables used in Table 3 are summarized below:

1. $N$ is the number of sensors

2. $\tau$ is the number of faulty sensors
3. $\delta(U) = \max(U) - \min(U)$
4. $\kappa$ refers to the accuracy, i.e., difference between the sensor's observed value and the ground truth, i.e., $\max\left|v_p - \hat{v}\right| \leq \kappa$
5. $d$ refers to the number of dimensions for BVC
6. $\gamma = \frac{1}{N(\frac{N}{(N-\tau)})}$
7. $\Omega_l = max_{1 \leq k < m} v_{kl}[t]$ in $m$ non-faulty sensors, where $v_{kl}[t]$ is the $l-th$ entry of the vector of the $k-th$ sensor in the $t-th$ round.
8. $\mu_l = min_{1 \leq k < m} v_{kl}[t]$ in $m$ non-faulty sensors, where $v_{kl}[t]$ is the $l-th$ entry of the vector of the $k-th$ sensor in the $t-th$ round.
9. $b_w = most\ right\ end\ point\ of\ overlapping\ intervals\ with\ weights \leq w$
10. $a_w = most\ left\ end\ point\ of\ overlapping\ intervals\ with\ weights \leq w$
11. $\alpha = \frac{N-\tau}{(2N-\tau)\tau}$

## 4.3. Byzantine Generals Problem

The Byzantine generals problem is a decision-making problem that was formalized by Lamport et al. [51], in which a commander-in-chief has multiple regiments of the Byzantine army camped outside an enemy city. Each regiment is commanded by a general and the generals can communicate with the generals only through a messenger. The generals must observe the enemy's actions and deduce a common plan of action. Though, some of the generals may be traitors and may attempt to obstruct the loyal generals from coming to a consensus. All the generals and their corresponding regiments must follow the

order of a reliable commander-in-chief for success. The problem demands an algorithm such that:

1. All generals decide on the same plan of action.
2. A small number of traitors should not direct the generals towards adopting a wrong plan.
3. One can easily draw parallels between the generals problem and our problem of sensor consensus/fusion by making necessary analogies. It can be restated as a system of N nodes, amongst which p may be faulty. The algorithm must ensure that:
4. The non-faulty nodes must come to a consensus about the data received from another node Z.
5. If Z is non-faulty, the consensus must match the message received from Z.

## 4.4. Brooks-Iyengar Sensor Fusion

A formalization of the Brooks-Iyengar Sensor Fusion is established in this section. Brooks-Iyengar Sensor Fusion merges the Fast Convergence Algorithm (FCA) presented by Mahaney and Schneider [53] with the optimal region algorithm to produce the Brooks Iyengar Algorithm which has superior accuracy and precision for distributed decision making. The abovementioned optimal region algorithm is based on Brooks and Iyengar's Multidimensional Agreement algorithm [57] and is comparable to Marzullo's fusion algorithm [58].

Given a system with $n$ sensors and $\zeta$ faulty sensors. Each sensor presents its measurement as an interval constructed using its precision. The output of the algorithm includes a point estimate of the measurement and a corresponding interval.

The algorithm follows the work of [59], [56]:

**Algorithm 1: Brooks-Iyengar Sensor Fusion Algorithm**

Input:

The measurements sent by sensor $k$ where $1 \leq k \leq N$, and the interval received from sensor $k$ can be denoted as $[l_k, h_k]$. Let $\zeta$ be the number of faulty sensors.

Output:

As suggested earlier, the output is a point estimate and an interval of the measurement

1. State estimator receives point estimate and corresponding interval from all the sensors.
2. Take the union of the intervals of the collected measurements.
3. Divide the union into mutually exclusive intervals based on the number of measurement intervals that intersect. We call the number of intersected intervals the weight for that interval.
4. Identify intervals with weight less than $N - \zeta$. Let $N - \zeta$ be denoted as $F$.
5. The set of remaining intervals $S = \{(I_1, w_1) \dots (I_F, w_F)\}$ where $I_i \; and \; w_i$ refer to the interval and weight for the $i^{th}$ sensor respectively
6. The point estimate can be calculated as:

$$p = \frac{\sum_F \frac{(l_i + h_i) \times w_i}{2}}{\sum_F w_i}$$

7. The interval estimate is evaluated as $(l_{I_1}, h_{I_L})$

According to [56], the accuracy of the sensor fusion measurement can be calculated as:

$$|p_i - \acute{p}| \leq \left|b_{w=N-\zeta} - a_{w=N-\zeta}\right| \leq \min_{\zeta+1}\{|u|: u \epsilon U\}$$

$where\ p_i\ is\ point\ estimate\ for\ sensor\ i, |u|\ denotes\ the\ length\ of\ u$

$and\ b_w = most\ right\ end\ point\ of\ overlapping\ intervals\ with\ weights \leq w$

$and\ a_w = most\ left\ end\ point\ of\ overlapping\ intervals\ with\ weights \leq w$

Also, the precision of the sensor fusion measurement can be calculated as in [56]:

$$\frac{1}{1+\alpha}\left|b_{w=N-\zeta} - a_{w=N-\zeta}\right|$$

$$where\ \alpha = \frac{N-\zeta}{(2N-\zeta)\zeta}$$

## 4.5. System Description

Our system has $n$ sensors measuring some physical variables and communicating with the flight controller over a shared bus. The sensors deliver the measurement in the form of intervals. The controller then calculates an interval containing all possible values of the true state based on the given precision. Given a precision guarantee of Δ, an interval sized 2Δ is constructed around the sensor measurement. The interval size may be expanded based on implementation restraints and delay or jitter sensitivities. The controller on receiving n intervals, fuses the measurements assuming $f$ faulty sensors. The fusion operates conservatively. A sensor is correct if its interval contains the true state value.

### 4.5.1. Problem Statement

The problem in this work is to neutralize an attack on the sensors. The attacker's policy to maximize the impact of the attack is first formalized. The goal is to reach the correct state estimate despite compromised sensors.

### 4.5.2. Attack Model

The fact that adversaries may control/spoof what sensor measurements are being passed to the controller is assumed. The attacker's goal is to maximize the fused interval and lead to an incorrect point measurement while remaining undetected. Thus, the attacker can spoof the sensors to maximize the width of the fusion interval and greatly disrupt the performance of an air vehicle. As discussed in section 2.3.2.4, there are multiple ways of spoofing sensors, either by modifying the software of the flight controller or through physical means.

Our attack model discusses attacks on the IMU of the MAV. As examined in chapter 3, since our state estimator uses the IMU readings instead of the MAV dynamic model to estimate the state, the system is heavily dependent on the IMU readings to ensure correct state estimation and consequently stable control.

The defense model is to use multiple IMUs to ensure no single point of failure. This is a viable solution, since IMUs are inexpensive, have low power requirements, and are light-weight. At the same time, one may also argue that compromising all the IMUs on a given system is not possible. Also, it may be possible that not all the components of the IMU are compromised and may partly be functional. The system has multiple sensors that measure the angular velocity and linear acceleration. Before the sensor readings are sent to the controller, the readings are passed through a pre-filter to assess for possible attacks or faults and discard the compromised measurements.

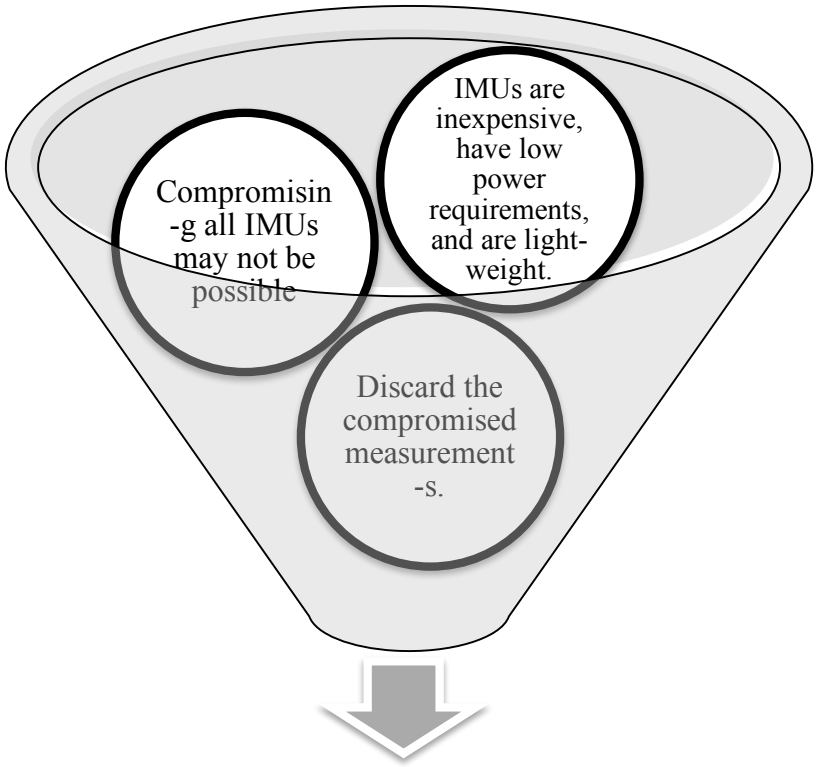


Figure 12: Defense strategy against sensor spoofing attacks

### 4.5.3. Fusion Methodology

Brooks and Iyengar proposed an interval-based resilient sensor fusion algorithm, wherein the accuracy of the fused sensor reading is better than the individual sensor readings. The fusion algorithm provides a point estimate as well as an interval around the point estimate. The interval size is bounded if the number of faulty sensors are bounded. Brooks Iyengar fusion assumes an upper bound on the number of faulty measurements or sensors, i.e., $n/3$.

The fusion algorithm outputs a fusion interval that is guaranteed to contain the true value. As discussed in 5.2. the fusion algorithm identifies the intersecting intervals and the corresponding weights. The fused estimate and interval is calculated by taking a weighted average of the intervals.

1. If $f < n/3$, then the fusion interval is bounded by a correct interval.
2. If $f < n/2$, then the fusion interval is bounded by an interval which might not be correct.
3. If $f \geq n/2$, then the fusion interval is not bounded and may not contain the correct measurement; where $f$ is the number of attacked sensors.
4. For our analysis, **we assume $\boldsymbol{f < n/3}$**.

#### 4.5.4. Attack Detection:

The fusion algorithm as discussed in section 4.5.3. detects an attack by checking if an interval intersects the calculated fusion interval (from the un-attacked sensors). Intervals not intersecting the fusion interval correspond to the attacked sensor since they cannot contain the true value. A possible criticism of this sensor algorithm is that it handles sensor faults and sensor attacks equivalently.

### 4.6. Secure State Estimator Structure

A state estimator that uses Brook Iyengar Fusion to fuse the sensor readings from IMUs prior to feeding the angular velocity and linear acceleration measurements into the EKF based state estimator as discussed in chapter 4 is proposed. A model of our state estimator is described in Figure 9.

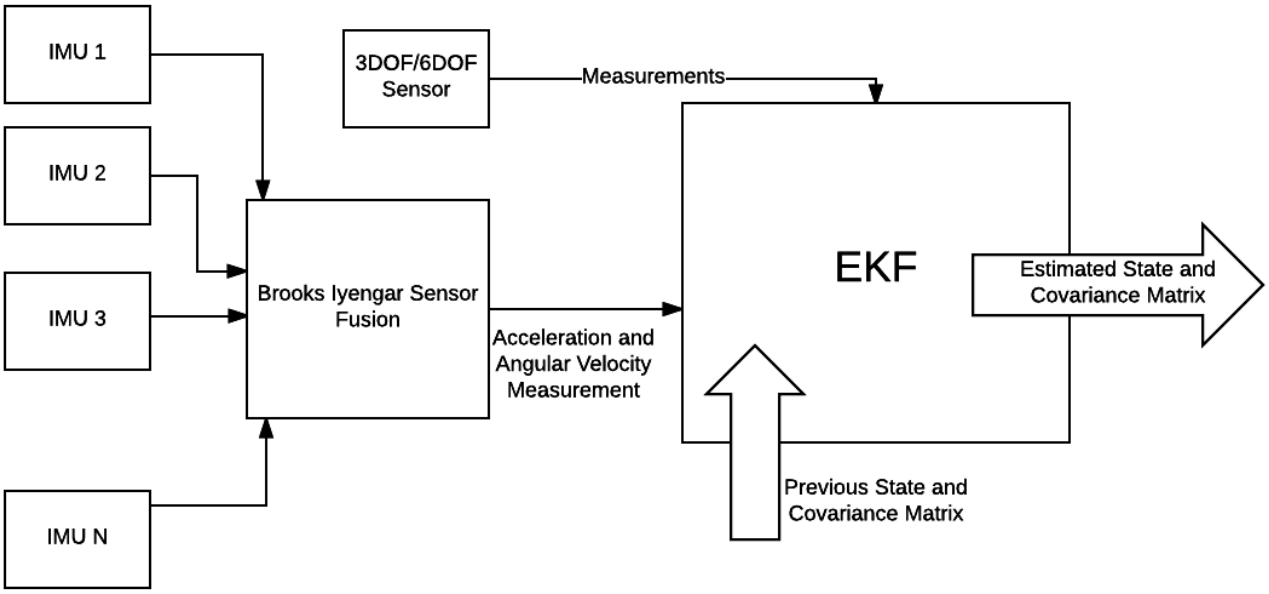


Figure 13: Secure State Estimator

## 4.7. Summary

In this chapter, a state of the art state estimation framework with an attack resilient sensor fusion technique to defend MAVs against sensor attacks was integrated. As highlighted in chapter 3, the IMU forms the core of the state estimation process. Since, IMUs are lightweight and have low energy requirements, we advocate using redundant IMUs to prevent a single point of failure and a compromised system. Consequently, we identify Brooks Iyengar Sensor Fusion as an attack resilient sensor fusion technique. The improvements in the state estimation resilience is exemplified through the simulations in Chapter 5. The experiments show that the resilient state estimator can correctly estimate the state of the system in the presence of sensor attacks.

# 5. Results and Analysis

The results of the software in the loop simulation are discussed in this section. Following which is an analysis to verify the proposed state estimator's performance and inferences that may be drawn from the results. The simulator was programmed in MATLAB (MATLABR_2015A) on a standard MacOSX personal computer. The simulations used the EuRoC MAV dataset [60], [61]. Available data includes [61]:

- Stereo Images (Aptina MT9V034 global shutter, WVGA monochrome, 2×20 FPS)
- MEMS IMU (ADIS16448, angular rate and acceleration, 200 Hz)
- VICON motion capture system (6D pose)
- Leica MS50 laser tracker (3D position)

The dataset represents the sensor measurements of a MAV following programmed waypoints. The simulation represents a MAV installed with multiple IMUs, and is also receiving position measurements from a Leica Total Station. The outputs of these sensors are fused together to generate an estimate for the position, velocity, and attitude of the system. One or some of these IMU measurements are attacked. A uniformly distributed random attack is simulated. The comparative performance of the state estimators in different conditions is presented. Please note that the simulations of the state estimator do not include of the control dynamics and response of the system. While examining the performance of the generic state estimator in the presence of adversarial attacks, it is vital to respect that the deference of the estimated state from the ground truth will be amplified by the response of the flight controller. The flight controller which was programmed to follow a certain trajectory or to reach certain way points will make position control and

attitude control calculations based on the incorrect state estimate. This will lead to incorrect actuator controls and lead the quadrotor to incorrect positions and lead to a greater state estimate error. This effect will be propagated and lead to a severely erroneous system performance and unpredictable behavior. Incorrect control signals in unknown states may possibly lead to the system crashing due to compounding error factors. For all the simulations, the blue trajectory is the estimated position of the MAV state estimator, and the red trajectory is the ground truth trajectory obtained from the VICON motion capture system. Broadly, the simulations in this section describe:

### 5.1. Generic state estimator performance without compromised sensors.

This simulation (Figure 14) describes the performance of the generic state estimator when the sensors are operating correctly. The red trajectory represents the ground truth coordinates obtained from the VICON tracker. The blue trajectory is the estimated position of the MAV state estimator. The axes are indicative of the position along the X, Y, Z directions.

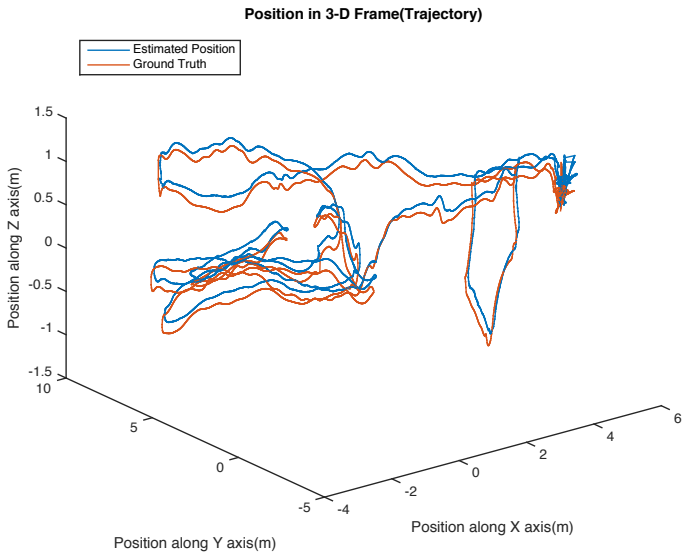


Figure 14: Trajectory of a Quadrotor in nominal conditions while using the standard state estimator.

Figure 15, Figure 16, Figure 17 represent the estimated X, Y, and Z positions of the MAV (blue trajectory) along with the ground truth X, Y, and Z positions of the MAV (red trajectory). The state estimator correctly tracks the position of the MAV.

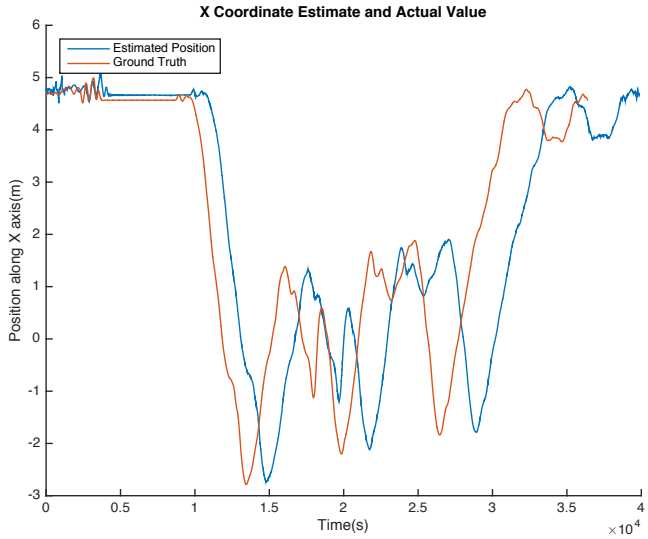


Figure 15: Position of Quadrotor along X-axis in nominal conditions while using the standard state estimator

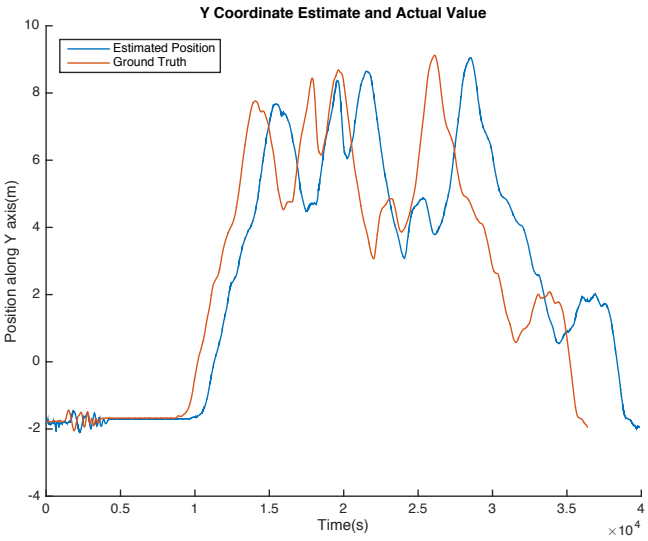


Figure 16: Position of Quadrotor along Y-axis in nominal conditions while using the standard state estimator.

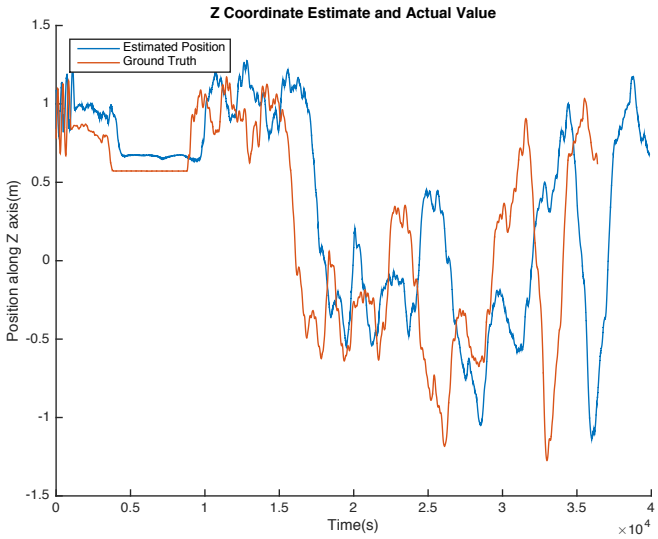


Figure 17: Position of Quadrotor along Z-axis in nominal conditions while using the standard state estimator.

## 5.2. Generic state estimator performance in the presence of sensor attacks.

This simulation (Figure 18) describes the performance of the generic state estimator when the IMU is under attack. The red trajectory represents the ground truth coordinates obtained from the VICON tracker. The blue trajectory is the estimated position of the MAV state estimator. The axes are indicative of the position along the X, Y, Z directions. The state estimator fails to estimate the position of the MAV.

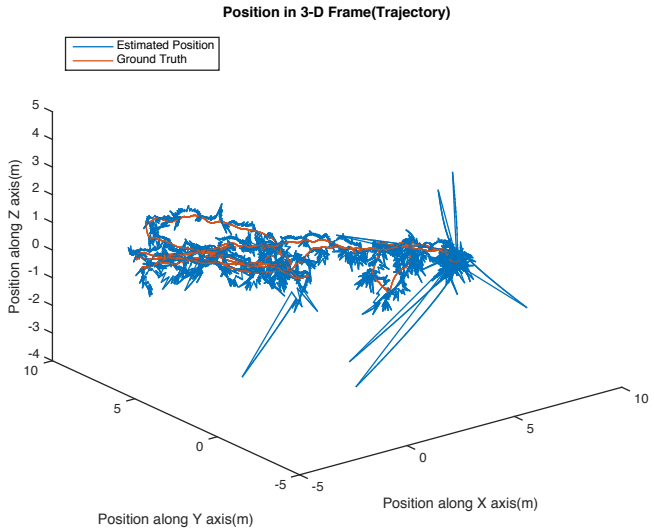


Figure 18: Trajectory of a Quadrotor under adversarial attacks while using the standard state estimator

Figure 19, Figure 20, Figure 21 represent the estimated X, Y, and Z positions of the MAV (blue trajectory) along with the ground truth X, Y, and Z positions of the MAV (red trajectory).

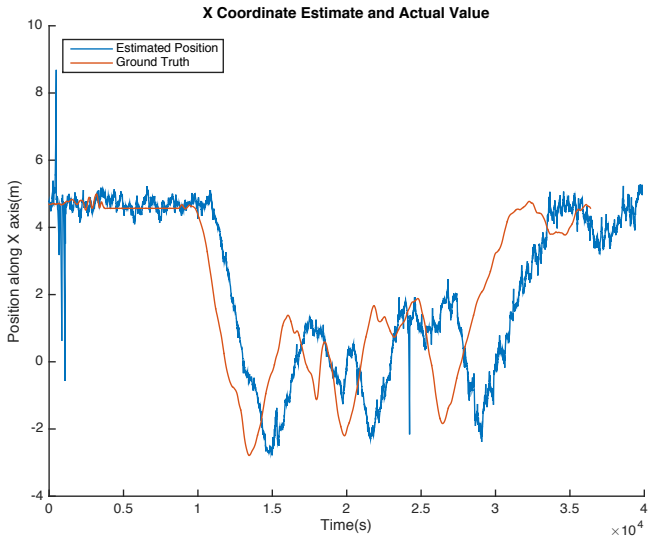


Figure 19: Position of Quadrotor along X-axis under adversarial attacks while using the standard state estimator

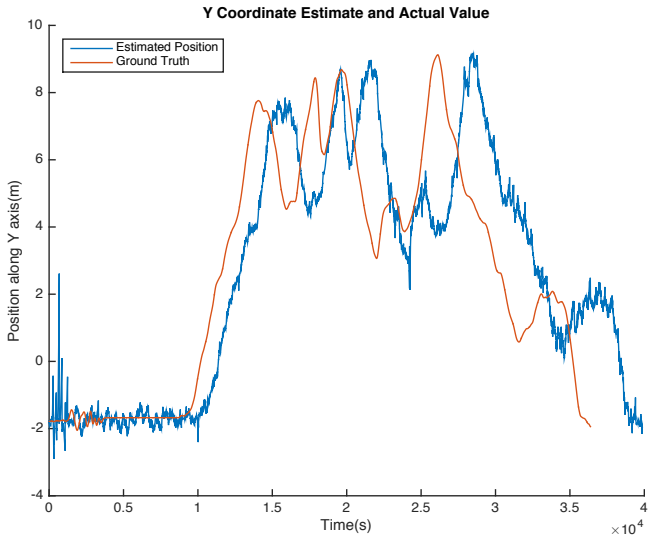


Figure 20: Position of Quadrotor along Y-axis under adversarial attacks while using the standard state estimator.

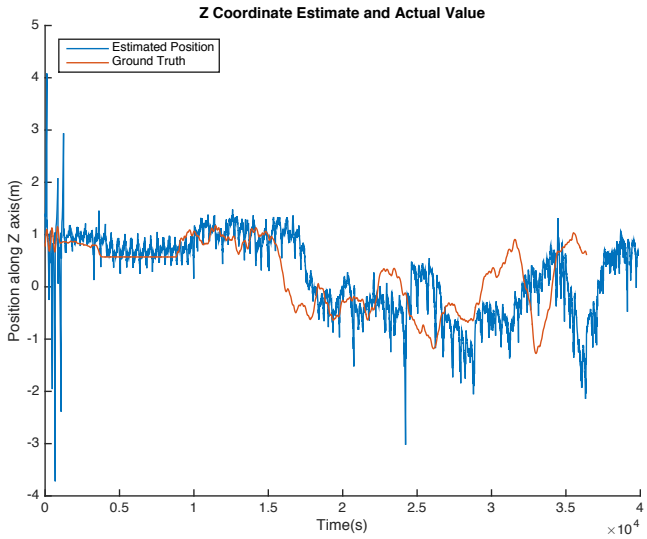


Figure 21: Position of Quadrotor along Z-axis under adversarial attacks while using the standard state estimator.

## 5.3. Secure state estimator performance in the presence of sensor attacks.

This simulation (Figure 22) describes the performance of the resilient state estimator when the sensors are operating correctly. The red trajectory represents the ground truth coordinates obtained from the VICON tracker. The blue trajectory is the estimated position of the MAV state estimator. The axes are indicative of the position along the X, Y, Z directions.

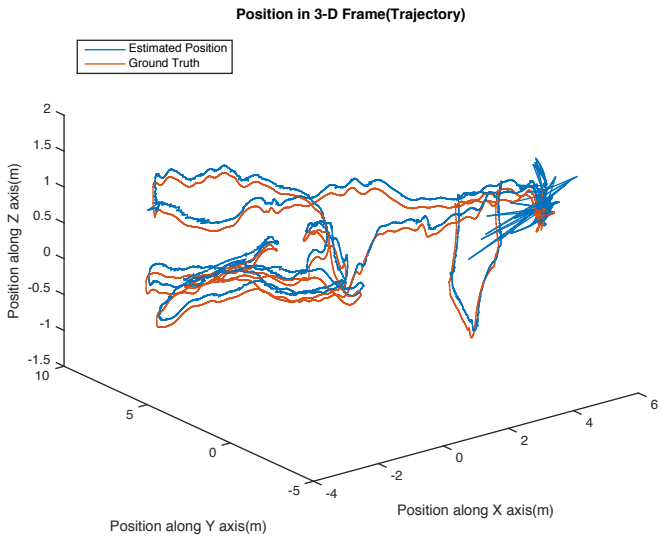


Figure 22: Trajectory of a Quadrotor under adversarial attacks while using the secure state estimator.

Figure 23, Figure 24, Figure 25 represent the estimated X, Y, and Z positions of the MAV (blue trajectory) along with the ground truth X, Y, and Z positions of the MAV (red trajectory). We observe an improved performance of the state estimation process with the integrated resilient sensor fusion module. The resilient state estimator correctly estimates the state of the MAV.

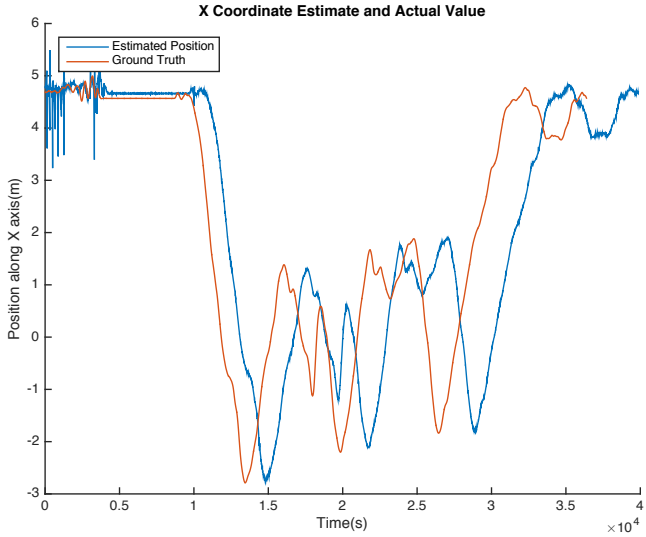


Figure 23: Position of Quadrotor along X-axis under adversarial attacks while using the secure state estimator.

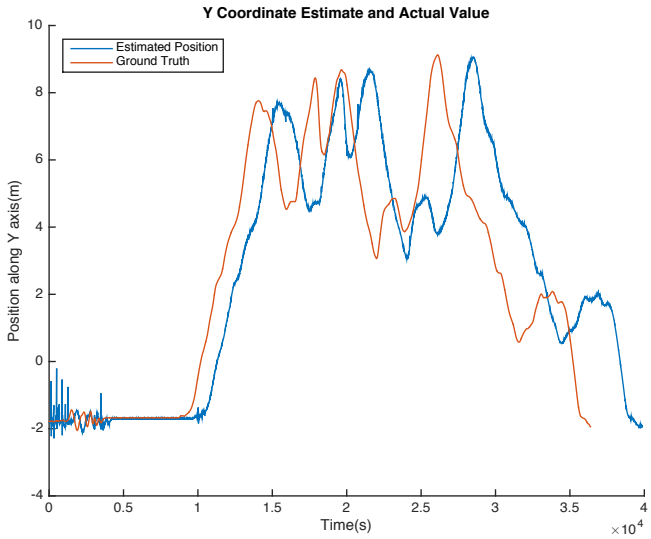


Figure 24: Position of Quadrotor along Y-axis under adversarial attacks while using the secure state estimator.

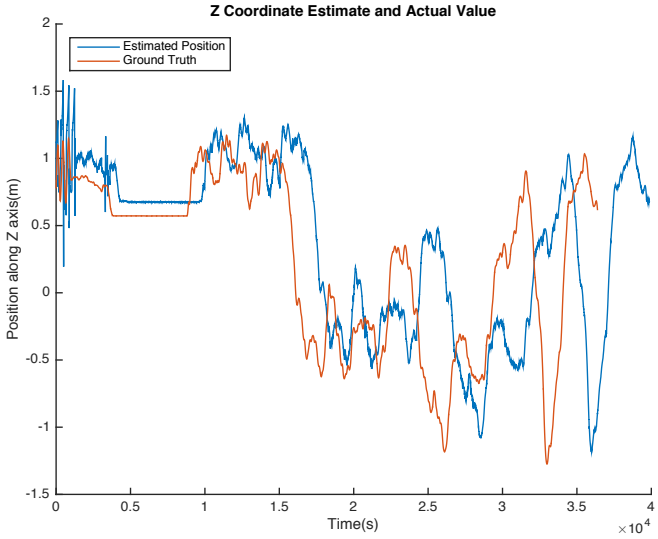


Figure 25: Position of Quadrotor along Z-axis under adversarial attacks while using the secure state estimator.

## 5.4. Quantitative Comparison

We evaluate the performance of the state estimators based on the Hausdorff distance [62]. The Hausdorff distance is a metric that compares the distance between two sets of spaces. It is used as a measure of trajectory similarity. The results of the Hausdorff distance calculations are mentioned in Table 4. A smaller value indicates a high similarity between

two trajectories. In the simulations, the ground truth trajectory and the estimated trajectories are compared.

| Simulation | Haursdroff Distance |
|---|---|
| Generic State Estimator without Attack | 0.3717 |
| Generic State Estimator with sensors under attack | **5.9983** |
| Resilient State Estimator with sensors under attack | 0.4819 |

Table 4: Quantitative comparison of State Estimator Performance

## 5.5. Summary

This chapter presents a proof of concept for the performance of the integrated resilient state estimator for MAVs in the presence of attacks on the IMU. The generic state estimator fails to correctly estimate the state of the MAV in the presence of attacks on the MAV (Figures 18 – 21). However, as observed in Figures 22 – 25, the resilient state estimator could successfully estimate the position of the MAV. A quantitative comparison between the state estimators is also presented in the form of the Hausdorff distance metric. In the next chapter, we conclude the thesis and set directions for the future work.

# 6. Conclusion and Future Work

The thesis investigated and presented a solution to the resilient state estimation problem. We integrated a state of the art state estimator with a resilient sensor fusion technique, and could reconstruct the state of the MAV in the presence of adversarial attacks. This is vital for existing commercial MAVs, for future drone delivery systems, and for other applications of MAVs. The proposed resilient state estimator makes no assumptions about the attack signal model.

Based on the formalization of Brooks Iyengar Fusion, one can fuse the measurements from the attacked sensors to give an accurate fused point estimate and an interval. Brooks Iyengar fusion is combined and used as a pre-filter to the EKF based state estimator for fusing multiple sensor measurements of the same physical signal. The performance of the generic state estimator and the proposed resilient state estimator is compared after introducing an attack on a sensor, i.e., modifying sensor readings. Specifically, simulations to show the performance of secure state estimators in navigation of a quadrotor under sensor attacks on the IMU, refer to Figure 10 – Figure 21. The number of attacks that can be tolerated, i.e., the number of sensors that can be attacked is a maximum of $n/3$ sensors while maintaining safe operation.

## 6.1. Contributions

The research goals were identified in the beginning of the thesis and have been addressed. The security vulnerabilities of MAVs, i.e., the different threat vectors were identified in chapter 2 (RG-1). In chapter 3, the current state of the art state estimation technique used in MAVs (RG - 2) was summarized to study the impact of sensor attacks on a generic state

estimator. Propositions to improve the existing state estimator were stated in chapter 4, where Brooks-Iyengar sensor fusion was identified as a viable solution (RG - 2 & RG - 3). Chapter 5 offered a proof-of-concept for our resilient state estimator (RG - 4) through a case study.

The primary contributions of this thesis are:

1. By understanding the state estimation process of MAVs, we identify the importance of the IMU as the most vulnerable attack surface for a MAV.
2. Having identified the IMU as the most vulnerable component of the MAV state estimation process, we develop a resilient state estimator for the MAV using existing results in resilient fusion. An important feature of the fusion algorithm is that it makes no assumptions about the system dynamics.
3. The resilient state estimator was validated by implementing it on a real dataset containing IMU, Leica, VICON data obtained from a MAV. The performance of the resilient state estimator is exemplified in chapter 5.

As a final comment, it is important to admit that much of the motivation and enthusiasm for this work is derived from the foresight of the future of MAVs. In the end, it is this vision that provided the guiding framework. However, it is equally critical to realize that many of the results and techniques developed in this thesis are not limited to MAVs. For example, a similar pre-filter sensor fusion approach is likely to be relevant for other autonomous vehicles – for autonomous cars, such redundancy could be useful for wheel encoders measuring the velocity of the car. Thus, even though MAVs motivated this thesis, its impact is likely to transcend to other robotic applications.

## 6.2. Future Work

Future directions for the work are to construct a resilient state estimator by adding redundancy in measured variables measured by different types of sensors. For example, if one wants to secure the velocity of a given MAV. Instead of using velocity measurements from redundant sensors of a single type that measure velocity like a GPS, the estimator can be modified to fuse measured velocity from different sensors like a GPS, IMU, Camera, LIDAR, etc. Redundancy through a broader range of sensors will make the system even more robust. One can also improve the attack detection procedure by leveraging the system's dynamical model.

Moreover, a major underlying assumption is that uncompromised sensors provide error-free measurements. An extension of this work would be to introduce random faults in the sensor measurements and study the system performance in more generalized conditions.

Another possible extension of this work is to incorporate sensor measurement history to better tune the state estimator to ignore faulty measurements. One may utilize past measurements in conjunction with a dynamical model of the system to reduce the size of the convex hull of the measurement intervals of the sensor readings and improve the precision of the fusion algorithm.

Finally, to better study the performance of the resilient state estimator, the next step is to implement a controller for the MAV and observe the response of the MAV when it uses the resilient state estimator.

# Appendix A

Appendix A presents the MATLAB code and helper functions used to simulate the state estimator.

## 1. Main Function

```
% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Simulator - Main Function
%
% Description:This function is the main script to simulate the state
% estimator.
%
% {trawny2005indirect,
%   title={Indirect Kalman filter for 3D attitude estimation},
%   author={Trawny, Nikolas and Roumeliotis, Stergios I},
%   journal={University of Minnesota, Dept. of Comp. Sci. \& Eng., Tech. Rep},
%   volume={2},
%   year={2005}
% }
%--------------------------Clearing Values-------------------------------
clc;
clear;
%----------------------Defining Filter Parameters-------------------------
n=20; %number of states
currTime=0;
%-----------------------------Noise values-------------------------------
```

```
r=0.1;    %std of measurement
errZ=0.01*eye(3); %Leica Measurement Noise Covariance Matrix
%Process Noise Covariance Matrix
varNw=0.00135;varMatNw=varNw^2*eye(3);
varBw=0.00011;varMatBw=varBw^2*eye(3);
varNa=(0.023*1e-6);varMatNa=varNa^2*eye(3);
varBa=(0.011/60);varMatBa=varBa^2*eye(3);
zer=zeros(3);
Qc=[varMatNa zer zer zer;
   zer varMatBa zer zer;
   zer zer varMatNw zer;
   zer zer zer varMatBw];
%----------------------------Alotting Space-----------------------------
%initial estimate
x=zeros(n,1);
x(17,1)=1;
x(7:10)=.1;
P = eye(n-1);                    % initial state covraiance
% total dynamic steps
%-----------------------Feed Sensor Data-----------------------------
fileID = fopen('imu.csv');
csvRawData = textscan(fileID, '%u64,%f,%f,%f,%f,%f,%f', 'headerLines', 1);
imudata1.t = csvRawData{1}';
imudata1.omega = [csvRawData{2}, csvRawData{3}, csvRawData{4}]';
imudata1.a = [csvRawData{5}, csvRawData{6}, csvRawData{7}]';
fileID = fopen('imu.csv');
csvRawData = textscan(fileID, '%u64,%f,%f,%f,%f,%f,%f', 'headerLines', 1);
imudata2.t = csvRawData{1}';
imudata2.omega = [csvRawData{2}, csvRawData{3}, csvRawData{4}]';
imudata2.a = [csvRawData{5}, csvRawData{6}, csvRawData{7}]';
```

```
 imudata2.a=imudata2.a+10;
  imudata2.omega=imudata2.omega;
fileID = fopen('imu.csv');
csvRawData = textscan(fileID, '%u64,%f,%f,%f,%f,%f,%f', 'headerLines', 1);
%Attacked IMU

imudata3.t = csvRawData{1}';
imudata3.omega = [csvRawData{2}, csvRawData{3}, csvRawData{4}]';
imudata3.a = [csvRawData{5}, csvRawData{6}, csvRawData{7}]';
imudata3.a= imudata3.a+varNa*randn(3,36820);
imudata4.t = csvRawData{1}';
imudata4.omega = [csvRawData{2}, csvRawData{3}, csvRawData{4}]';
imudata4.a = [csvRawData{5}, csvRawData{6}, csvRawData{7}]';
imudata4.a= imudata4.a+varNa*randn(3,36820);
fileID = fopen('leica.csv');
csvRawData = textscan(fileID, '%u64,%f,%f,%f', 'headerLines', 1);
leicadata.t = csvRawData{1}';
leicadata.pos = [csvRawData{2}, csvRawData{3}, csvRawData{4}]';
fileID = fopen('ground.csv');
csvRawData = textscan(fileID,
'%u64,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f', 'headerLines', 1);
grounddata.t = csvRawData{1}';
grounddata.pos = [csvRawData{2}, csvRawData{3}, csvRawData{4}]';
grounddata.vel=[csvRawData{5}, csvRawData{6}, csvRawData{7}]';
grounddata.quat=[csvRawData{8}, csvRawData{9},
csvRawData{10},csvRawData{11}]';
grounddata.biasw=[csvRawData{12}, csvRawData{13}, csvRawData{14}]';
grounddata.biasa=[csvRawData{15}, csvRawData{16}, csvRawData{17}]';
prevTime=min(imudata1.t(1),leicadata.t(1));
measItr=1;
xV = zeros(n,numel(imudata1.t));     %estimate
```

```
sV = zeros(n,numel(imudata1.t));          %actual
xV(:,1)=x;
k=2;
ctr=2;
%-----------------------Filter operation---------------------------------
while(k<numel(imudata1.t))
   if(imudata1.t(k)<leicadata.t(measItr))
      currTime=imudata1.t(k);
      g=[0;0;9.8];
      dt=double(currTime-prevTime)/1e9;
      sV(1:3,ctr)=leicadata.pos(1:3,measItr);
      w0=imudata2.omega(1:3,k-1);
      w1=imudata2.omega(1:3,k);
%       a11=marzulloFusion({[imudata1.a(1,k)-2.5,imudata1.a(1,k)+1],[imudata2.a(1,k)-
8,imudata2.a(1,k)+7],[imudata3.a(1,k)-2,imudata3.a(1,k)+1],[imudata4.a(1,k)-
1,imudata4.a(1,k)+2]},4,1);
%       a12=marzulloFusion({[imudata1.a(2,k)-2.5,imudata1.a(2,k)+1],[imudata2.a(2,k)-
8,imudata2.a(2,k)+7],[imudata3.a(2,k)-2,imudata3.a(2,k)+1],[imudata4.a(2,k)-
1,imudata4.a(2,k)+2]},4,1);
%       a13=marzulloFusion({[imudata1.a(3,k)-2.5,imudata1.a(3,k)+1],[imudata2.a(3,k)-
8,imudata2.a(3,k)+7],[imudata3.a(3,k)-2,imudata3.a(3,k)+1],[imudata4.a(3,k)-
1,imudata4.a(3,k)+2]},4,1);
%       a1=[a11;a12;a13];
%       a11=approxAgreement([imudata1.a(1,k);imudata2.a(1,k);imudata3.a(1,k);imudat
a4.a(1,k)],1);
%       a12=approxAgreement([imudata1.a(2,k);imudata2.a(2,k);imudata3.a(2,k);imudat
a4.a(2,k)],1);
%       a13=approxAgreement([imudata1.a(3,k);imudata2.a(3,k);imudata3.a(3,k);imudat
a4.a(3,k)],1);
%       a1=[a11;a12;a13];
```

```
% a1=(imudata1.a(1:3,k)+imudata2.a(1:3,k)+imudata3.a(1:3,k)+imudata4.a(1:3,k))/4;
a11=sensorFusion({[imudata1.a(1,k)-0.5,imudata1.a(1,k)+0.5],[imudata2.a(1,k)-
0.5,imudata2.a(1,k)+0.5],[imudata3.a(1,k)-0.5,imudata3.a(1,k)+0.5],[imudata4.a(1,k)-
0.5,imudata4.a(1,k)+0.5]},4,1);
    a12=sensorFusion({[imudata1.a(2,k)-0.5,imudata1.a(2,k)+0.5],[imudata2.a(2,k)-
0.5,imudata2.a(2,k)+0.5],[imudata3.a(2,k)-0.5,imudata3.a(2,k)+0.5],[imudata4.a(2,k)-
0.5,imudata4.a(2,k)+0.5]},4,1);
    a13=sensorFusion({[imudata1.a(3,k)-0.5,imudata1.a(3,k)+0.5],[imudata2.a(3,k)-
0.5,imudata2.a(3,k)+0.5],[imudata3.a(3,k)-0.5,imudata3.a(3,k)+0.5],[imudata4.a(3,k)-
0.5,imudata4.a(3,k)+0.5]},4,1);
    a1=[a11;a12;a13];
    pos_hat_avg_0=xV(1:3,ctr-1);
    vel_hat_avg_0=xV(4:6,ctr-1);
    q_hat_avg_0 = xV(7:10,ctr-1);
    bw=xV(11:13,ctr-1);
    ba = xV(14:16,ctr-1);
    lambda = xV(17,ctr-1);
    posgps = xV(18:20,ctr-1);
    C=qGetRotation(q_hat_avg_0);
    w_hat_0 = w0;
    %State Propogation
    % We instead proceed as follows:
    % 1. We propagate the bias (assuming the bias is constant over the integration
interval)
    ba1 = ba; % TODO : Here is a problem : we have no way of validating or updating
the bias. Update it externally periodically can do the trick.
    bw1 = bw;
    % 2. Using the measurement w1 and b1, we form the estimate of the new turn rate
w_hat_1
    w_hat_1 = w1-bw1 ;
    a_hat_1 = C'*(a1-ba1)-g;
```

```
        % 3. We propagate the quaternion using a first order integrator with w_hat_0 and
w_hat_1 tp obtain q_hat_avg_1
        w_avg = (w_hat_0 + w_hat_1) / 2;
        pos_hat_avg_1= pos_hat_avg_0+vel_hat_avg_0*dt;
        vel_hat_avg_1=vel_hat_avg_0+a_hat_1*dt;
        q_hat_avg_1 = NormalizeV((expm(1/2*Omega(w_avg)*dt) +
1/48*(Omega(w_hat_1)*Omega(w_hat_0) - Omega(w_hat_0)*Omega(w_hat_1))*dt^2)
* q_hat_avg_0);
        lambda1=lambda;
        posgps1=posgps;
        C=qGetRotation(q_hat_avg_1);
        xV(:,ctr)=[pos_hat_avg_1;vel_hat_avg_1;q_hat_avg_1;bw1;ba1;lambda1;posgps1];
        F = propogateMatrix(a1,w1,ba1,bw1,C,dt);
        Qd=returnQd(a1,w1,ba1,bw1,C,dt,Qc);
        % 5. We Compute the state covariance matrix according to the Extended Kalman
Filter equation
        P=double(P);
        P = double(F*P*F' + Qd);
        prevTime=imudata1.t(k);
        k=k+1;
        ctr=ctr+1;
    else
        currTime=leicadata.t(measItr);
        dt=double(currTime-prevTime)/1e9;
        sV(1:3,ctr)=leicadata.pos(1:3,measItr);
        w0=imudata2.omega(1:3,k-1);
        w1=imudata2.omega(1:3,k);
%       a11=marzulloFusion({[imudata1.a(1,k)-2.5,imudata1.a(1,k)+1],[imudata2.a(1,k)-
8,imudata2.a(1,k)+7],[imudata3.a(1,k)-2,imudata3.a(1,k)+1],[imudata4.a(1,k)-
1,imudata4.a(1,k)+2]},4,1);
```

```
%       a12=marzulloFusion({[imudata1.a(2,k)-2.5,imudata1.a(2,k)+1],[imudata2.a(2,k)-
8,imudata2.a(2,k)+7],[imudata3.a(2,k)-2,imudata3.a(2,k)+1],[imudata4.a(2,k)-
1,imudata4.a(2,k)+2]},4,1);
%       a13=marzulloFusion({[imudata1.a(3,k)-2.5,imudata1.a(3,k)+1],[imudata2.a(3,k)-
8,imudata2.a(3,k)+7],[imudata3.a(3,k)-2,imudata3.a(3,k)+1],[imudata4.a(3,k)-
1,imudata4.a(3,k)+2]},4,1);
%       a1=[a11;a12;a13];
% a1=(imudata1.a(1:3,k)+imudata2.a(1:3,k)+imudata3.a(1:3,k)+imudata4.a(1:3,k))/4;
%
a11=approxAgreement([imudata1.a(1,k);imudata2.a(1,k);imudata3.a(1,k);imudata4.a(1,k
)],1);
%       a12=approxAgreement([imudata1.a(2,k);imudata2.a(2,k);imudata3.a(2,k);imudat
a4.a(2,k)],1);
%       a13=approxAgreement([imudata1.a(3,k);imudata2.a(3,k);imudata3.a(3,k);imudat
a4.a(3,k)],1);
%       a1=[a11;a12;a13];
a11=sensorFusion({[imudata1.a(1,k)-0.5,imudata1.a(1,k)+0.5],[imudata2.a(1,k)-
0.5,imudata2.a(1,k)+0.5],[imudata3.a(1,k)-0.5,imudata3.a(1,k)+0.5],[imudata4.a(1,k)-
0.5,imudata4.a(1,k)+0.5]},4,1);
    a12=sensorFusion({[imudata1.a(2,k)-0.5,imudata1.a(2,k)+0.5],[imudata2.a(2,k)-
0.5,imudata2.a(2,k)+0.5],[imudata3.a(2,k)-0.5,imudata3.a(2,k)+0.5],[imudata4.a(2,k)-
0.5,imudata4.a(2,k)+0.5]},4,1);
    a13=sensorFusion({[imudata1.a(3,k)-0.5,imudata1.a(3,k)+0.5],[imudata2.a(3,k)-
0.5,imudata2.a(3,k)+0.5],[imudata3.a(3,k)-0.5,imudata3.a(3,k)+0.5],[imudata4.a(3,k)-
0.5,imudata4.a(3,k)+0.5]},4,1);
    a1=[a11;a12;a13];
    pos1=leicadata.pos(1:3,measItr);
    [xV(:,ctr),P]=ekf(Qc,xV(:,ctr-1),P,[w0 w1],a1,pos1,errZ,dt);
    P=double(P);
    prevTime=leicadata.t(measItr);
    measItr=measItr+1;
```

```
        ctr=ctr+1;
    end
end

figure
[t,len]=size(xV);
plot3(xV(1,2:len),xV(2,2:len),xV(3,2:len),grounddata.pos(1,:),grounddata.pos(2,:),groun
ddata.pos(3,:))
title('Position in 3-D Frame(Trajectory)')
xlabel('Position along X axis(m)')
ylabel('Position along Y axis(m)')
zlabel('Position along Z axis(m)')
legend('Estimated Position','Ground Truth','Location','northwest');

figure
hold on
plot(xV(1,2:len))
plot(grounddata.pos(1,:))
title('X Coordinate Estimate and Actual Value')
ylabel('Position along X axis(m)')
xlabel('Time(s)')
legend('Estimated Position','Ground Truth','Location','northwest');

figure
hold on
plot(xV(2,2:len))
plot(grounddata.pos(2,:))
title('Y Coordinate Estimate and Actual Value')
ylabel('Position along Y axis(m)')
xlabel('Time(s)')
legend('Estimated Position','Ground Truth','Location','northwest');
```

```
figure
hold on
plot(xV(3,2:len))
plot(grounddata.pos(3,:))
title('Z Coordinate Estimate and Actual Value')
ylabel('Position along Z axis(m)')
xlabel('Time(s)')
legend('Estimated Position','Ground Truth','Location','northwest');
```

**2. EKF Function**

```
% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Simulator - Extended Kalman Filter
%
% Description:This function is responsible for the EKF Predict and Update
% steps.
function [ xV1, P1] = ekf( Qc,xV0, P0, w,a1,gps0,  errZ, dt)
% xV0 = current estimate of state
% w1=input gyroscope readings
% a1 = input accelerometer readings
% gps0 = input gps readings
% P0 = state covariance matrix
% Qc = noise covariance matrix
%% Initial Setup
varNw=0.0135;
```

```
varNa=(0.23*1e-6);
% We have gyroscope measurements w0 and w1.
w0 = w(:,1);
w1 = w(:,2);
% Gravity
g=[0;0;9.8];
% We have an estimate of the quaternion q_hat_avg and the bias b0
% xV0 = [pos; vel; q_hat_avg_0 ;ba; bw;lambda;pos]
pos_hat_avg_0=xV0(1:3);
vel_hat_avg_0=xV0(4:6);
q_hat_avg_0 = xV0(7:10);
bw_0=xV0(11:13);
ba_0 = xV0(14:16);
lambda_0 = xV0(17);
posgps_0 = xV0(18:20);

C=qGetRotation(q_hat_avg_0);
w_hat_0 = w0 - bw_0-varNw*randn;
%% State Prediction
% Propagate the bias
ba1 = ba_0;
bw1 = bw_0;
% Estimate new angular velocity and linear acceleration using the measurement w1 and
b1
w_hat_1 = w1 - bw1-varNw*randn;
a_hat_1 = C'*(a1 - ba1-varNa*randn)-g;
% Propagate the quaternion using first order integrator
w_avg = (w_hat_0 + w_hat_1) / 2;
pos_hat_avg_1= pos_hat_avg_0+vel_hat_avg_0*dt;
vel_hat_avg_1=vel_hat_avg_0+a_hat_1*dt;
```

```
q_hat_avg_1 = NormalizeV((exp(1/2*Omega(w_avg)*dt) +
1/48*(Omega(w_hat_1)*Omega(w_hat_0) - Omega(w_hat_0)*Omega(w_hat_1))*dt^2)
* q_hat_avg_0);
lambda1=lambda_0;
posgps1=posgps_0;
xV0=[pos_hat_avg_1;vel_hat_avg_1;q_hat_avg_1;bw1;ba1;lambda1;posgps1];
% Compute the transition matrix F and Qd
C=qGetRotation(q_hat_avg_1);
F = propogateMatrix(a1,w1,ba1,bw1,C,dt);
Qd=returnQd(a1,w1,ba1,bw1,C,dt,Qc);
% Update the state covariance matrix
P1_ = F*P0*F' + Qd;

%% Compute Kalman Gain
% Calculate the measurement matrix H
H = returnH(lambda1,C,posgps1,pos_hat_avg_1);
S = H*P1_*H' + errZ;
K = P1_*(H'*inv(S));

% Calculate residual error r according to r = z - z_hat
r=gps0-pos_hat_avg_1;
deltax=double(K*r);
dq=deltax(7:9)/2;
if (dq'*dq) > 1
   dq_hat_avg_1 = (1/sqrt(1 + (dq'*dq))) * [dq ; 1];
else
   dq_hat_avg_1 = [dq ; (sqrt(1 - (dq'*dq)))];
end
q = quaternionMult(dq_hat_avg_1, q_hat_avg_1);
temp=[deltax(1:6);q;deltax(10:19)];
xV1=xV0+temp;
```

```
xV1(7:10)=q;
P1 = double((eye(19) - K*H) * P1_ * (eye(19) - K*H)' + K*errZ*K');
end
```

### 3. Brooks Iyengar Sensor Fusion Function

```
% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Simulator - Brooks Iyengar Fusion
%
% Description:This function is used to fuse the readings from the IMUs
% using the Brooks-Iyengar Fusion
%


function [pointEstimate] = sensorFusion(intervals, N, a)

   %Divide intervals into mutually exclusive intervals...HOW?

   %create empty array that will store all possible interval bounds (upper as well as
lower, independently)
   I = [];

   %now fill up this array with all values from intervals
   for i = 1 : N
      I = [I intervals{i}(1) intervals{i}(2)];
   end
```

```
% delete repeated values from this array and sort it
I = unique(I);

%now create an empty cell array, each of whose entries will store the
%newly created intervals from I(i) & I(i+1), as well as the weights of
%each interval
A = {};
for i = 1 : (length(I) - 1)
   w = 0;
   x1 = I(i);
   y1 = I(i+1);
   for j = 1 : N
      x2 = intervals{j}(1);
      y2 = intervals{j}(2);
      %fprintf("[%f,%f] ... [%f,%f] ... ? \n",x1,y1,x2,y2)
      if ((x1 <= x2) && (x2 < y1)) || ((x1 < y2) && (y2 < y1)) || ((x1 >= x2) && (x1 <
y2)) || ((y1 > x2)&&(y1 < y2))
         w = w + 1;
         %fprintf("\tyes! w = %d\n")
      end
   end
   A{i} = {[x1, y1],w};
   %fprintf("[%f, %f],%d\n",A{i}{1}(1),A{i}{1}(2),A{i}{2})
end

%Let's say after the above step, we obtain our new intervals with the
%corresponding weights
%remove intervals with weights less than N - a
%{
for i = length(A):1
```

```
    %disp(A{i}{1}(1))
    %disp(A{i}{1}(2))
    w = (A{i}{2});
    if (w < N - a)
        A(:,i) = [];
        disp(A{i})
    end
end
%}

%let's just print A to see everything is fine

%{
for i = 1 : length(A)
    fprintf("[%f, %f],%d\n",A{i}{1}(1),A{i}{1}(2),A{i}{2})
end
%}


foundFirstInterval = false;
lastInterval = 0;
firstInterval = 0;

P_E_Numerator = 0;
P_E_Denominator = 0;

for i = 1 : length(A)
    if(A{i}{2} >= N-a)
        %fprintf("%d >= %d ? \n",A{i}{2}, N-a)
        if (foundFirstInterval == false)
            foundFirstInterval = true;
```

```
            firstInterval = A{i}{1}(1); %first found interval where W >= N-a is the overall
lower interval
        end
        lastInterval = A{i}{1}(2);      %last interval will always be the latest interval for
W >= N-a
        P_E_Numerator = P_E_Numerator + A{i}{2}*(A{i}{1}(1) + A{i}{1}(2))/2;
        P_E_Denominator = P_E_Denominator + A{i}{2};
    end
  end

  pointEstimate = P_E_Numerator / P_E_Denominator  ;   %Final point estimate
  outputIntervals = [firstInterval, lastInterval]  ;   %Final intervals
end
```

**4. Helper Functions**

```
% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Skew Matrix
%
% Description:This function is responsible for returning the skew matrix
% representation for a said quaternion vector.
function [ quatx ] = skew( quat )
  quatx = [ 0   -quat(3)  quat(2);
       quat(3)   0   -quat(1);
      -quat(2)  quat(1)   0 ];
end
```

```
% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Function - Noise Covariance Matrix
%
% Description:This function is responsible for calculating the system noise
% covariance matrix.

function Qd=returnQd(a,w,ba,bw,C,dt,Qc)

    if (dt~=0)
       Qc(1:3,:)=Qc(1:3,:)/dt;
       Qc(4:6,:)=Qc(4:6,:)*dt;
       Qc(7:9,:)=Qc(7:9,:)/dt;
       Qc(10:12,:)=Qc(10:12,:)*dt;
    end



    zero=zeros(3,3);
    iden=eye(3,3);
    a=a-ba;
    w=w-bw;
    A=-C'*skew(a)*((iden*((dt^2)/2))-
(((dt^3)/6)*skew(w))+(((dt^4)/24)*skew(w)*skew(w)));
    B=-C'*skew(a)*(-(iden*((dt^3)/6))+(((dt^4)/24)*skew(w))-
(((dt^5)/120)*skew(w)*skew(w)));
```

```
    Ch=-C'*skew(a)*((iden*dt)-(((dt^2)/4)*skew(w))+(((dt^3)/6)*skew(w)*skew(w)));
    D=-A;
    E=iden-dt*skew(w)+0.5*dt*dt*skew(w)*skew(w);
    F=-iden*dt+0.5*dt*dt*skew(w)-((dt^3)/6)*skew(w)*skew(w);
    Fd=double([iden dt*iden A B -C'*dt*dt*(1/2) zeros(3,4);
        zero iden Ch D -C'*dt zeros(3,4);
        zero zero E F zero zeros(3,4);
        zero zero zero iden zero zeros(3,4);
        zero zero zero zero iden zeros(3,4);
        zeros(4,3) zeros(4,3) zeros(4,3) zeros(4,3) zeros(4,3) zeros(4,4)]);
    G=[zero zero zero zero;
      -C' zero zero zero;
       zero zero -iden zero;
       zero zero zero iden;
       zero iden zero zero;
       zeros(4,3) zeros(4,3) zeros(4,3) zeros(4,3)];
    Qd=double(Fd*G*Qc*G'*Fd');

end

% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
% Project: Secure State Estimation for Quadrotors
% Component: Function - Observation Matrix
%
% Description:This function is responsible for calculating the observation
% matrix.

function H=returnH(lambda,C,ps,pw)
```

```
    H1=lambda*eye(3,3);
    zer=zeros(3,3);
    H3=-C'*skew(ps)*lambda;
    H6=pw+C'*ps;
    H7=C'*lambda;
    H=double([H1 ;zer; H3; zer ;zer; H6'; H7]');

end

% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Function - Multiply Quaternions
%
% Description:This function is responsible for multiplying two quaternions

function mult=quaternionMult(q_quat,p)
    mult=[q_quat(4)*p(1) + q_quat(3)*p(2) - q_quat(2)*p(3) + q_quat(1)*p(4);
-q_quat(3)*p(1) + q_quat(4)*p(2) + q_quat(1)*p(3) + q_quat(2)*p(4);
q_quat(2)*p(1) - q_quat(1)*p(2) + q_quat(4)*p(3) + q_quat(3)*p(4);
-q_quat(1)*p(1) - q_quat(2)*p(2) - q_quat(3)*p(3) + q_quat(4)*p_quat(4)];
end

% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
```

```
% Project: Secure State Estimation for Quadrotors
% Component: Function - Rotation Matrix
%
% Description:This function is responsible for calculating the rotation
% matrix for the quaternion vector

function c = qGetRotation( quat )
c = eye(3) - 2*quat(4)*skew(quat) + 2*(skew(quat)*skew(quat));

% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Function - Propogate Matrix
%
% Description:This function is responsible propogating the state variable
% and updating the covariance matrix.

function x=propogateMatrix(a,w,ba,bw,C,dt)
   zero=zeros(3,3);
   iden=eye(3,3);
   a=a-ba;
   w=w-bw;
   A=-C'*skew(a)*((iden*((dt^2)/2))-
(((dt^3)/6)*skew(w))+(((dt^4)/24)*skew(w)*skew(w)));
   B=-C'*skew(a)*(-(iden*((dt^3)/6))+(((dt^4)/24)*skew(w))-
(((dt^5)/120)*skew(w)*skew(w)));
   Ch=-C'*skew(a)*((iden*dt)-(((dt^2)/4)*skew(w))+(((dt^3)/6)*skew(w)*skew(w)));
   D=-A;
```

```
    E=iden-dt*skew(w)+0.5*dt*dt*skew(w)*skew(w);
    F=-iden*dt+0.5*dt*dt*skew(w)-((dt^3)/6)*skew(w)*skew(w);
    x=double([iden dt*iden A B -C'*dt*dt*(1/2) zeros(3,4);
        zero iden Ch D -C'*dt zeros(3,4);
        zero zero E F zero zeros(3,4);
        zero zero zero iden zero zeros(3,4);
        zero zero zero zero iden zeros(3,4);
        zeros(4,3) zeros(4,3) zeros(4,3) zeros(4,3) zeros(4,3) zeros(4,4)]);
end

% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Function - Omega Representation
%
% Description:This function is responsible for evaluating the omega
% representation based on the angular velocity.
function [omega] = Omega(w)
    omega = [ 0    w(3) -w(2) w(1);
            -w(3) 0     w(1) w(2);
             w(2) -w(1) 0    w(3);
            -w(1) -w(2) -w(3) 0 ];
end

% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
```

```
%
% Project: Secure State Estimation for Quadrotors
% Component: Function - Normalization Function
%
% Description:This function is a generic normalizer function.
function [normalized] = NormalizeV(aVector)
   normalized = aVector./norm(aVector);
end

% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Function - Convert To Quaternion
%
% Description:This function is responsible for converting a axisangle
% rotation to a quaternion representation.
function [quat] = convertToQuaternion(angle)
quat = [angle(1:3).*sin(angle(4)/2); cos(angle(4)/2)];
end

% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Function - Convert To Axis Angle
%
```

```
% Description:This function is responsible for converting a quaternion
% vector to its axis angle representation.
function [angle] = convertToAxisAngle(quat)
t = wrapToPi(2*acos(quat(4)));
k = quat(1:3)./sin(t/2);
angle = [NormalizeV(k);t];
end
%
% function x=wrapToPi(a)
% x = a - 2*pi*floor( (a)/(2*pi) );
% end

% Date: Thu, 20 Feb, 2017
% Last updated: 15 March, 2017
% Author: Akshay Prasad
% Organization: University of Maryland, College Park (MSSE Student)
%
% Project: Secure State Estimation for Quadrotors
% Component: Function - Gc Matrix
%
% Description:This function calculates the Gc matrix
function [ x ] = calculateGc( C )
   temp=zeros(3);
   I_3=eye(3);
   x=[temp temp temp temp;
      -C' temp temp temp;
      temp temp -I_3 temp;
      temp temp temp I_3;
      temp I_3 temp temp;
      zeros(4,3) zeros(4,3) zeros(4,3) zeros(4,3)
      ];
```

```
end
```

# References

[1] A. Greenberg, "Hackers Remotely Kill a Jeep on the Highway—With Me in It", *Wired,* July 2015.

[2] N. O. Tippenhauer *et al*, "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM Conference on Computer and Communications Security,* 2011, pp. 75-86.

[3] Y. Shoukry *et al*, "Non-invasive spoofing attacks for anti-lock braking systems," in *International Workshop on Cryptographic Hardware and Embedded Systems,* 2013, pp. 55-72.

[4] A. Teixeira *et al*, "Attack models and scenarios for networked control systems," in *Proceedings of the 1st International Conference on High Confidence Networked Systems,* 2012, pp. 55–64.

[5] M. Pajic *et al*. Design and implementation of attack-resilient cyberphysical systems: With a focus on attack-resilient state estimators. *IEEE Control Systems 37(2),* pp. 66-81. 2017. DOI: 10.1109/MCS.2016.2643239.

[6] *Amazon Prime Air*. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

[7] Project Wing. *Project Wing*. Available: https://x.company.

[8] (December 14). *FAA Announces Small UAS Registration Rule*. Available: https://www.faa.gov/news/press_releases/news_story.cfm?newsId=19856.

[9] Federal Aviation Administration, "FAA aerospace forecasts".

fiscal years 2016-2036.

[10] C. Arthur. SkyGrabber: The $26 software used by insurgents to hack into US drones. *The Guardian* 2009. Available: https://www.theguardian.com/technology/2009/dec/17/skygrabber-software-drones-hacked.

[11] S. Gorman, Y. J. Dreazen and A. Cole. Insurgents hack U.S. drones. *Wall Street Journal* 2009. Available: http://www.wsj.com/articles/SB126102247889095011.

[12] F. Gardner. Iran shows film of captured US drone. *BBC News* 2011. Available: http://www.bbc.com/news/world-middle-east-16098562.

[13] P. Peterson and S. Faramarzi, "Iran hijacked US drone, says Iranian engineer," *Christian Science Monitor,* December 2011.

[14] D. Shepard, J. A. Bhatti and T. E. Humphreys, "Drone Hack: Spoofing Attack Demonstration on a Civilian Unmanned Aerial Vehicle," *GPS World,* August. 2012.

[15] N. M. Rodday. "Exploring Security Vulnerabilities of Unmanned Aerial Vehicles", University of Twente, 2015.

[16] T. H. Cox *et al*, "Civil UAV capability assessment," *NASA, Tech.Rep., Draft Version,* 2004.

[17] R. Austin, *Unmanned Aircraft Systems: UAVS Design, Development and Deployment.* Wiley, 2010.

[18] (). *Pixhawk - Open source Hardware*. Available: https://pixhawk.org/.

[19] A. Kim *et al*. Cyber attack vulnerabilities analysis for unmanned aerial vehicles. Presented at Infotech@Aerospace 2012. June 19, 2012, Available: https://arc.aiaa.org/doi/abs/10.2514/6.2012-2438.

[20] J. Villasenor, "Compromised by design? securing the defense electronics supply chain," November 4, 2013.

[21] S. Gallagher. (8/15/). *Parrot drones easily taken down or hijacked, researchers demonstrate*. Available: https://arstechnica.com/security/2015/08/parrot-drones-easily-taken-down-or-hijacked-researchers-demonstrate/.

[22] D. Kushner, "The Real Story of Stuxnet," *IEEE Spectrum: Technology, Engineering, and Science News,* 2/26/. 2013.

[23] Y. Son *et al*, "Rocking drones with intentional sound noise on gyroscopic sensors," in *24th USENIX Security Symposium (USENIX Security 15),* 2015, pp. 896.

[24] S. M. Weiss. "Vision Based Navigation for Micro Helicopters", ETH Zürich, 2012.

[25] M. W. Achtelik, "Advanced Closed Loop Visual Navigation for Micro Aerial Vehicles", ETH-Zürich, 2014.

[26] N. Trawny and S. Roumeliotis, "Indirect kalman filter for 3D attitude estimation," University of Minnesota, March. 2005.

[27] R. S. Smith, "A decoupled feedback structure for covertly appropriating networked control systems," IFAC Proceedings Volumes, vol. 44, pp. 90-95, 2011.

[28] C. Scherer, *Theory of Robust Control.* Delft University of Technology, 2001.

[29] H. J. LeBlanc *et al*. Resilient asymptotic consensus in robust networks. *IEEE Journal on Selected Areas in Communications 31(4),* pp. 766-781. 2013. DOI: 10.1109/JSAC.2013.130413.

[30] Mitra and S. Sundaram, "Secure distributed observers for a class of linear time invariant systems in the presence of byzantine adversaries," in *IEEE 55th Conference on Decision and Control (CDC),* December 2016, pp. 2709-2714.

[31] H. Zhang and S. Sundaram, "A simple median-based resilient consensus algorithm," in *50th Annual Allerton Conference on Communication, Control, and Computing (Allerton),* October 2012, pp. 1734-1741.

[32] S. Sundaram *et al*, "The wireless control network: Monitoring for malicious behavior," in *49th IEEE Conference on Decision and Control (CDC),* December 2010, pp. 5979-5984.

[33] Y. Mo, R. Chabukswar and B. Sinopoli. Detecting integrity attacks on SCADA systems. *IEEE Transactions on Control Systems Technology 22(4),* pp. 1396-1407. 2014. DOI: 10.1109/TCST.2013.2280899.

[34] Mo, S. Weerakkody and B. Sinopoli. Physical authentication of control systems: Designing watermarked control inputs to detect counterfeit sensor outputs. *IEEE Control Systems 35(1),* pp. 93-109. 2015. DOI: 10.1109/MCS.2014.2364724.

[35] F. Pasqualetti, F. Dörfler and F. Bullo, "Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design," in *50th IEEE Conference on Decision and Control and European Control Conference,* December 2011, pp. 2195-2201.

[36] M. H. Manshaei *et al*. Game theory meets network security and privacy. *ACM Comput. Surv. 45(3),* pp. 25:1–25:39. 2013. Available: http://doi.acm.org/10.1145/2480741.2480742. DOI: 10.1145/2480741.2480742.

[37] A. Gueye, V. Marbukh and J. C. Walrand, "Towards a metric for communication network vulnerability to attacks: A game theoretic approach," in *Game Theory for Networks,* 2012/5/24, pp. 259-274.

[38] F. Miao, M. Pajic and G. J. Pappas, "Stochastic game approach for replay attack detection," in *52nd IEEE Conference on Decision and Control,* December 2013, pp. 1854-1859.

[39] H. Fawzi, P. Tabuada and S. Diggavi. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Transactions on Automatic Control 59(6),* pp. 1454- 1467. 2014. DOI: 10.1109/TAC.2014.2303233.

[40] Q. Hu, Y. H. Chang and C. J. Tomlin. Secure estimation for unmanned aerial vehicles against adversarial cyber attacks. 2016. Available: https://arxiv.org/abs/1606.04176.

[41] Y. Shoukry *et al*. Secure state estimation for cyber physical systems under sensor attacks: A satisfiability modulo theory approach. *IEEE Transactions on Automatic Control PP(99),* pp. 1-1. 2017. DOI: 10.1109/TAC.2017.2676679.

[42] M. S. Chong, M. Wakaiki and J. P. Hespanha, "Observability of linear systems under adversarial attacks," in *American Control Conference (ACC),* July 2015, pp. 2439-2444.

[43] M. Pajic *et al*, "Robustness of attack-resilient state estimators," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS),* April 2014, pp. 163-174.

[44] M. Pajic *et al*, "Attack-resilient state estimation in the presence of noise," in December 2015, pp. 5827-5832.

[45] J. Weimer *et al*, "Towards assurance cases for resilient control systems," in *IEEE International Conference on Cyber-Physical Systems, Networks, and Applications,* August 2014, pp. 1-6.

[46] H. Jeon *et al*, "Resilient State Estimation for Control Systems Using Multiple Observers and Median Operation," *Mathematical Problems in Engineering,* vol. 2016, 2016.

[47] S. Z. Yong, M. Zhu and E. Frazzoli, "Resilient state estimation against switching attacks on stochastic cyber-physical systems," in *54th IEEE Conference on Decision and Control (CDC),* December 2015, pp. 5162-5169.

[48] N. Forti *et al*, "A bayesian approach to joint attack detection and resilient state estimation," in *IEEE 55th Conference on Decision and Control (CDC),* December 2016, pp. 1192-1198.

[49] Y. Nakahira and Y. Mo, "Dynamic state estimation in the presence of compromised sensory data," in *54th IEEE Conference on Decision and Control (CDC),* December 2015, pp. 5808- 5813.

[50] D. Han, Y. Mo and L. Xie, "Towards a unified resilience analysis: State estimation against integrity attacks," in *35th Chinese Control Conference (CCC),* July 2016, pp. 7333-7340.

[51] L. Lamport, R. Shostak and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst. 4(3),* pp. 382–401. 1982. Available: http://doi.acm.org/10.1145/357172.357176. DOI: 10.1145/357172.357176.

[52] D. Dolev *et al*. Reaching approximate agreement in the presence of faults. *J. Acm 33(3),* pp. 499–516. 1986. Available: http://doi.acm.org/10.1145/5925.5931. DOI: 10.1145/5925.5931.

[53] S. R. Mahaney and F. B. Schneider, "Inexact agreement: Accuracy, precision, and graceful degradation," in *Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing,* 1985, pp. 237–249.

[54] N. H. Vaidya and V. K. Garg, "Byzantine vector consensus in complete graphs," in *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing,* 2013, pp. 65–73.

[55] H. Mendes and M. Herlihy, "Multidimensional approximate agreement in byzantine asynchronous systems," in *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing,* 2013, pp. 391–400.

[56] B. Ao *et al*. On precision bound of distributed fault-tolerant sensor fusion algorithms. *ACM Comput. Surv. 49(1),* pp. 5:1–5:23. 2016. Available: http://doi.acm.org/10.1145/2898984. DOI: 10.1145/2898984.

[57] R. R. Brooks and S. S. Iyengar, "Optimal matching algorithm for multidimensional sensor readings," in *Photonics East'95,* 1995, pp. 91-99.

[58] K. Marzullo. Tolerating failures of continuous-valued sensors. *ACM Trans. Comput. Syst. 8(4),* pp. 284–304. 1990. Available: http://doi.acm.org/10.1145/128733.128735. DOI: 10.1145/128733.128735.

[59] R. R. Brooks and S. S. Iyengar. Robust distributed computing and sensing algorithm. *Computer 29(6),* pp. 53–60. 1996. Available: http://dx.doi.org/10.1109/2.507632. DOI: 10.1109/2.507632.

[60] M. Burri *et al*. The EuRoC micro aerial vehicle datasets. *The Int'L Journal of Robotics Research 35(10),* pp. 1157-1163. 2016. Available: http://journals.sagepub.com/doi/abs/10.1177/0278364915620033. DOI: 10.1177/0278364915620033.

[61] *The EuRoC MAV Dataset*. Available: http://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets.

[62] Besse, Philippe, et al. "Review and perspective for distance based trajectory clustering." *arXiv preprint arXiv:1508.04904* (2015).