Fault-Tolerant Sensor Fusion for Aircraft Height Estimation

by

Adrian Peter Wiegman

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering in Electric and Electronic Engineering in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr J.A.A Engelbrecht
Prof H.A. Engelbrecht

April 2019

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....

Abstract

This thesis presents the design and verification of a fault-tolerant sensor fusion system to provide robust height estimates for commercial airliners. Current commercial aircraft systems obtain a height estimate by taking the median of redundant radio altimeter measurements. However, this approach is not robust to a failure that affects all radio altimeters. The proposed system uses technology redundancy by combining the measurements from the available aircraft sensors that provide either height or altitude measurements: the radio altimeter, the GPS, the inertial sensors, and the instrument landing system (ILS). An onboard terrain map is used to convert altitude measurements to height measurements.

The robust height estimation problem is divided into two major sub-problems: *fault detection and isolation* and *sensor fusion*. To address the fault detection and isolation sub-problem, a variety of data-driven and model-based techniques were investigated. Two general approaches were considered: fault diagnosis using sensor measurements from a single time instant, and fault diagnosis using sensor measurements from a window of consecutive time instants. For the single time instant approach, only data-driven techniques were applied, including outlier detectors, binary classifiers, and multi-class classifiers. For the multiple time instant approach, both model-based and data-driven techniques were applied. The model-based techniques included the Bank of Kalman filters and the Robust Kalman filter, while the data-driven techniques include Model Consensus, Dynamic Principal Component Analysis, and Gaussian Naïve Bayes with time as an additional variable. Finally, to address the sensor fusion sub-problem, three methods were investigated: using the median of the sensor measurements, using the weighted average of the sensor measurements, and using a Kalman filter to perform optimal sensor fusion.

A simulation model was used to generate synthetic training and testing data for the data-driven techniques. Mathematical models were established for the aircraft motion, the sensors, and the terrain. The structure and nominal parameters for the sensor models were based on information sourced from literature, and then the sensor parameters were tuned to fit a real dataset provided by Airbus. Fault models for six types of sensor faults were also created. The simulation model was used to generate a large dataset of representative sensor measurements containing both "no-fault" and "fault" conditions.

The fault detection and isolation, and the sensor fusion were tested using both simulated data and real datasets of actual flight data with synthetic sensor failures injected. The single time instant approach achieved fault diagnosis accuracies from 85 % (Support Vector Machine) to 94 % (k-Nearest Neighbors). The multiple time instant approach achieved fault diagnosis accuracies from 93 % (Model Consensus) to 99 % (Robust Kalman filter). The three sensor fusion approaches produced height estimates with average accuracies from 5.1 m to 7.4 m.

Uittreksel

Hierdie tesis beskryf die ontwerp en verifikasie aan van 'n fouttolerante sensorfusie-stelsel om betroubare hoogte afskattings vir kommersiële vliegtuie te verskaf. Huidige kommersiële vliegtuigstelsels verkry 'n hoogte afskatting deur die mediaan van veelvuldige radio-hoogtemeter metings te neem. Hierdie benadering is egter nie betroubaar vir 'n fout wat alle radio-hoogtemeters gelyktydig beïnvloed nie. Die voorgestelde stelsel gebruik tegnologie oorbodigheid deur die metings te kombineer van die beskikbare vliegtuig sensors wat hoogte bo grond of hoogte bo seevlak meet: die radio-hoogtemeter, die GPS, die inersiële sensors, en die instrumentlandingstelsel (ILS).

Die robuuste hoogte afskatting probleem word verdeel in twee subprobleme: *foutopsporing en isolasie*, en *sensorfusie*. Om die foutopsporing en isolasie subprobleem aan te spreek, is 'n verskeidenheid data-gedrewe en modelgebaseerde tegnieke ondersoek. Twee algemene benaderings is oorweeg: foutdiagnose met behulp van sensormetings vanaf 'n enkele tydstip, en foutdiagnose met behulp van sensormetings vanaf 'n venster van opeenvolgende tydstippe. Vir die enkel tydstip benadering is verskeie data-gedrewe tegnieke toegepas, insluitende uitskieter opspoorders, binêre klassifiseerdes en multi-klas klassifiseerders. Vir die meervoudige tydstip benadering is beide modelgebaseerde en data-gedrewe tegnieke toegepas. Die modelgebaseerde tegnieke het die Bank van Kalman filters en die Robuuste Kalman filter ingesluit, terwyl die data-gedrewe tegnieke Model Konsensus en Dinamiese Hoofkomponent Analise. Laastens, om die sensorfusie subprobleem aan te spreek is drie metodes ondersoek: gebruik van die mediaan van die sensormetings, gebruik van die geweegde gemiddelde van die sensormetings, en gebruik van 'n Kalman filter vir optimaal sensorfusie.

'n Simulasiemodel is ontwikkel om sintetiese opleiding en toetsingdata te skep vir die data-gedrewe tegnieke. Wiskundige modelle is afgelei vir die vliegtuig beweging, die sensors, en die terrein. Die struktuur en nominale parameters vir die sensormodelle is gegrond op inligting wat uit literatuur verkry is, waarna die sensorparameters ingestel is om 'n werklike datastel, wat deur Airbus verskaf is, te pas. Foutmodelle vir ses tipes sensorfoute is ook geskep. Die simulasiemodel is gebruik om 'n groot datastel van verteenwoordigende sensormetings te genereer wat beide "geen fout" en "fout" toestande bevat.

Die foutopsporing en isolasie en die sensorfusie is getoets met behulp van beide gesimuleerde data en datastelle van werklike vlugdata met byvoeging van sintetiese sensorfoute. Die enkel tydstip benadering foutdiagnose akkuraathede behaal van 85 % ("Support Vector Machine") tot 94 % (k-Naaste Bure). Die meervoudige tydstip benadering het foutdiagnose akkuraathede behaal van 93 % (Model Konsensus) tot 99 % (Robuuste Kalman filter). Die drie sensor-fusie benaderings het hoogte afskattings met gemiddelde akkuraatheid van 5.1 m tot 7.4 m gelewer.

Acknowledgements

I would firstly like to thank Airbus for funding this research and the invaluable insights that were made. A special mention must be made of Philippe Goupil and his team from Airbus Toulouse for their technical input and feedback. I would also like to thank Louise Travé of LAAS for her help and assistance in understanding the DyClee algorithm. I'm incredibly grateful for the time and effort invested by Prof Herman Engelbrecht and Dr Japie Engelbrecht into this project. Their guidance and contributions were invaluable. Lastly a massive thank-you to my girl friend, friends and family for their constant support and assistance during this time.

Contents

List of Figures

List of Tables

Nomenclature

Constants

$g = 9.81\,m/s^2$
$T_s = 0.125\,s$
$x_{st} = 286\,m$
$z_{st} = -8\,m$

Variables

μ_{terr}	Average terrain database error	[m]
σ_{terr}	Variance of terrain database error	[m^2]
γ	Glide Slope Angle	[°]
h	Aircraft height	[m]
x	Aircraft position along X_R axis	[m]
$\dot{x}$	Aircraft velcoity along X_R axis	[m]
z	Aircraft position along Z_R axis	[m]
$\dot{z}$	Aircraft velocity along X_R axis	[m]
H_{RA}	Radio altimeter height	[m]
X_{GPS}	GPS position along X_R axis	[m]
X_{IRS}	IRS position along X_R axis	[m]
$\dot{X}_{IRS}$	IRS velocity along X_R axis	[m/s]
Z_{GPS}	GPS relative altitude	[m]
Z_{ILS}	ILS relative altitude	[m]
Z_{IRS}	IRS relative altitude	[m]
$\dot{Z}_{IRS}$	IRS velocity along Z_R axis	[m/s]
$Z_{T1}(x)$	Terrain database with terrain height (100 m resolution)	[m]
$Z_{T2}(x)$	Terrain database with terrain height (500 m resolution)	[m]

Abbreviations

AFCS	Automatic Flight Control System
AGL	Above Ground Level
AIS	Aircraft Installation Delay
API	Application Program Interface
CART	Classification And Regression Tree
DDM	Difference in Depth of Modulation

D-PCA	Dynamic Principal Component Analysis
DT	Decision Tree
EE	Elliptic Envelope
ERR	Equal Error Rate
FDI	Fault Detection and Isolation
FMCW	Frequency Modulated Continuous Wave
GM	Gauss-Markov
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPS	Ground Proximity Warning System
HDOP	Horizontal Dilution Of Precision
HGMF	Hybrid Gaussian Mixture Filter
IF	Isolation Forest
ILS	Instrument Landing System
IRS	Inertial Sensor
KF	Kalman Filter
kNN	k-Nearest Neighbor
LOF	Local Outlier Factor
LR	Logistic Regression
MCD	Minimum Covariance Determinant
MEMS	Micro-Electro Mechanical Systems
MSL	Mean Sea Level
NB	Naïve Bayes
NED	North East Down
PCA	Principal Component Analysis
RA	Radio Altimeter
RBF	Radial Basis Function
RMS	Root Mean Square
ROC	Receiver Operating Characteristic
UAV	Unmanned Aerial Vehicle
UERE	User Equivalent Range Error
SPE	Square Projection Error
SVM	Support Vector Machine
VDOP	Vertical Dilution Of Precision

f

Chapter 1

Introduction

1.1 Background

This thesis presents the design and verification of a fault-tolerant sensor fusion system that provides robust height estimates for commercial airliners. The proposed system combines the measurements from all the aircraft sensors that provide either height or altitude measurements: the radio altimeter, the GPS, the inertial sensors, and the instrument landing system (ILS). An onboard terrain map is used to convert altitude measurements to height measurements. The system performs fault detection and isolation on the sensor measurements before combining them to produce an estimate of the aircraft height that is robust to sensor failure in at most one technology.

A reliable and accurate measurement of the aircraft height above the ground is required because it is used by the autopilot during automated landings. Currently, commercial aircraft systems use multiple redundant radio altimeter sensors and take the median of their height measurements. However, this approach is not robust to a failure that affects all radio altimeters. A malfunction of the radio altimeter system may cause erroneous autopilot behaviour during the landing phase that may potentially result in an aircraft crash. In 2009, Turkish Airlines flight TK1951 crashed during landing at Amsterdam Schipol airport due to a faulty radio altimeter reading. The autopilot reduced the engine thrust during approach, causing the aircraft to stall and crash. This incident prompted research into using other aircraft sensors to validate the radio altimeter measurement and provide a robust height estimate in the event of a radio altimeter system malfunction.

This project addresses the need to validate the radio altimeter measurement using the other aircraft sensors. The scope is expanded to incorporate fault detection and isolation on all of the sensors (not only the radio altimeter), before using sensor fusion to combine all the sensor measurements and produce an aircraft height estimate that is robust to a malfunction in any one sensor technology.

The robust height estimation problem presents two main challenges: Firstly, the radio altimeter provides a height measurement while all the other aircraft sensors provide altitude measurements. The sensor measurements must therefore be transformed to the same domain so that homogeneous variables are used for the fault diagnosis and sensor fusion. Secondly, the sensor characteristics are not stationary and change as a function of height, altitude, and time as the aircraft descends for landing.

1.2 Altitude versus Height

When considering the robust height estimation problem, it is important to distinguish between the following three terms: altitude, relative altitude, and height, as explained in figure 1.1. The altitude of the aircraft is its elevation above Mean Sea Level (MSL). The relative altitude of the aircraft is its elevation above the runway threshold. The height of the aircraft is its elevation above the terrain directly below it.

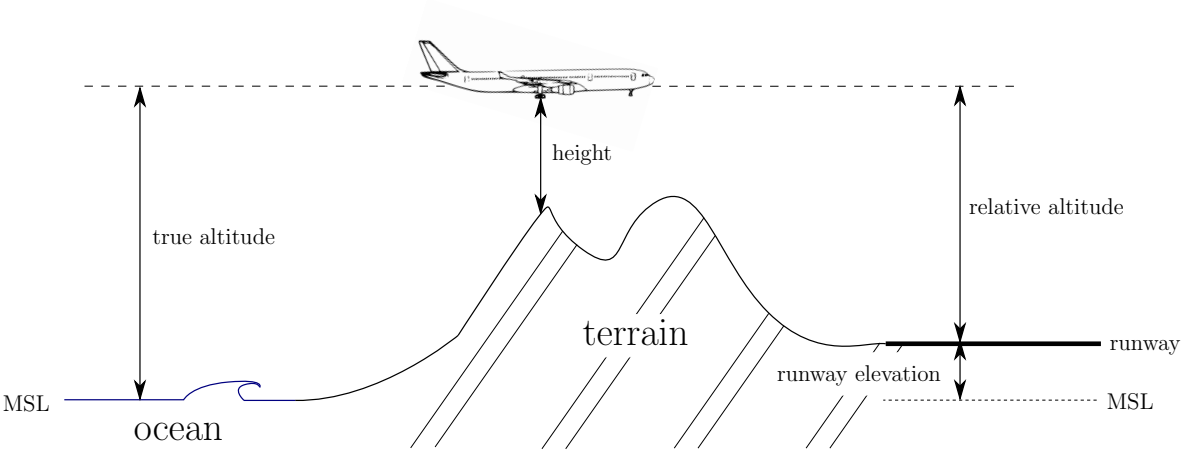


Figure 1.1: Terminology used when referring to the elevation of an aircraft.

The radio altimeter provides a measurement of the height above the terrain, the GPS sensor provides measurements of the altitude above Mean Sea Level, the inertial sensors propagate the altitude from a fixed reference (runway aircraft took off from), and the Instrument Landing System (ILS) provides sensor measurements of the aircraft's altitude relative to the runway threshold it is approaching.

1.3 Non-Stationary Sensor Characteristics

The sensor characteristics of the different aircraft sensors (radio altimeter, GPS sensor, inertial sensors, and ILS) are not stationary and change as a function of height, altitude, and time as the aircraft descends for landing, as illustrated in figure 1.2.

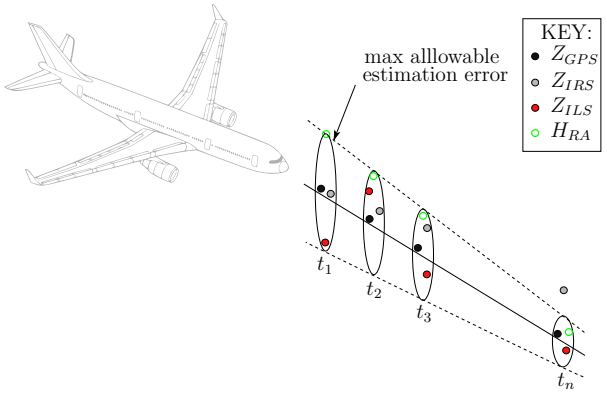


Figure 1.2: The maximum allowable estimation error decreases the closer an aircraft gets to landing. The sensor characteristics vary as a function of time, altitude and height.

The GPS sensor characteristics are independent of height, altitude or time, and do not change significantly over the course of the landing as illustrated in figure 1.3. The inertial sensor measurements become less accurate as time passes (due to sensor drift) as illustrated in figure 1.4. The ILS sensor measurements become more accurate as the aircraft descends (relative altitude decreases) as illustrated in figure 1.5. In figure 1.6 the radio altimeter measurements become more accurate as the aircraft descends (as height decreases).

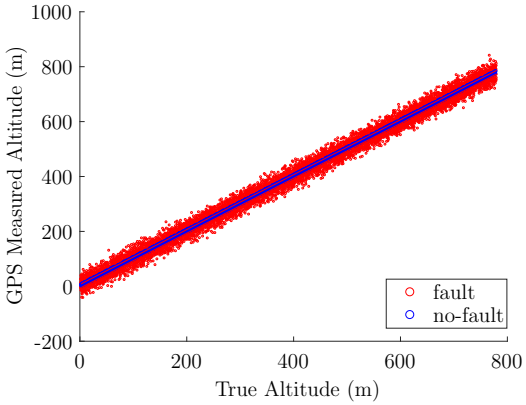


Figure 1.3: GPS sensor characteristics as a function of altitude.

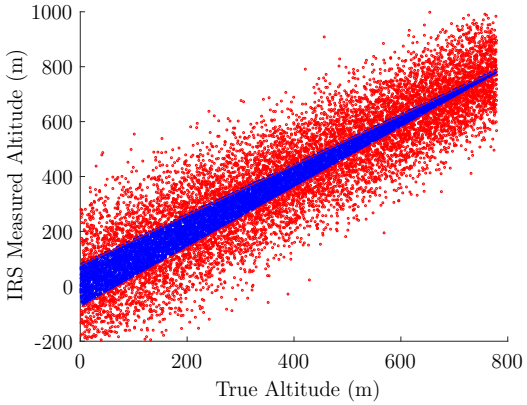


Figure 1.4: Inertial sensor characteristics as a function of altitude.

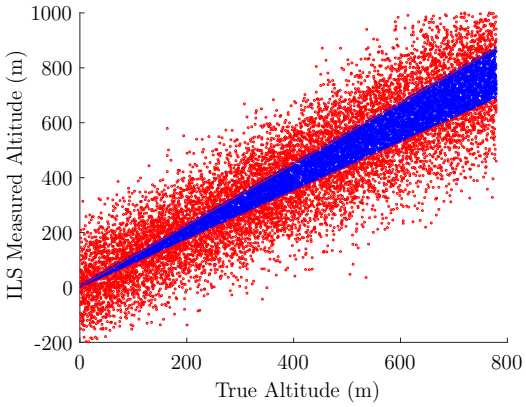


Figure 1.5: ILS sensor characteristics as a function of altitude.

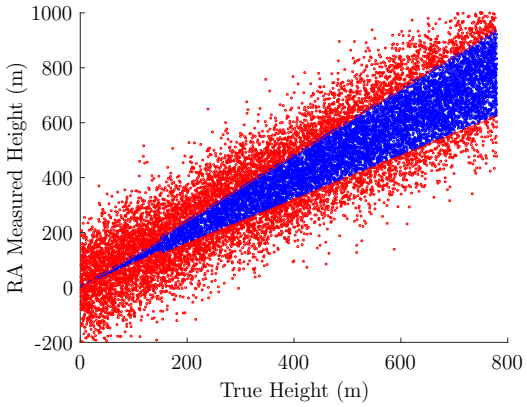


Figure 1.6: Radio altimeter sensor characteristics as a function of height.

1.4 Literature Review

Only a single application of aircraft height verification was found in literature. However, many applications were found that relate to sensor fusion and fault detection and isolation. This section provides an overview of the most applicable literature.

Thomas et al. [2] investigated the topic of robust sensor fusion for relative altimetry

with the aim of improving the height estimate of a commercial passenger aircraft by removing any undesired events relating to the misbehavior of the radio altimeters. Their approach establishes a Bank of Kalman filters for each sensor and its possible modes of operation, and uses probabilistic decision logic in the form of a likelihood function to determine the current operating mode.

Another application of robust sensor fusion was proposed by Szanfranski et al. [1] who developed a sensor fusion architecture for vertical take-offs and landings that fuses measurements from an ultrasonic range finder and a pressure altimeter for unmanned aerial vehicle (UAV) applications. Their solution is an alternative to hardware redundancy that is deemed unfavorable due to space and weight constraints [1]. Many of the sensor technologies used on UAVs are designed or selected for minimum size and cost, and therefore sacrifice some quality and reliability. The combination of two dissimilar sensors ensures that there are measurements available at both lower and higher heights. Their fusion methodology employs an initial filtering stage before sensor measurements are combined using weights that vary as a function of height as shown in figure 1.7. The initial filtering stage is responsible for fault isolation by addressing the known shortcomings of the two sensors. The pressure altimeter measurements are prone to fluctuations with sudden pressure changes due to wind gusts, while the ultrasonic sensor will occasionally register no measurement. Their architecture was demonstrated on measurement data from the two sensors as illustrated in figure 1.8.

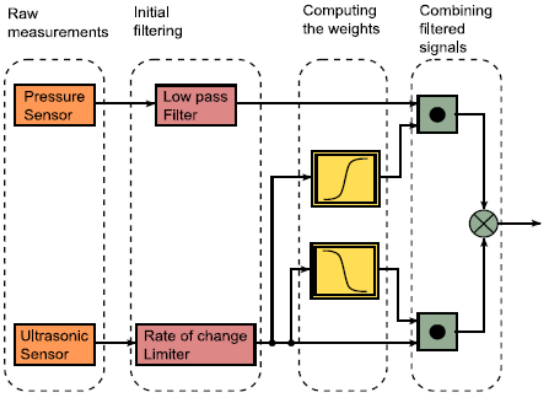


Figure 1.7: The sensor fusion architecture for relative altitude estimation of a UAV [1].

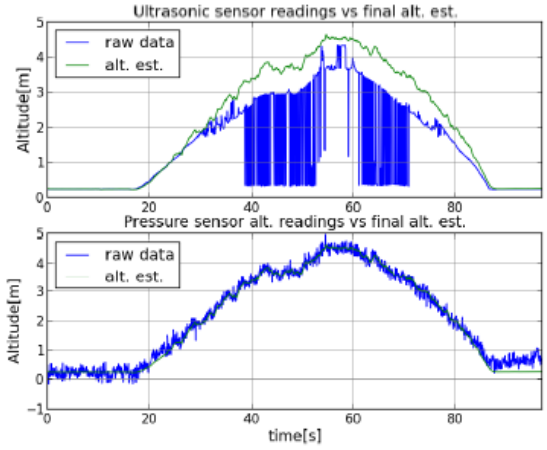


Figure 1.8: The raw measurements and estimated relative altitude versus time. The results for ultrasonic sensor is shown above and the pressure altimeter below [1].

The use of sensor fusion to accurately estimate aircraft states was proposed by Schettini et al. [6] who use a Kalman filter to fuse measurements from the global positioning system (GPS) receivers, the inertial sensors, and the air data system. The assumption is made that all critical sensors required are redundant and that fault detection and isolation is performed at a sensor level [6]. Their sensor fusion architecture uses an extended and adaptive Kalman filter that firstly linearises around the current operating point before adapting the sensor measurement noise based on flight conditions.

The implementation of an aerodynamic upset detection system that uses machine learning algorithms to classify dissimilar sensor measurements is investigated by Malan [7]. His system identified different types of aerodynamic upsets through the use of separate classifiers that were each trained to identify a specific upset. The techniques used for upset detection can also be used for fault detection and isolation.

1.5 Related Research

Thomas et al. [2] investigated the topic of sensor fusion for relative altimetry with the goal of improving the height estimate of a commercial passenger aircraft. Their primary objective was to remove the undesired events relating to misbehavior of the radio altimeters. Classical solutions for fusion of radio altimeter measurements employ hardware redundancy, and thresholding or voting logic to determine the validity of a sensor [2]. Unfortunately these approaches cannot handle multiple failures.

The approach proposed by Thomas et al. establishes a hybrid model for each individual sensor that consists of two separate states, namely a discrete and continuous state. The discrete state reflects the current operating mode of the sensor, for example "nominal", "biased", "frozen" or "non computed data." The continuous state is estimated by a Bank of Kalman filters, where the design of an individual Kalman filter is based on a different operating mode. In other words, the model of the system dynamics for the individual Kalman filters will vary based on the fault or hypothesis being modelled. A probabilistic approach is taken to the decision making process, where a likelihood function is established that uses the measurement residual and its covariance matrix to determine the probability associated with each operating mode. The discrete state is updated to reflect the operating mode that maximises the likelihood function. A Hybrid Gaussian Mixture Filter (HGMF) is used to estimate the most likely aircraft height and respective operating modes for each sensor involved, as seen in figure 1.9.

The performance of the hybrid models was demonstrated on a simulation setup consisting of two radio altimeters. Table 1.1 enumerates the sensor operating modes that were defined.

n	**Mode**	**Remark**
1	Nominal	No-fault
2	Biased	Bright spot
3	Fixed at 7.5 m	Locked on landing gear
4	Fixed at −1.8 m	Coupling between antennas
5	Frozen on last value	Damage to altimetry system

Table 1.1: Possible operating modes for radio altimeter.

The "coupling" fault occurs when there is water that covers the fuselage and "bright spot" when there is a mountain face or building that returns a stronger reflection than the

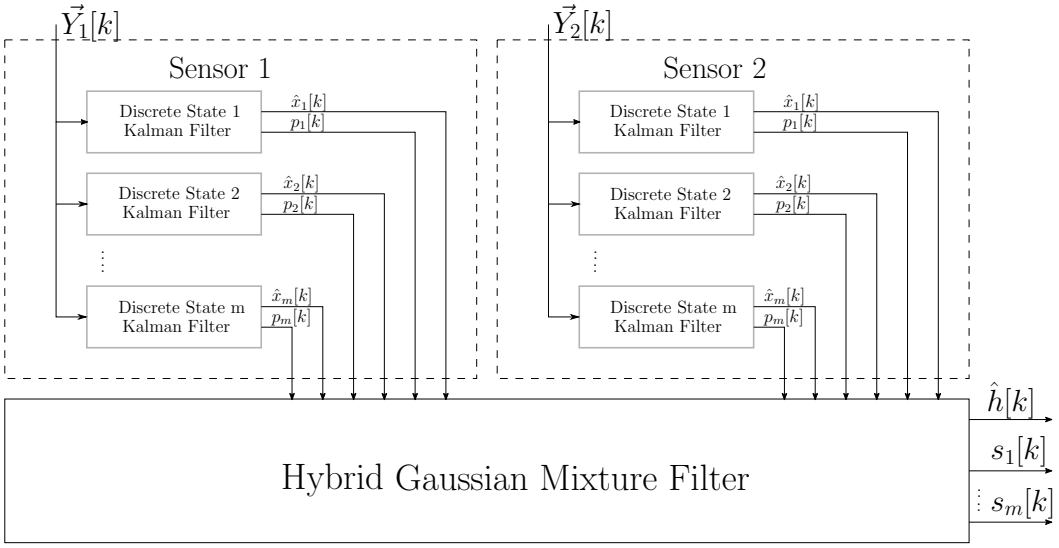


Figure 1.9: The interpreted sensor fusion relative altimetry architecture of Thomas et al. [2], where separate Banks of Kalman filters are established for each sensor. The Hybrid Gaussian Mixture Filter outputs the estimated aircraft height ($\hat{h}$) and the discrete status (s_i) for each sensor reflecting its operating mode.

terrain directly below the aircraft. Different simulation runs were used, where faults were injected onto one of the radio altimeter measurements. The estimated and true height were plotted as a function of time to demonstrate the system performance. The results showed that the proposed sensor fusion architecture is capable of detecting a "biased" fault, "fixed at $-1.8\,\mathrm{m}$" fault and "frozen on last value" fault. The sensor measurements and results for a "biased fault" are shown in figure 1.10 and figure 1.11 respectively.

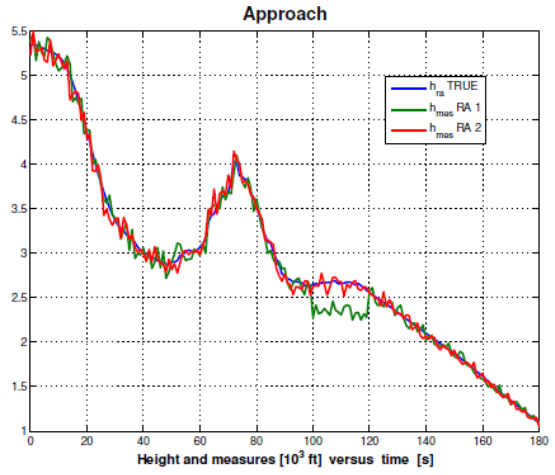


Figure 1.10: The true and measured height versus time for the sensors with a bias error on RA 1 [2].

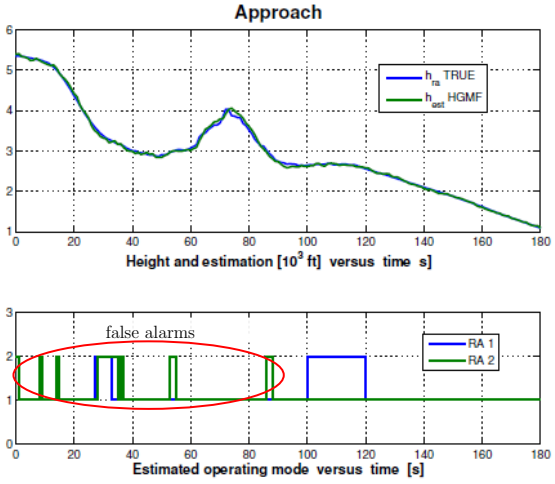


Figure 1.11: The true and estimated HGMF height versus time above. The estimated operating mode versus time is shown below [2].

A problem with the architecture proposed by Thomas et al, is the frequency of false alarms, as illustrated in Figure 1.7. (A value of 1 indicates the "Nominal" mode, while

a value of 2 indicates a "Biased" fault.) Their research is also limited in that they did not perform a quantitative analysis of their system's height estimate error, and the fact that their approach was not validated on actual sensor data. Also, their approach is only applied to multiple hardware-redundant radio altimeters, and will therefore not be robust to a failure that affects the entire radio altimetry system. (However, they do suggest incorporating additional independent sensor technologies.)

1.6 Research Goal

The purpose of this project is to develop a fault-tolerant sensor fusion system that provides robust height estimates for commercial airliners. It should deliver an aircraft height estimate that is robust to the failure of at most one of the height or altitude sensors: the radio altimeter, the GPS, the inertial sensors, and the instrument landing system (ILS).

1.7 Project Approach

The robust height estimation problem is divided into two major sub-problems: *fault detection and isolation* and *sensor fusion*. To address the fault detection and isolation sub-problem, a variety of analytical redundancy approaches were investigated. These included data-driven and model-based techniques as illustrated in figure 1.12. Two general approaches were considered: fault diagnosis using sensor measurements from a single time instant, and fault diagnosis using sensor measurements from a window of consecutive time instants.

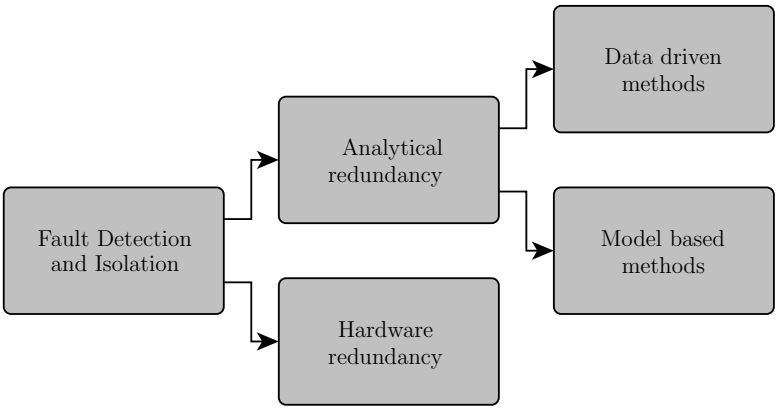


Figure 1.12: Overview of the approaches to fault detection and isolation.

For the single time instant approach, several data-driven techniques were applied, including outlier detectors, binary classifiers, and multi-class classifiers. The outlier detectors that were used include Elliptic Envelope, Local Outlier Factor and Isolation Forest. The binary and multi-class classifiers investigated include Logistic Regression, Naïve Bayes, k-Nearest Neighbors, Decision Tree and Support Vector Machine. Data preprocessing was used with the aim of improving class separability and classifier performance. Different data preprocessing techniques were investigated, including no preprocessing, the orthogonal transform, and the kernel transform. A custom preprocessing technique (called the residual transform) was also proposed, prompted by the height verification problem. The classifiers were trained on equal percentages of each possible operating mode to ensure

unbiased classification. All fault detection and isolation approaches were evaluated and compared according to the percentage of faults that were correctly detected and operating modes that were correctly identified. The locations of all incorrect operating mode classifications were analysed. The approach was validated on actual measurement data and the reported versus actual operating mode was plotted as a function of time.

Both model-based and data-driven techniques were applied for consecutive time instant fault detection and isolation. The model-based techniques include the Bank of Kalman filters and the Robust Kalman filter, while the data-driven techniques include Model Consensus (a variant of random sample consensus), Dynamic Principal Component Analysis, and the best performing single time instant classifier with time as an additional variable. Similar to the single time instant approach, the consecutive time instant techniques listed above were evaluated and compared according to fault detection and isolation accuracy, and the locations of all incorrect classifications were analysed. After considering a wide variety of approaches and techniques, the best performing fault detection and isolation techniques were identified.

Finally, to address the sensor fusion sub-problem, three methods were investigated: using the median of the sensor measurements, using the weighted average of the sensor measurements, and using a Kalman filter to perform optimal, fault-tolerant sensor fusion. The sensor fusion methods were compared by analysing the height estimation error.

A simulation model was used to generate synthetic training and testing data. Mathematical models were established for the aircraft motion, the sensors, and the terrain. The structure and nominal parameters for the sensor models were based on information sourced from literature, and then the sensor parameters were tuned to fit a real dataset provided by Airbus. Fault models for six types of sensor faults were also created. The simulation model was used to generate a large dataset of representative sensor measurements containing both “no fault” and “fault” conditions. The fault detection and isolation, and the sensor fusion were tested using both simulated data and real datasets of actual flight data with synthetic sensor failures injected.

1.8 Project Objectives

The research objectives for this project were formulated as follows:

- Establish a simulation model that can be used to generate representative relative altitude and height data with the aid of the on-board terrain database and detailed terrain map. This includes the following secondary objectives:
  - Establish a model for the aircraft motion.
  - Derive realistic sensor models from a combination of literature and actual data for the following aircraft sensors: global positioning system (GPS) receiver, inertial reference system (IRS) sensors, instrument landing system (ILS), and radio altimeter (RA).
  - Generate faulty sensor measurement data by injecting any of the following fault profiles onto sensor measurements: bias, jamming, oscillation, Non-Return-to-Zero, increased noise, or runaway.

- Investigate and apply fault detection and isolation algorithms. This includes the following secondary objectives:
  - Perform a literature review of a wide variety of approaches to fault detection and isolation. Fields explored should be inclusive of outlier detection methods, supervised machine learning techniques and classical model-based approaches.
  - Evaluate and compare the performance of the most promising fault detection and isolation approaches with respect to fault detection and isolation accuracy as well as analyzing the location of the incorrect fault predictions. There should be no false alarms from no-fault sensor measurements as these decrease system reliability and would result in additional stress for the pilot.
  - Design a fault detection and isolation system for robust height estimation using the best performing model-based or data-driven technique.
- Investigate and apply sensor fusion algorithms that estimate the height of a commercial aircraft. This includes the following secondary objectives:
  - Investigate and establish methods by which the altitude sensors can be used to validate the height measurement.
  - Investigate a variety of techniques to ensure that the estimated height of a commercial aircraft is robust to sensor failure.
- Validate and demonstrate the performance of robust height estimation architectures on actual measurement data.

1.9 Thesis Overview

Chapter 1 introduced and motivated the research problem of robust height estimation for commercial airliners, reviewed the relevant literature, formulated the research goal, identified the project objectives, and provided an overview of the project approach. Chapter 2 conceptualises the problem of fault-tolerant sensor fusion for robust height estimation, and presents the mathematical modelling of the aircraft motion, the terrain, the height and altitude sensors, and the sensor faults. Chapter 3 presents background theory on fault detection and isolation, and provides an overview of various model-based and data-driven techniques that are used for fault detection and isolation. Chapter 4 presents the design of a fault detection and isolation architecture that serves as the framework for the model-based and data-driven techniques. Two general approaches were considered for fault diagnosis, namely using sensor measurements from a single time instant, and using measurements from a window of consecutive time instants. Chapter 5 details the design and evaluation of fault detection and isolation techniques that use measurements from a single time instant. Similarly Chapter 6 details the design and evaluation of fault detection and isolation techniques that use a window of measurements from consecutive time instants. Chapter 7 investigates the application and evaluation of different sensor fusion techniques. Chapter 8 presents a summary of the work done, discusses the conclusions, and gives recommendations for future research in robust height estimation.

Chapter 2

Conceptualisation and Modelling

In this chapter, the robust height estimation problem is conceptualised, and mathematical models are established for the aircraft motion, the sensors, and the terrain. Fault models for six types of sensor faults are also defined. The mathematical models serve as the basis for the simulation model that is used to generate synthetic training and testing data for the robust height estimation techniques.

This chapter will initially conceptualise the simulation set-up, and problem of robust height estimation. An overview of the simulation model is then presented and explained. The aircraft and terrain models that are used to establish true relative altitude and height data are presented first. Thereafter the structure and nominal parameters for the sensor models are sourced from literature. The parameter values are tuned to fit actual data. Lastly the simulation model is expanded to account for sensor faults.

2.1 Conceptualisation

This section introduces the frame of reference, terminology and notation that will be used to differentiate between the respective height and relative altitude sensors. This is necessary information for the reader to better conceptualise the problem of fault-tolerant sensor fusion for aircraft height estimation.

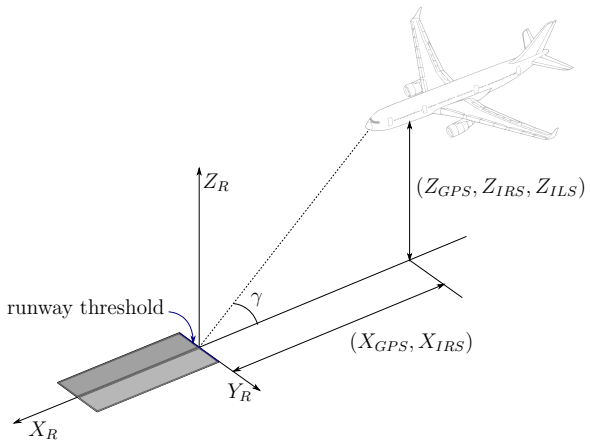


Figure 2.1: Illustration of the standard notation and conceptualisation for the relative altitude sensors

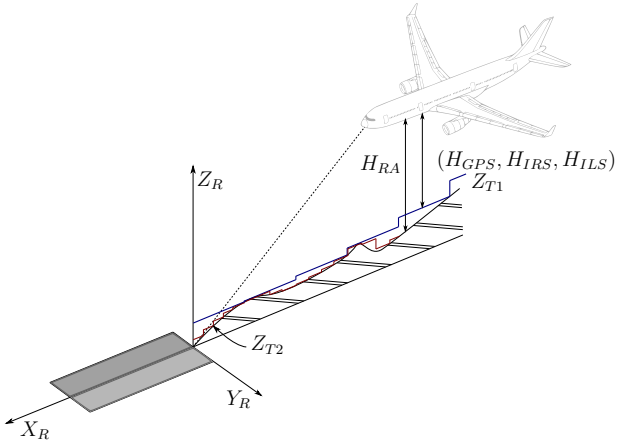


Figure 2.2: Illustration of the standard notation and conceptualisation for the height sensor

The runway axis system (X_R, Y_R, Z_R) serves as the reference frame for the position and velocity of the aircraft as illustrated in figure 2.1 and figure 2.2. The runway axis system assumes a flat earth approximation. The X_R axis is in the horizontal plane, and is aligned along the centre of the runway such that the heading of an approaching plane is in the positive X_R direction. The Z_R axis is perpendicular to X_R axis and is defined as pointing in a positive upward direction when using the flat earth approximation. The zero reference point for the runway axis system is the runway threshold which is defined as the start of the runway for an aircraft on approach to land. Lastly the Y_R axis is perpendicular to both the X_R and Z_R axes, but is not considered in this project.

Relative altitude is defined as the elevation of an aircraft above the runway it is moving towards upon final approach to land. The relative altitude sensors include the global positioning system (GPS), inertial reference system (IRS) and instrument landing system (ILS) as shown in figure 2.1. The GPS sensor makes use of satellite navigation to determine the aircraft location. The IRS sensor propagates the aircraft position using the measurements from the accelerometers and gyroscopes strapped to the aircraft body axes. The ILS sensor employs trigonometry to determine the relative altitude of the aircraft using the glide slope measurement (γ) and aircraft location along the X_R axis, given by X_{GPS} or X_{IRS}.

Height is defined as the elevation of the aircraft above the ground directly below it. The aircraft height is a function of the aircraft motion and the terrain below it as illustrated in figure 2.2. The only height sensor is the radio altimeter (RA). The radio altimeter is a radar that measures the height of an aircraft above the surrounding terrain. The terrain height is given by database Z_{T2} that gives the maximum height of the terrain within 100 m segments along the X_R axes. The relative altitude sensor measurements can be converted to the height domain through the use of the terrain database Z_{T1}. This database is stored in the onboard flight computer and gives the maximum height of the terrain within 500 m segments along the X_R axes.

The necessary notation that will be used to refer to databases, height and altitude sensors is summarised as follows:

$X_{<>}$,$Z_{<>}$	The co-ordinates of the aircraft in the runway axis system as returned by sensor <>
$H_{<>}$	The height of the aircraft as returned by sensor <>
$\dot{X}_{<>}$,$\dot{Z}_{<>}$	The velocity of the aircraft in runway axis system as returned by sensor <>
Z_{T1}	The relative altitude of the terrain directly below the aircraft as returned by the onboard terrain map in runway axes
Z_{T2}	The true relative altitude of the terrain directly below the aircraft in runway axes
γ	Glide slope angle followed by an aircraft on approach to landing

2.2 Overview of Simulation Model

This project required the development of simulation models for the aircraft height and altitude sensors. These are necessary to generate representative measurement data for comprehensive training and testing purposes.

Three essential components were necessary to achieve this, namely a model of the aircraft dynamics, a model of the terrain below the aircraft, and lastly mathematical models for the respective sensors. The aircraft model is used to generate true relative altitude and height data. The true aircraft height is given by the difference between the terrain model and true relative altitude of the aircraft. The mathematical models for the four different sensor technology types use the true height and relative altitude data to generate representative sensor measurements based on known sensors accuracies. Each sensor has small measurement errors that contribute towards the effective accuracy associated with that sensor. The terrain height and sensor measurements will be used by the fault tolerant sensor fusion methods to provide a robust height estimate $\hat{h}$. An overview of the simulation model is illustrated in figure 2.3.

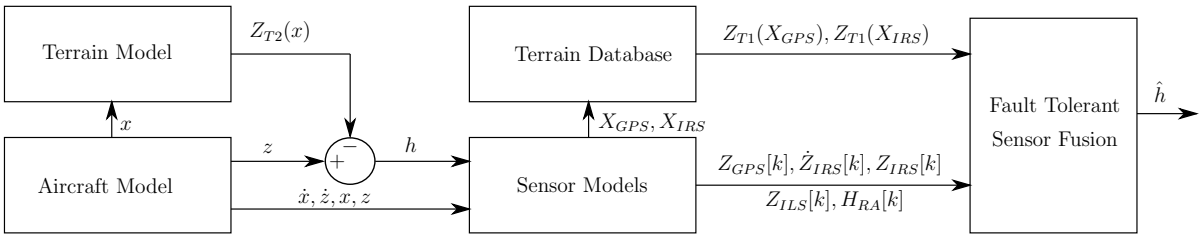


Figure 2.3: Overview of the simulation model used to generate sensor data

The aircraft simulations were initialised at the point of intersection with the $\gamma = 3°$ glide slope that the aircraft follows along its final approach until touch down when the simulation ends. The true X_R location of the aircraft, x, is used by the accurate terrain database, Z_{T2}, to determine the true aircraft height, h. The input data required by the sensor models consists of the x and z position of the aircraft, the $\dot{x}$ and $\dot{z}$ velocity of the aircraft, and lastly the height of the aircraft, h.

2.3 Aircraft Model

The deterministic forces and moments that act on an aircraft can be divided into three groups, namely aerodynamic, gravitational and propulsion [8]. The aerodynamic forces and moments are attributed to the flow of air around the aircraft fuselage. The gravitational force acts on the aircraft in a down direction, while the location and engine type will determine the resulting propulsion forces and moments. The individual components from each of these groups are combined to give the total forces and moments acting along the aircrafts body axes. The mathematical model for an aircraft assumes a six degree of freedom rigid body with non-linear translational and rotational forces acting on it as illustrated in figure 2.4. The control inputs allow for commands that adjust the level of thrust for propulsion systems as well as setting the aileron, elevator and rudder deflections.

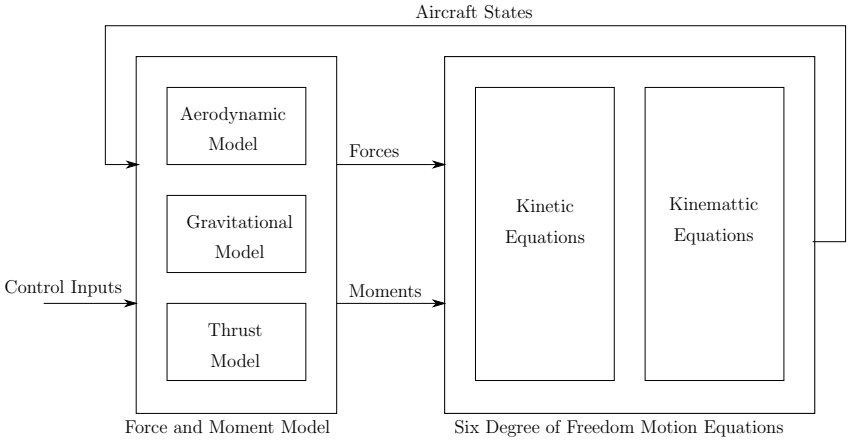


Figure 2.4: Block diagram of Aircraft Model [9]

While the model above allows for accurate representation of aircraft dynamics it was deemed unnecessary for the simulation of height and altitude data. Instead, a reduced-order model of the aircraft's point mass translational kinematics was used to generate simulated position and velocity profiles for an aircraft on approach. This reduce-order model reduces the computational burden and allows for the simulation of large quantities of realistic height and altitude measurement data, where the effects of pilot manual control and environmental disturbances such as wind gusts are more easily exaggerated.

The model assumes that the longitudinal and lateral motion of the aircraft are decoupled. The lateral movements are not considered as they are assumed to have a negligible effect on the height and relative altitude measurements. The initialization boundaries used for the simulation model are a maximum relative altitude of $z_{\max} = 1194\,\text{m}$ and horizontal position of $x_{\max} = -22\,500\,\text{m}$ in the runway axis system. This corresponds to the maximum possible relative altitude for a glide slope of $\gamma = 3°$ and terrain database that is $22\,500\,\text{m}$ long. The relative altitude motion of the aircraft is modelled with the following equations:

$$\dot{z}(t) = \eta_z(t) + \dot{z}(0) \tag{2.3.1}$$

$$z(t) = \int_0^t \dot{z}(t)dt + z(0) \tag{2.3.2}$$

where $\dot{z}(0)$ is the initial climb rate, $z(0)$ is the initial relative altitude of the aircraft and $\eta_z(t)$ is Gaussian noise that imposes a random walk on the initial climb rate to simulate the effect of wind gusts and pilot manual control.

The horizontal translational motion of the aircraft relative to the runway threshold is modelled with the following equations:

$$\dot{x}(t) = \eta_x(t) + \dot{x}(0) \tag{2.3.3}$$

$$x(t) = \int_0^t \dot{x}(t)dt + x(0) \tag{2.3.4}$$

where $x(0)$ is initial X_R aircraft position, $\dot{x}(0)$ is the X_R aircraft velocity and $\eta_x(t)$ is Gaussian white noise to simulate the effect of wind gusts and pilot manual control.

The velocity vector of the aircraft model was initialised such that the aircraft approaches the runway along the glide slope $\gamma = 3°$ as illustrated in figure 2.5. The random noise components η_z and η_x result in small variations of the aircraft position around the glide slope. Note that the horizontal position (x) of the aircraft will have a negative value on the X_R runway axis at the start of the approach, and will travel with a positive horizontal velocity towards the runway threshold (x=0).

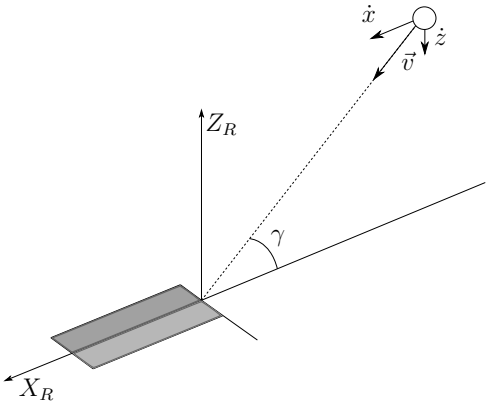


Figure 2.5: The breakdown of the velocity vector $\vec{v}$ along the X_R and Z_R runway axis system to ensure that the aircraft approaches the runway along the glide slope γ

2.4 Terrain Model and Terrain Database

The terrain model gives the relative altitude in the runway axis system of a bare earth surface for 22.5 km leading up to the runway threshold. Airbus provided two terrain databases for a specific runway at Toulouse Airport in France. The first database Z_{T1} is more coarse with a resolution in the X_R axis of 500 m. The database functions as a lookup table, where the value for a 500 m segment corresponds to the highest point of terrain within that segment. The second database Z_{T2} is a more accurate database with a resolution in the X_R axis of 100 m. This database was used as the terrain model and functions as the ground truth. The true height of the aircraft h is difference between relative altitude and the lookup value from Z_{T2}. Figure 2.6 plots the two terrain databases as a function of the aircraft's X_R position.

An aircraft will not land at the runway threshold as this is the start of the runway, therefore consideration needs to be made for the terrain map beyond the runway threshold. The assumption was made that the runway at the airport is flat and therefore the terrain model is zero for all $x > 0$. The lookup value for the relative altitude of the terrain $Z_{T1}(x)$ is given by:

$$Z_{T1}(x) \leftarrow \begin{cases} 0 & \text{for } x > 0 \\ Z_{T1}(x) & \text{for x} \leq 0 \end{cases} \tag{2.4.1}$$

where x is the aircraft location along the X_R axis.

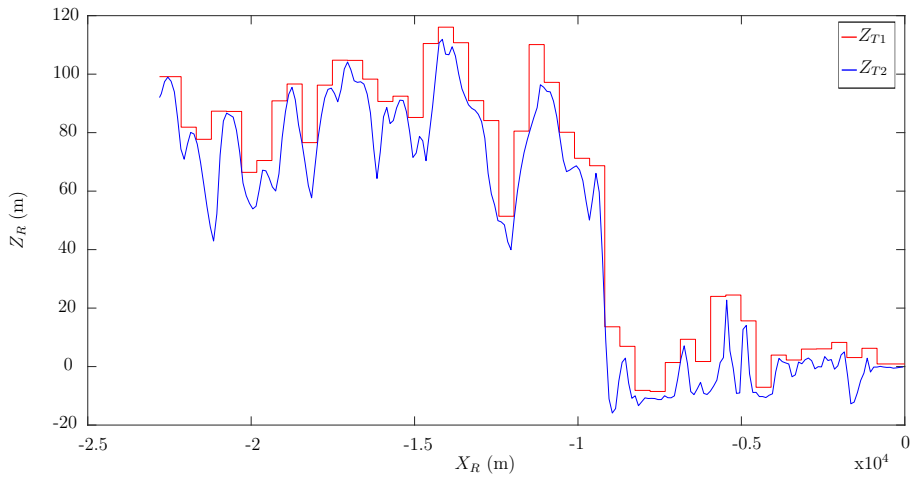


Figure 2.6: The terrain databases supplied by Airbus. The blue database has a resolution of 100 m, while the red database has a resolution of 500 m

The terrain database accuracy characteristics were derived from the difference between $Z_{T2}(x)$ and $Z_{T1}(x)$. The mean error and corresponding standard deviation are given by:

$$\mu_{\text{terr}} = 10.22\,\text{m} \tag{2.4.2}$$

$$\sigma_{\text{terr}} = 9.16\,\text{m} \tag{2.4.3}$$

where μ_{terr} and σ_{terr} are the mean and standard deviation terrain database error respectively.

2.5 Sensor Models

2.5.1 Introduction to Sensor Modeling

The height and altitude sensors use different technologies and methods whose performance are subject to a variety of factors such atmospheric conditions, equipment age, and topography. These factors result in errors between the actual and measured aircraft position. There are many different sources of error that affect a sensor, including bias, noise, quantization, drift, bandwidth, misalignment, and scale factor. Some of these error sources are illustrated in figures 2.7-2.10. The true signal is represented by the black line, while the error has been exaggerated in blue.

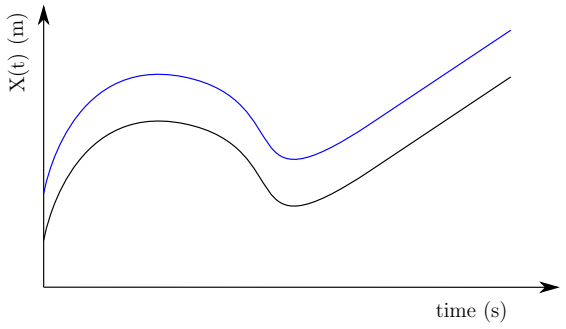


Figure 2.7: Example bias error imposes a constant offset error

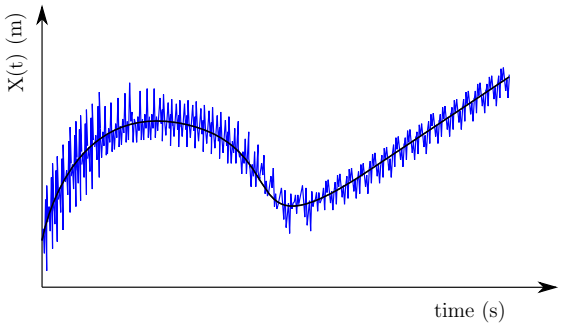


Figure 2.8: Example noise error results in additive white noise

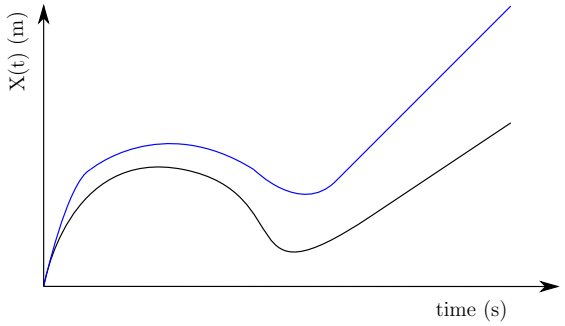


Figure 2.9: Example drift error that increases with time

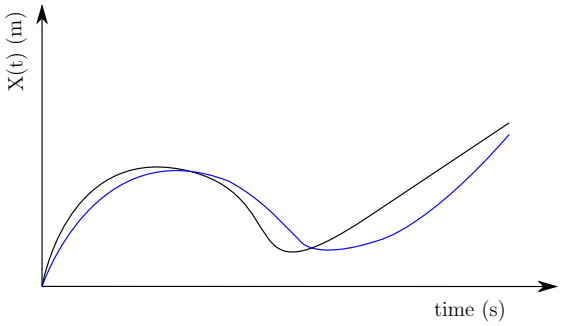


Figure 2.10: Example bandwidth error that results in a delayed response to change

A combination of stochastic variables and processes were used to model the different error components. The discrete-time equation below shows how a realistic measurement (X_{meas}) is modelled given the truth (X_{true}):

$$X_{meas}[k] = X_{true}[k] + b + \eta[k] + \Delta_{drift} \cdot kT_s + W[k] \tag{2.5.1}$$

where b is magnitude of the bias, $\eta[k]$ is zero mean discrete-time white noise, T_s is the sampling period, Δ_{drift} is the rate of drift and $W[k]$ is a stochastic process.

Two different stochastic processes were also considered for modelling, namely random walk and a first-order Gauss Markov process. A random walk model integrates white noise to produce an output that will slowly "walk" or vary around some initial value. A Gauss-Markov model is an extension of random walk that allows for the modelling of the signals that are a mixture of random walk and noise.

Random Walk

Random walk is a simple and effective sensor error model, where the varying noise signal is obtained by integrating discrete-time samples. The discrete-time model for a continuous random walk process is given by:

$$W[k] = W[k-1] + \eta[k] \tag{2.5.2}$$

The noise component is usually modelled by a zero mean Gaussian white noise signal. The variance of a random walk process is non-stationary as it increases over time. The relation between the variance of the random process noise (σ_{wk}) and the discrete-time white noise ($\sigma_{\eta k}$) is given by $\sigma^2_{wk} = \sigma^2_{\eta k} \cdot kT_s$ [4]. An example of a random walk signal is illustrated in figure 2.11. An extension of random walk is to add a drift component that results in the expected value of the process increasing at a fixed rate. The random walk with drift model is given by:

$$W[k] = W[k-1] + \Delta_{drift} + \eta[k] \tag{2.5.3}$$

The resultant continuous noise signal will follow a deterministic trend given by ($kT_s \cdot \Delta_{drift}$ + $W[0]$) with a stochastic trend superimposed on the drift component as illustrated in figure 2.12.

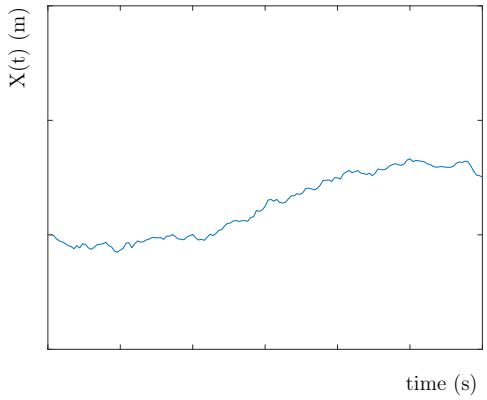


Figure 2.11: Example random walk stochastic process

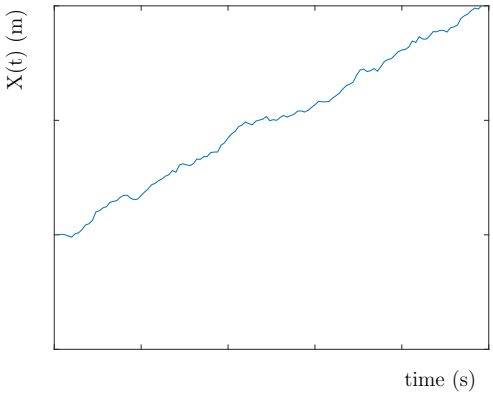


Figure 2.12: Example random walk with drift stochastic process

First Order Gauss-Markov

The first-order Gauss-Markov (GM) process is a flexible random error model as it can represent a large number of systems with relatively simple mathematical equations. The equation for a discrete-time first-order GM system is given by:

$$W[k] = e^{-T_s/T_c} W[k-1] + \eta[k] \tag{2.5.4}$$

where $W[k]$ is a discrete-time stochastic process with correlation time T_c [4].

A first-order GM model is advantageous as it can model a variety of stochastic processes that vary between random walk and Gaussian white noise. As $e^{-T_s/T_c} \rightarrow 1$ the stochastic process becomes more representative of random walk and likewise as $e^{-T_s/T_c} \rightarrow 0$ it becomes more representative of white noise as illustrated in figure 2.13-2.14. The time constant determines the amount of correlation in the resulting noise. A larger time constant, $T_c >> T_s$, results in a stochastic process that is more correlated and less noisy. The discrete-time noise variance is given by:

$$\sigma_\eta^2 = \sigma_w^2 (1 - e^{-2T_s/T_c}) \tag{2.5.5}$$

where σ_w^2 is the variance of the GM process and σ_η^2 in the variance of the white noise component.

Both these example stochastic processes started at an initial value of $\mathrm{X}(t = 0\,\mathrm{s}) = 5$. The effect of e^{-T_s/T_c} being a fraction in the range of zero to one is that the first-order GM stochastic process will be centered around zero as seen in figures 2.13-2.14.

Sensor Accuracy

Now that the error models used to establish realistic measurements have been discussed it is necessary to define what is meant by sensor accuracy. It is important that sensor accuracy is well defined as it determines the boundary between what will be considered a faulty and no-fault measurement. Each sensor has an expected accuracy defined as the maximum acceptable difference between an actual and measured value. Differences beyond this range are considered to be faults. This definition is clear when sensor accuracy is stated as the absolute maximum error bounds, but obscure for sensor accuracies

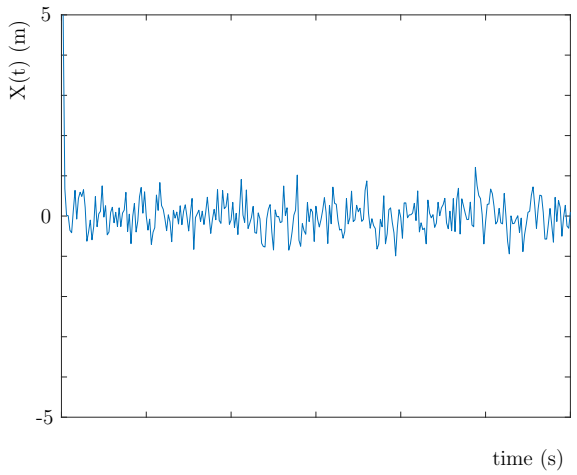


Figure 2.13: Example first-order GM stochastic process where $e^{-T_s/T_c} = 0.1$ that resembles white noise

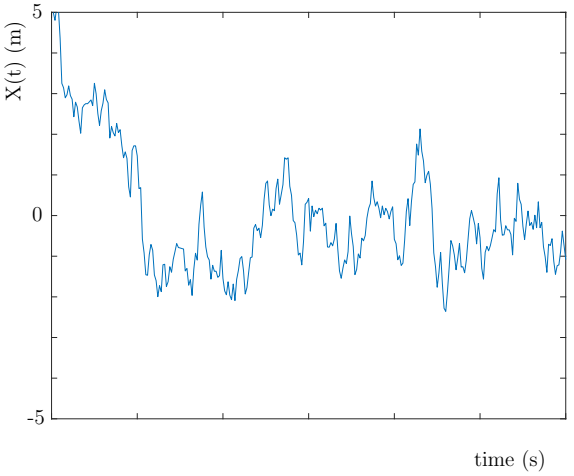


Figure 2.14: Example first-order GM stochastic process where $e^{-T_s/T_c} = 0.9$ that resembles random walk

quantified as a standard deviation value. The need therefore exists to define a boundary or threshold between what will be considered a fault and no-fault measurement for a sensor accuracy stated as a standard deviation. The John Tukey fence is a method from literature that provides a quantifiable definition for an outlier. Faults can be viewed as outliers as they are dissimilar to other sensor measurements. The following lower and upper bounds for classifying measurements as inliers or outliers are proposed by Adil et al. [10]:

$$\text{lower bound} = Q_1 - k \cdot IQR \tag{2.5.6}$$

$$\text{upper bound} = Q_3 + k \cdot IQR \tag{2.5.7}$$

where Q_1 is the first quartile, k is a non-negative constant, Q_3 the third quartile and IQR the interquartile range (Q_3-Q_1).

Measurements that fall between the boundaries are classified as inliers and are considered no-fault measurements. Measurements that fall outside the boundaries are classified as outliers and are considered faulty measurements. The Tukey fence defines a threshold for both mild (k=1.5) and extreme outliers (k=3) [10]. The first quartile for a zero mean normal distribution is equal to -0.6745σ and the third quartile to 0.6745σ. The IQR is given by their difference $Q_3 - Q_1$. The threshold for a mild sensor fault was used and is given by:

$$\text{lower bound} = -2.6985\sigma \text{ AND upper bound} = +2.6985\sigma \tag{2.5.8}$$

where σ is the standard deviation.

According to this definition for sensor accuracy, 99.65 % of actual no-fault sensor measurements would be classified as no-fault when using a normal distribution.

2.5.2 Overview of Sensor Models

An overview of each sensor's inputs and outputs are illustrated in the figure below.

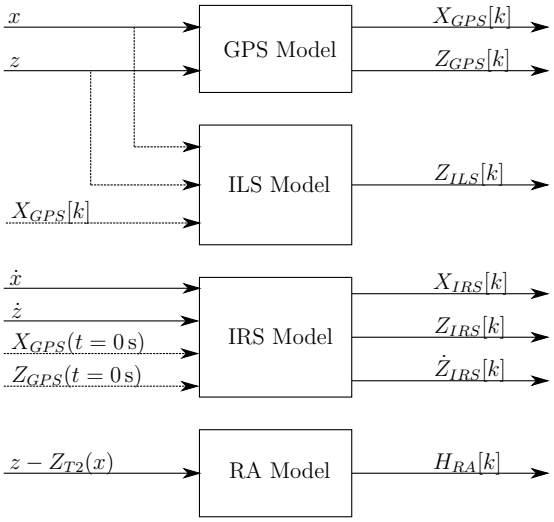


Figure 2.15: An overview of the inputs and outputs for each sensor model.

The remainder of this section is dedicated to the modelling of the sensors listed in figure 2.15 above. Firstly, the necessary knowledge and theory to understand the sensor operation is presented. Thereafter, a literature survey of existing sensor error models is conducted, before fine tuning model parameter values to fit a real dataset provided by Airbus. Next, the theoretical accuracy associated with a specific sensor is derived. This is used to differentiate between fault and no-fault measurement data. Lastly, the sensor model is validated by plotting the theoretical accuracy that is a function of relative altitude against the actual measurement data. The sensor measurements supplied by Airbus should all be situated within the no-fault region. This allows for an objective comparison of how well the sensor model characteristics fit the actual measurement data.

2.5.3 GPS Sensor Model

2.5.3.1 Introduction

The Global Positioning System (GPS) was developed by the United States Department of Defense as an all-weather, space-based navigation system that could accurately determine the position, velocity and time from a fixed reference system anywhere on or near earth [11]. It should be noted that GPS is not the only satellite based navigation system. Global Navigation Satellite Systems (GNSS) is the umbrella term used when referring to a group of space-based navigation systems that includes GPS, GLONASS, Galileo and BeiDou. These independent systems are all backed by different governments or world powers. Galileo is supported by the European Union, while GLONASS is operated by the Ministry of Defence Force of the Russia, and BeiDou by the Chinese National Space Administration. All these systems work on similar principles and for all intents and purposes the satellite navigation system that will be investigated is GPS.

GPS comprises three separate components, namely a space, control and user segment

[3]. The space segment refers to the satellites orbiting the earth that transmit radio signals from space. The control segment is the network of tracking stations located around the world that track and monitor the satellite's health and status. These tracking stations provide the master control station with information on the satellite's orbit and atomic drift, so that corrections to positional data can be relayed to the satellites. The user segment is the product or interface that is fitted with the GPS receiver. In an aircraft the GPS receiver uses the time data of the satellite's atomic clocks that it received to calculate the distance the aircraft is located from the satellite at that instant. Triangulation between 4 or more satellites can be used to accurately determine the three-dimensional coordinates of the aircraft.

The GPS receiver provides the aircraft's position as a latitude, longitude, and altitude, where the altitude is given with respect to mean sea level.

2.5.3.2 Literature on GPS Sensor Models

The GPS model described by Beard and McLain [3] incorporates the main sources of error into a single stochastic model. Beard and McLain consider two different categories of GPS error that have a significant effect on the accuracy of GPS measurements. The first category involves errors relating to the estimation of the pseudorange, while the second category addresses the quality of the geometric constellation of the satellites.

The time of flight of the radio waves is used to calculate the distance between a satellite and the GPS receiver. Due to synchronization errors between the satellite and receiver clocks the perceived distance differs from the actual geometric distance. For example a timing error as small as 10 ns results in a position error of 3 m [3]. The perceived distance is called the pseudorange. There are a few sources of error that arise for the pseudorange estimation, namely ephemeris data, satellite clock accuracy, atmospheric conditions, multipath signals and receiver measurement errors. Their collective effect is known as the User Equivalent Range Error (UERE). Ephemeris data is orbital information that is used to determine the location of the satellite. Uncertainties in the satellite's position will lead to errors in the pseudorange calculation. The satellite clock has drift in its atomic clocks that produce a deviation from the true time of flight used to calculate the pseudorange. The largest source of error is due to the role that the earth's atmosphere plays in the propagation of the GPS signals, particularly in the Ionosphere and Troposphere. The Ionosphere is characterised by the presence of free electrons that result in a delay in signal propagation. The Troposphere is the lowest layer of the earth's atmosphere, where temperature, humidity and pressure affect the time of flight, leading to errors in the pseudorange calculation. Multipath errors are those that arise from the reflection of signals off large surfaces located near the receiver that could mask the intended signal. The last error source considered is receiver measurement error due to the limitations with which the timing of the satellite can be resolved.

Table 2.1 gives typical standard deviation values for the respective error sources. Considering both the bias and the random noise components as independent random variables that are modelled by zero mean normal distributions, the total error is given by the sum of the two. The theoretical sum of two independent normally distributed random variables

is given by:

$$X \sim \mathcal{N}(\mu_x, {\sigma_x}^2) \text{ and } Y \sim \mathcal{N}(\mu_y, {\sigma_y}^2) \tag{2.5.9}$$

$$Z = X + Y = \mathcal{N}(\mu_x + \mu_y, {\sigma_x}^2 + {\sigma_y}^2) \tag{2.5.10}$$

Therefore the standard deviation of the total error, σ_T, is give by:

$$\sigma_T = \sqrt{\sigma_b^2 + \sigma_\eta^2} \tag{2.5.11}$$

where σ_b is the standard deviation of the bias, and σ_η is the standard deviation of the random noise.

The bias error, random noise error, and total error for each error source are combined to produce the User Equivalent Range Error.

Error Source	Bias (σ_b)	Random Noise (σ_η)	Total Error (σ_T)
Ephemeris	2.1	0	2.1
Satellite clock	2.0	0.7	2.1
Ionosphere	4.0	0.5	4.0
Troposphere	0.5	0.5	0.7
Multipath	1	1	1.4
Receiver Measurement	0.5	0.2	0.5
UERE(rms)	5.1	1.4	5.3

Table 2.1: Typical pseudorange standard deviation values in metres [3].

The second category of errors affecting the accuracy of a GPS sensor is the quality of the geometric constellation. This resultant error is represented by a numeric value called Dilution of Precision (DOP) and is a measure of the quality of the satellite constellation that determines the GPS position. A large DOP is an indication of a poor constellation, where direct line of sight with one or more satellites could be blocked by obstructions. This will result in larger errors for the pseudorange estimate. A low DOP is an indication of a desirable constellation. There are two types of DOP, namely VDOP (vertical DOP) and HDOP (horizontal DOP). VDOP and HDOP differentiate between GPS accuracy in the vertical and horizontal plane respectively. The nominal VDOP value is given by 1.8 m and HDOP by 1.3 m [3]. The total vertical and horizontal RMS error is given by [12]:

$$\text{Vetical error (RMS)} = VDOP \cdot \sigma_{UERE} \tag{2.5.12}$$

$$= (1.8)(5.1)m \tag{2.5.13}$$

$$= 9.2\,\text{m} \tag{2.5.14}$$

$$\text{Horizontal error (RMS)} = HDOP \cdot \sigma_{UERE} \tag{2.5.15}$$

$$= (1.3)(5.1)m \tag{2.5.16}$$

$$= 6.6\,\text{m} \tag{2.5.17}$$

where σ_{UERE} is the standard deviation of the User Equivalent Range Error.

A first-order Gauss-Markov process is used to model the GPS error. The parameter values given in table 2.2 are used in conjunction with equation 2.5.4. The Gauss Markov process is initialised by drawing a random bias, $W(t = 0\,\text{s})$, from a Gaussian distribution $\mathcal{N}(0, \sigma_b)$m, where σ_b is either the vertical or horizontal RMS error. The resultant error model is a predominant bias error that varies slowly with time.

Direction	σ_b (m)	σ_w (m)	σ_η (m)	T_c (s)	T_s (s)
Altitude (Z)	9.2	0.7	0.4	1100	1.0
Horizontal (X)	4.7	0.4	0.21	1100	1.0

Table 2.2: Gauss Markov model parameters for GPS model [3].

An identical GPS model was used by Maier et al. [12] who attempted to improve GPS accuracy in urban areas where the effect of the multipath signal reflections are significantly greater. Rankin [13] derived a more detailed model than Beard and McLain by modelling individual pseudorange error sources with separate Gauss-Markov models. The time constant T_s/T_c for each error source is different to increase the accuracy around the model of each component. This model however is outdated as it dates back to 1994 when the effects of selective availability were still relevant. It also neglects the effects of the geometric constellation.

2.5.3.3 GPS Sensor Model

The model from literature was adapted to fit the theoretical sensor characteristics and measurement data supplied by Airbus and is illustrated in figure 2.16.

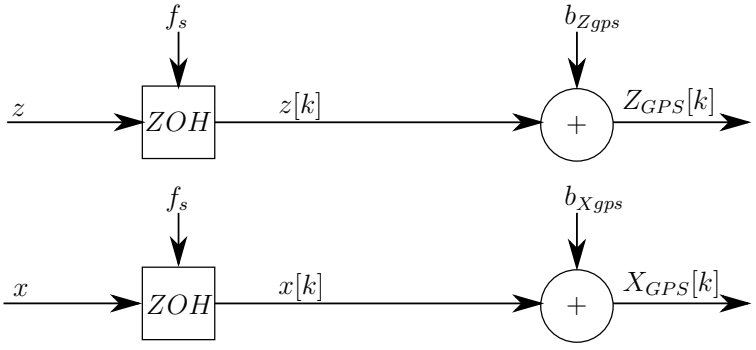


Figure 2.16: Stochastic GPS sensor model.

The GPS error model for a x and z measurement is modelled by a quasi-static random bias drawn from a uniform distribution. The term quasi-static describes an error that changes so slowly with time that it appears to be practically static over the time frame that it is observed. The motivation behind this decision was two-fold. Firstly, the random noise of the GPS error model was significantly smaller than that of the random bias, $\sigma_b >> \sigma_\eta$. Secondly, the random walk effect is negligible due to the fact that the aircraft landing is executed over a relatively short period of time. Also, the correlation time T_c is significantly larger than the sampling period T_s. The resultant error model is therefore

reduced to $W[k] = W(t = 0\,\text{s})$, which is a random bias drawn from a uniform distribution $\mathcal{U}(-\delta_{gps}, \delta_{gps})$. A uniform distribution is used because the GPS sensor error parameters supplied by Airbus were given in terms of error bounds. The X_{GPS} and Z_{GPS} measurements are expressed in runway axes. The mathematical model for GPS measurements is therefore given by:

$$Z_{GPS}[k] = z[k] + b_{Zgps} \tag{2.5.18}$$
$$X_{GPS}[k] = x[k] + b_{Xgps} \tag{2.5.19}$$

where b_{Zgps} and b_{Xgps} are the random quasi-static GPS biases along the Z_R and X_R axis respectively, $z[k]$ and $x[k]$ are the discrete-time aircraft position measurements sampled at frequency f_s.

The continuous aircraft position is sampled at a f_s =8 Hz to give the discrete-time aircraft position x_k and z_k. The quasi-static altitude bias is drawn from a uniform distribution with characteristics $b_{Zgps} \sim \mathcal{U}(-\delta_{Zgps}, \delta_{Zgps})$ at simulation time $t = 0\,\text{s}$. Similarly, the horizontal position bias is drawn from an uniform distribution with characteristics $b_{Zgps} \sim \mathcal{U}(-\delta_{Xgps}, \delta_{Xgps})$.

2.5.3.4 GPS Accuracy

The GPS accuracy remains constant as a function of altitude with parameter values:

$$\Delta_{Zgps} = \delta_{Zgps} = 1.6764\,\text{m} \tag{2.5.20}$$
$$\Delta_{Xgps} = \delta_{Xgps} = 15\,\text{m} \tag{2.5.21}$$

2.5.3.5 Validation of GPS Model

The theoretical GPS sensor accuracy was validated against actual data as shown in figure 2.17.

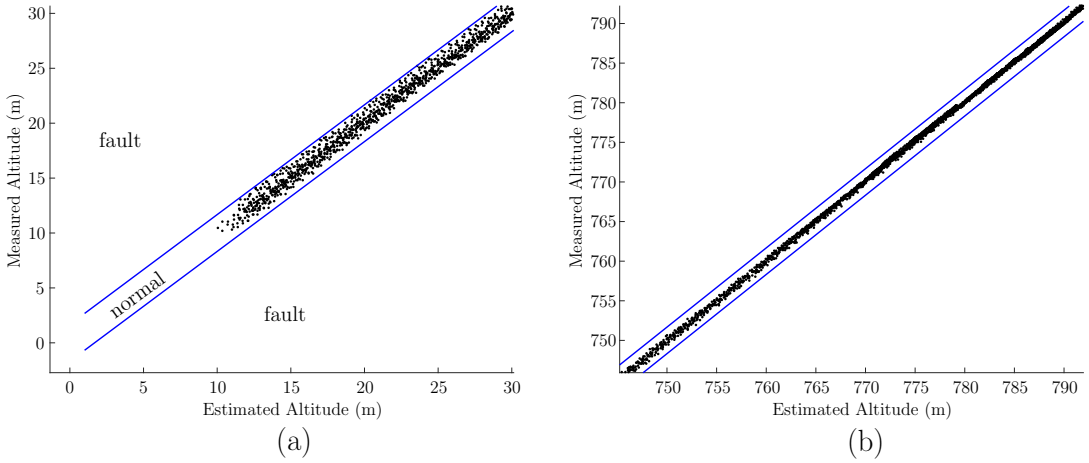


Figure 2.17: The GPS accuracy is shown as a function of the estimated relative altitude of an aircraft for actual measurement data. The blue line shows the theoretical accuracy limits for the GPS data and the black data points is a scatter plot of actual GPS measurement data. Figure (a) on the left shows a greater variance as there is more measurement data at a lower relative altitude, while figure (b) is at a higher relative altitude.

Figure 2.17 (a) shows the GPS sensor accuracy at lower relative altitudes, namely leading up to touch down, while figure 2.17 (b) shows the GPS sensor accuracy at higher relative altitudes, namely upon interception with the glide slope. There were no sensor faults in the dataset, and therefore it is expected that all of the GPS data will fall within the blue lines defined as the boundary for normal GPS sensor measurements. Any data points beyond the accuracy threshold are considered to be faulty. Given that there were only 26 landing runs in the dataset supplied by Airbus and that the true relative altitude of the aircraft is not known, it is very difficult to prove the statistical properties established above. Engineering judgment was used to ensure that the GPS measurement data all remained within the boundary of theoretical accuracy. The measurement data was a good fit of the theoretical accuracy at lower relative altitudes as seen in figure 2.17 (a). At higher relative altitudes there is less measurement data as many of the landing runs start at lower relative altitudes and therefore the boundaries appear to be a poor fit of the data as seen in figure 2.17 (b). However without more data this claim cannot be verified. If the boundaries were made any tighter, not all the data points would lie within the normal or no-fault region.

2.5.4 Inertial Sensor

2.5.4.1 Introduction

The inertial sensor gives the angular rates and accelerations with respect to the body axes of an aircraft as the accelerometers and gyroscopes are strapped along the body axis of the aircraft. The Euler 3-2-1 attitude parameterization is used to convert these measurements from the body axis system to the inertial axis system. These rates and accelerations propagate the aircraft's position forward in time. The inertial navigation system used in an aircraft is reset by the pilot when stationed on the runway before take-off. All displacement measurements will therefore be with reference to the departure airport.

2.5.4.2 Literature on Inertial Sensor Models

There are many different stochastic inertial sensor models found in literature. The majority of these models are for micro-electromechanical systems (MEMS), where recent advancements have paved the way for large scale fabrication of small and inexpensive sensors used in unmanned aerial vehicles. Beard et al. [3] established a model of a MEMS accelerometer that takes the true aircraft acceleration as input and incorporates the main sources of expected error before outputting a voltage measurement, Υ_{accel}.

$$\Upsilon_{accel} = k_{accel} \cdot A + \beta_{accel} + \eta_{accel} \tag{2.5.22}$$

where k_{accel} is the gain, A is the true acceleration , β_{accel} is the bias and η_{accel} is zero mean Gaussian noise.

While this model is theoretically correct, it fails to quantify the error sources and link them with the eventual relative altitude measurement. Quinchia et al. [4] investigated and compared the stochastic errors experienced by a MEMS inertial sensor using a single-axis accelerometer model developed by Skog et al. [14] shown in figure 2.18.

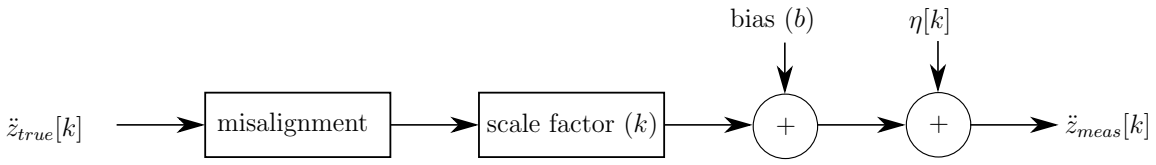


Figure 2.18: The IRS sensor model derived by Skog et al [14] for an accelerometer measurement along a single dimension.

(Note that Skog's single-axis model can be expanded to a three-axis model.) Quinchia et al. simplified the model of Skog et al. by grouping the error sources into two different categories, namely stochastic and deterministic. The deterministic errors are modelled with a bias, and are usually associated with manufacturing defects or alignment issues. The second category of error are random in nature and modelled by stochastic processes such as random walk or Gauss-Markov models.

The four different types of errors associated with an inertial sensor can be summarised as follows:

- misalignment - the non-orthogonality of the sensor axes or misalignment with the body axes. These are modelled by a deterministic error.
- scale factor - represents the sensitivity of the sensor. Errors are the result of manufacturing tolerances and aging. Ideally there should be a linear relationship between sensor input and output. Scale factor is modelled as having both a deterministic component and a stochastic component.
- bias - a constant or slowly varying additive error. A constant bias is a deterministic error, while a slowly-varying bias is a stochastic error.
- random error - system and sensor noise resulting from interference. The random error is stochastic in nature.

Deterministic error types such as misalignment and scale factor are negated by calibration and can therefore be ignored [4]. The stochastic error models and their coefficients are derived by examining the accelerometer measurement noise when at rest. Time domain and frequency analysis techniques are used to achieve this [15]. The slopes of a one-sided power spectral density are used to identify the stochastic noise models. Thereafter Allan variance is used to compute the coefficients for the respective error models. The resultant stochastic error model for a single-axis accelerometer used in an inertial sensor has two components, namely a slow-varying bias and measurement noise [15]:

$$\ddot{z}_{meas}[k] = \ddot{z}_{true}[k] + \eta[k] + W[k] \tag{2.5.23}$$
$$\dot{z}_{meas}[k] = \dot{z}_{meas}[k-1] + \ddot{z}_{meas}[k] \tag{2.5.24}$$
$$z_{meas}[k] = z_{meas}[k-1] + \dot{z}_{meas}[k] \tag{2.5.25}$$

where $W[k]$ is a first-order Gauss Markov model for a slow-varying bias.

An analysis was carried out by Quinchia et al. on a variety of accelerometers along each inertial axis. An example of the results obtained is given in table 2.3.

Direction	σ_w ($\mathrm{m\,s^{-2}}$)	T_c (s)	T_s (s)
Altitude (Z)	0.0068	20.74	0.02
Horizontal (X)	0.0063	4.56	0.02

Table 2.3: Example of Gauss Markov model parameters for slow-varying bias of a 3DM-GX3 IMU inertial sensor [4].

An identical approach was used by Naranjo et al. [16] who considered each axis separately. The same error types as discussed earlier are used, except that their model accounts for a scale factor error, K. The impact of the different error types on the positional measurement is given by Naranjo [16]:

$$z_{error} = \iint (K \cdot \ddot{z}_{true} + b)dtdt = \frac{1}{2}(K \cdot \ddot{z}_{true} + b)t^2 \tag{2.5.26}$$

where b is the bias error.

The double integration results in a quadratic increase in error as a function of time.

2.5.4.3 Inertial Sensor Model

The model of the inertial sensor found in literature was adapted to fit the data and sensor characteristics supplied by Airbus. The inertial sensor measurements provided by Airbus only contained velocity measurements in the X_R and Z_R axes, and did not contain the acceleration measurements themselves. The inertial sensor model will therefore only consist of two stages, namely velocity and displacement. The model proposed in this section incorporates some of the error types found in literature, and is based on a combination of the sensor characteristics supplied by Airbus and observations from the actual flight data that was provided.

The first modelling decision that had to be made, was how to initialise the inertial sensor's estimate of the aircraft position at the beginning of a simulation, from which future X_{IRS} and Z_{IRS} position measurements would be propagated by integrating the velocity measurements $\dot{X}_{IRS}$ and $\dot{Z}_{IRS}$. In reality there is no way of knowing the initial offset associated with the inertial sensor when the aircraft starts its final approach along the glide slope of 3°. Knowing that the inertial sensor was last zeroed just before take-off suggests that the effects of drift would be significant over an extended period of time. The design decision was taken to propagate future states from the GPS sensor measurement at time $t = 0\,\mathrm{s}$ such that $Z_{IRS}(t = 0) = Z_{GPS}(t = 0$ and $X_{IRS}(t = 0) = X_{GPS}(t = 0)$. The second modelling decision made was that misalignment and scale errors do not have to be modelled as they are accounted for during calibration as discussed in the model from literature.

The inertial sensor model architecture and parameter values were based on a combination of inertial sensor accuracy specifications provided by Airbus, and actual inertial sensor measurements recorded during several flights, also provided by Airbus. The inertial sensor accuracy specifications supplied by Airbus only specified the accuracies in the horizontal plane, and did not specify the accuracy of the relative altitude measurements. The accuracy of the relative altitude measurements was therefore identified by

analysing the flight data supplied by Airbus. The recorded IRS sensor measurements were compared with the recorded GPS sensor measurements. Since the GPS sensor bias and random noise remain practically constant over the duration of a landing, the GPS relative altitude measurements were used as the reference against which to compare the IRS relative altitude measurements. The IRS relative altitude measurements were obtained by initialising IRS sensor with the GPS relative altitude at the start of the simulation, and then propagating the IRS relative altitude by integrating the IRS vertical velocity measurements over time. The IRS and GPS relative altitude measurements are plotted against the GPS horizontal position in figure 2.19, and the difference between the IRS and GPS altitude measurements are plotted in figure 2.20.

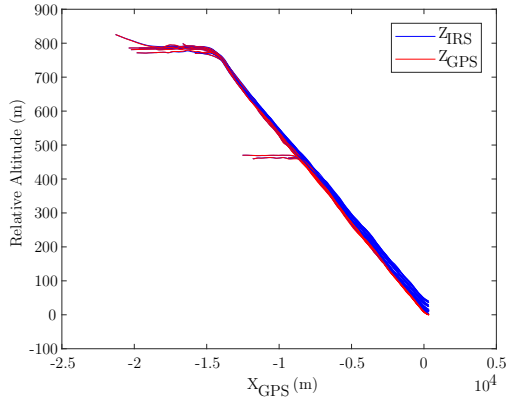


Figure 2.19: The Z_{IRS} and Z_{GPS} measurements versus X_{GPS} for the dataset supplied by Airbus.

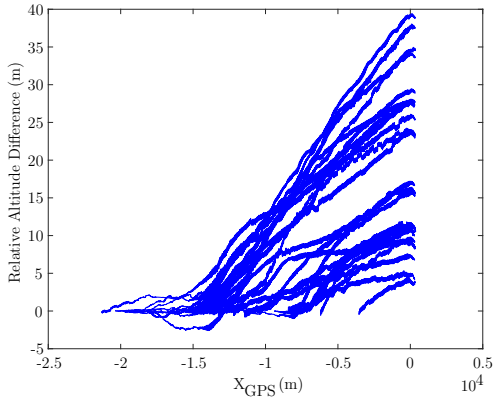


Figure 2.20: The difference between Z_{IRS} and Z_{GPS} measurements versus X_{GPS}.

A few notable observations can be made. Firstly, each simulation run has a drift error that results in a positive offset. Secondly, the individual gradients of the difference plots in figure 2.20 are all similar. The main component of the stochastic error model is a slow-varying bias on the velocity measurement that results in a displacement error that randomly walks around a deterministic gradient. This is modelled using the combination of a constant bias and a Gaussian white noise signal. The proposed model for the inertial sensor is given in figure 2.21.

The discrete-time equations that describe the inertial sensor relative altitude model shown in figure 2.21 are given by:

$$\dot{Z}_{IRS}[k] = \dot{z}[k] + b_{\dot{Z}irs} + \eta_{Zirs}[k] \tag{2.5.27}$$

$$Z_{IRS}[k] = \dot{Z}_{IRS}[k] \cdot T_s + Z_{IRS}[k\text{-}1] \tag{2.5.28}$$

where $b_{\dot{Z}irs}$ is the random IRS climb rate measurement bias and $\eta_{Zirs}[k]$ is Gaussian white noise.

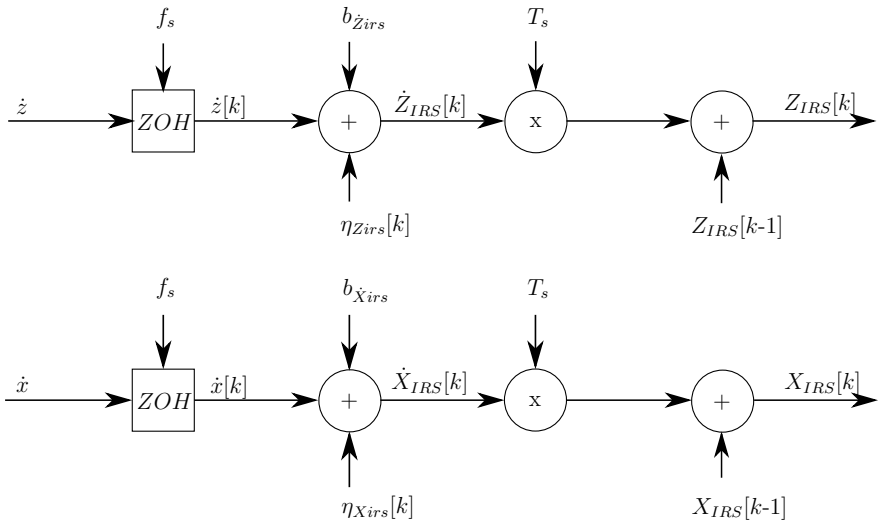


Figure 2.21: The stochastic IRS sensor model used to generate realistic climb rate and relative altitude measurements.

Similarly, the discrete-time equations that describe the inertial sensor measurement along the X_R axis are given by:

$$\dot{X}_{IRS}[k] = \dot{x}[k] + b_{\dot{X}irs} + \eta_{Xirs}[k] \tag{2.5.29}$$

$$X_{IRS}[k] = \dot{X}_{IRS}[k] \cdot T_s + X_{IRS}[k\text{-}1] \tag{2.5.30}$$

where $b_{\dot{X}irs}$ is the random IRS horizontal displacement bias and $\eta_{Xirs}[k]$ is Gaussian white noise.

The inertial sensor model will output both the aircraft position and climb rate measurement. The random climb rate bias is drawn from a uniform distribution with characteristics $b_{Zirs} \sim \mathcal{U}(\text{-}\delta_{\dot{Z}irs},\delta_{\dot{Z}irs})$ at simulation time $t = 0\,\text{s}$. The random climb rate bias will result in a constant IRS drift error where the rate of drift varies between simulation runs. η_{Zirs} is a Gaussian white noise component that will result in a random walk around the bias component. Similarly, the X_{IRS} measurement has a drift error that is the result of a bias and random component. The horizontal displacement bias is drawn from a uniform distribution with characteristics $b_{Xirs} \sim \mathcal{U}(\text{-}\delta_{\dot{X}irs},\delta_{\dot{X}irs})$ at simulation time $t = 0\,\text{s}$ and η_{Xirs} is Gaussian white noise component.

The results from figure 2.19 and figure 2.20 were used to quantify the model parameter values for the Z_{IRS} sensor using reverse engineering. The decision was made not to include the effect of the random noise component, η_{Zirs}, as its effect is negligible in comparison with that of the bias, b_{Zirs}. The Z_{IRS} drift error resulting from the bias error in the velocity measurement was approximated as a linear function of time. An IRS sensor with no bias error should therefore yield similar or identical measurements to that of the GPS sensor. Table 2.4 shows a summary of the measurement data in the DataSUN database supplied by Airbus that was used to derive error model parameter values. The average bias as a function of time was calculated in equation 2.5.31.

$$b_{Zirs} \text{ (per sample)} = \frac{Z_{IRS}[N] - Z_{GPS}[N]}{\text{N}} \tag{2.5.31}$$

where N is the number of samples, $Z_{IRS}[N]$ and $Z_{GPS}[N]$ are the relative altitude measurement from the inertial and GPS sensor at touch down.

DataSUN Number	**Number Samples (N)**	$Z_{GPS}[0]$ **(m)**	$Z_{IRS}[N] - Z_{GPS}[N]$ **(m)**	**Average** b_{Zirs} **(m/sample)**	**Average** b_{Zirs} **(m/m)**
1	1757	770	7	0.004	0.0091
2	994	463	5	0.005	0.011
:	:	:	:	:	:
26	447	4	190	0.0089	0.021

Table 2.4: IRS deviation values for the aircraft landings.

The average, standard deviation, and maximum values for the b_{Zirs} per sample across the landing runs is given by:

$$\begin{aligned} \mu &= 0.01273 \text{ m/sample} \\ \sigma &= 0.00475 \text{ m/sample} \\ \max &= 0.021\text{m/sample} \end{aligned}$$

The IRS drift errors across all the landing runs had a positive value. Due to the limited number of flight landings available the conservative decision was taken to assume that IRS drift could be both positive or negative. A zero-mean uniform distribution was therefore chosen to model the drift error between maximum negative and positive drift rates.

2.5.4.4 IRS Accuracy

Initially the IRS sensor accuracy is equivalent to that of the GPS as the IRS position is initialised with the GPS position at t = 0 s. Due to the sensor drift, the accuracy of the IRS changes with time. The average theoretical sensor accuracy for the IRS as a function of time can be summarised as follows:

$$\Delta_{Zirs}(t) = \Delta_{Zgps} + k \cdot \delta_{\dot{Z}irs} \tag{2.5.32}$$

$$\Delta_{Xirs}(t) = \Delta_{Xgps} + k \cdot \delta_{\dot{X}irs} \tag{2.5.33}$$

where k is the sample number, $\Delta_{Zgps} = 1.67\,\text{m}$, $\Delta_{Xgps} = 15\,\text{m}$, $\delta_{\dot{Z}irs} = 0.0127$ m/sample and $\delta_{\dot{X}irs} = 0.107$ m/sample.

More information on the model parameter values is given in Appendix A. It was also deemed necessary to develop a time-independent function for IRS sensor accuracy. Equation 2.5.32-2.5.33 are time-dependent functions which will result in the decision boundary being different for each landing run depending on the aircraft velocity and initial position.

Therefore it was decided to rather derive the IRS accuracy as a function of relative altitude that will remain objective and achieve uniformity irrespective of the relative altitude at which an aircraft intersects the glide path. The decision was made to specify the Z_{IRS} accuracy based on the worst-case scenario that would result in the maximum altitude drift over a landing. This coincides with the maximum possible aircraft position (z_{max}) and the largest velocity measurement bias b_{Zirs}. The average rate of drift, δ_{drift}, as a function of relative altitude was determined using a similar approach to the manner in which the average bias was derived as a function of time.

$$\delta_{drift} = \frac{Z_{IRS}[N] - Z_{GPS}[N]}{Z_{GPS}[0]} \tag{2.5.34}$$

where $Z_{IRS}[N] - Z_{GPS}[N]$ will be the drift error over the landing, and $Z_{GPS}[0]$ the initial relative altitude of the aircraft when it intercepts the glide slope.

The average, standard deviation and max values for the δ_{drift} across the landing runs in table 2.4 is given by:

$$\begin{aligned} \mu &= 0.031 \text{ m/m} \\ \sigma &= 0.011 \text{ m/m} \\ \max &= 0.051 \text{m/m} \end{aligned}$$

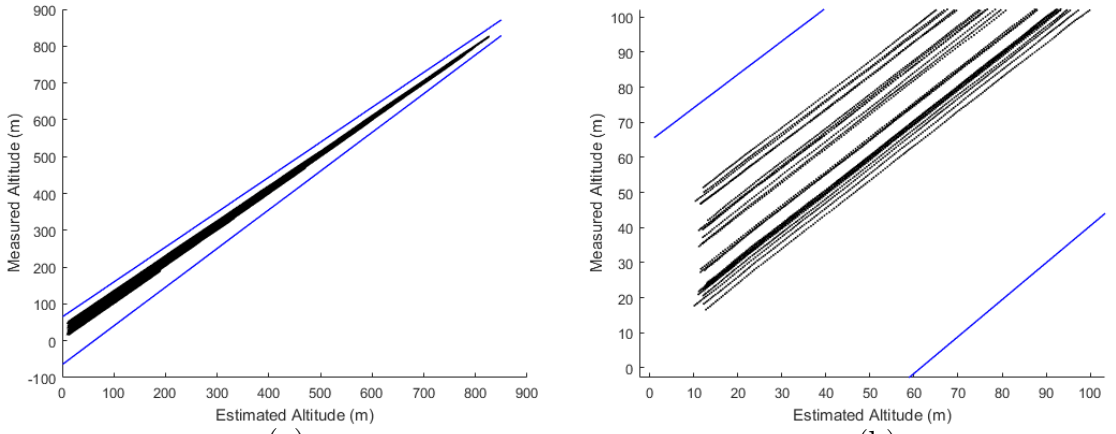
δ_{drift} was therefore chosen to be equal to 0.051m/m. The theoretical IRS sensor accuracy as a function of relative altitude is therefore given by:

$$\Delta_{Zirs}(z) = \Delta_{Zgps} + (z_{\max} - z) \cdot \delta_{\text{drift}} \tag{2.5.35}$$

where $\Delta_{Zgps} = 1.67\,\text{m}$, $z_{\max} = 1194\,\text{m}$, z is the instantaneous true relative altitude of the aircraft and $\delta_{\text{drift}} = 0.051\text{m/m}$.

2.5.4.5 Validation of IRS Model

The theoretical IRS sensor accuracy was validated against actual data as shown in figure 2.22. Figure 2.22 (a) plots the accuracy bounds for all the sensor data, while figure 2.22 (b) focuses on the accuracy bounds when approaching touch down.



(a) (b)

Figure 2.22: The IRS accuracy is shown as a function of the estimated relative altitude of an aircraft for actual measurement data.

The blue line shows the theoretical accuracy limits for the IRS data, and the black data points are a scatter plot of actual IRS measurement data. It is clear from figure 2.22 that the IRS drifts are only positive and that the theoretical accuracy appears to be a poor fit of the actual data. This is explained by the derivation of IRS accuracy that assumes the worst-case scenario and that the drift is equally likely to be positive or negative. If all aircraft intercepted the glide slope at z_{max} then the decision boundary would be a much tighter fit.

2.5.5 Instrument Landing System

2.5.5.1 Introduction

The Instrument Landing System (ILS) is an approach aid that assists pilots to control the aircraft along a predetermined glide path when landing. It is manipulated to give the relative altitude of the aircraft with knowledge of the aircraft's x position and the location of the Glide Slope Station for a particular runway as illustrated by figure 2.23. The ILS uses two radio beams and high intensity lights to provide pilots with lateral and vertical guidance as an aircraft is on final approach.

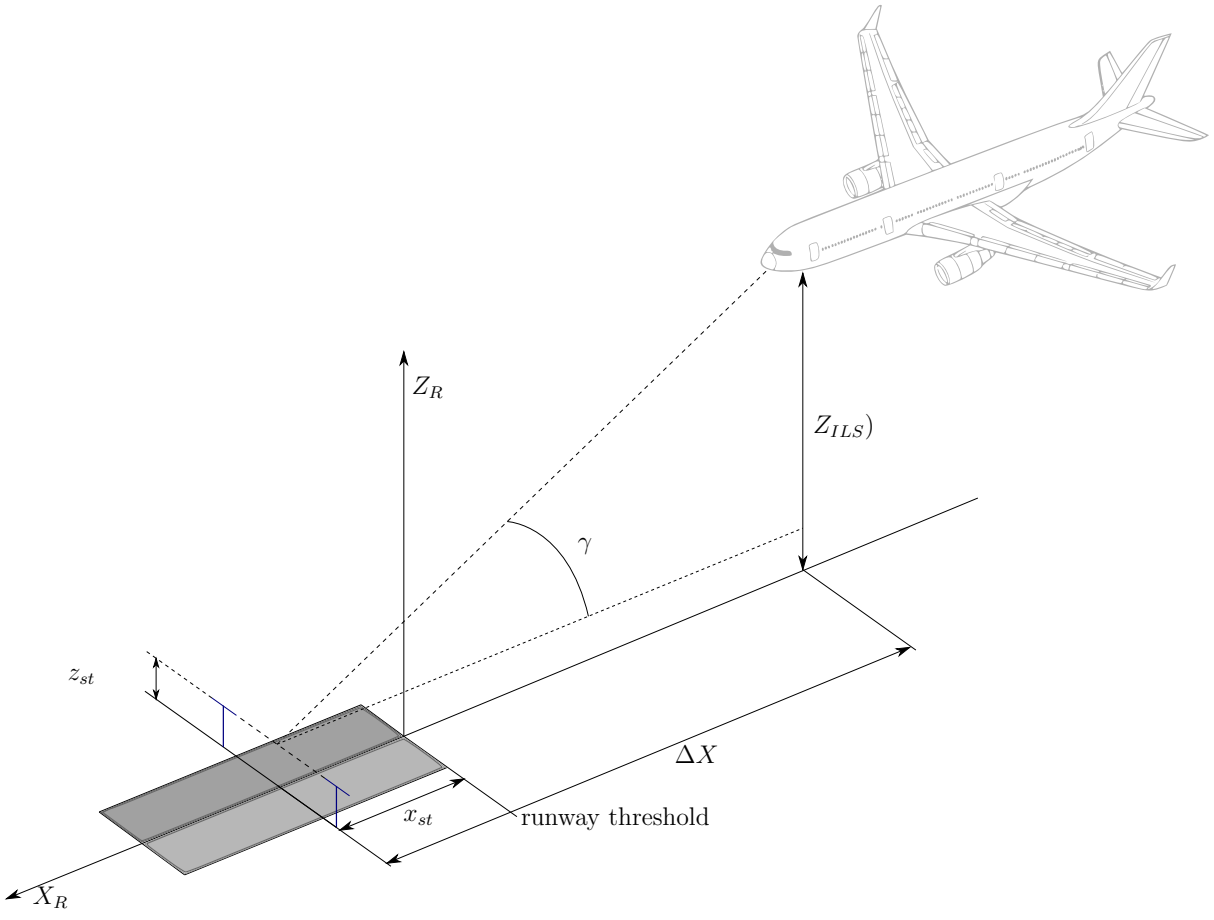


Figure 2.23: A visual explanation of the set-up used to manipulate the Instrument Landing System into a relative altitude sensor.

There are three different elements that make up the ground system of an ILS, namely

marker beacons, localiser and glide path radio beams. The localiser provides the pilot with lateral guidance to the runway. It consists of two separate antennas that create radiative patterns. One lies slightly to the right of the runway centerline, the other to the left. Where the two radiative patterns intersect is known as "on LOC" and indicates to the pilot that his aircraft is aligned with the runway. An identical approach is used to provide a pilot with vertical information known as the glide slope indication. The glide path is the designated approach angle for a runway and is usually about 3° [17]. This may differ from runway to runway as it depends on the configuration of the Glide Slope Station antennas. The third element of a ILS system is the marker beacons. These provide indications of the distance to the runway and may have up to three different beacons, namely outer, middle and inner marker [18]. The outer marker is usually located between 6.5 km and 11.5 km from the runway threshold. The middle marker is located roughly 1.3 km from the runway threshold and indicates to the pilot that aircraft is 65 m above ground level. The inner marker designates the decision point where the pilot must decide whether or not to continue the approach.

The glide slope is the vertical assistance portion of the ILS. A radiation pattern is generated in space. A glide slope measurement is proportional to the vertical deviation between the aircraft position and the glide path. Two antennas are used to develop a spatial glide path pattern as seen depicted in figure 2.24. The relative positions of the antennas and their positions relative to the ground determine the spatial properties of the radiation pattern. The one antenna produces a radiative pattern predominantly above the glide path that is modulated at 90 Hz, whilst the other antenna produces a radiative pattern predominantly below the glide path that is modulated at 150 Hz. The difference in depth of modulation (DDM) between the two sinusoidally modulated signals can be transformed into an angular deviation and provides guidance information to the pilot [19]. The aircraft contains a detector circuit that will amplify and filter the received signal to isolate the 90 and 150 Hz components. A bridge rectifier is then used to give a DC voltage that is proportional to the depth of modulation for the 90 and 150 Hz components. When the aircraft is flying along the predefined glide path the DDM is zero. An offset from the glide path is proportional to the DDM. A positve DDM indicates that the aircraft is above the reference glide path, while a negative DDM indicates that the aircraft is below the reference glide path.

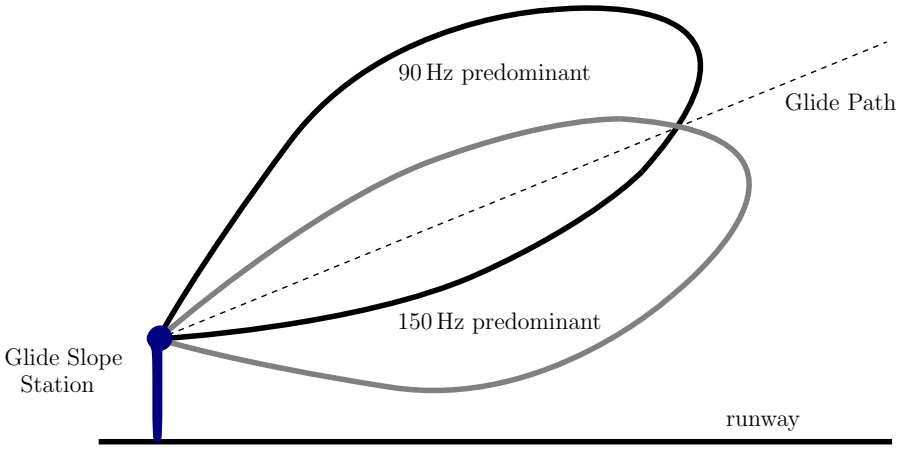


Figure 2.24: An exaggerated view of the spatial glide pattern to determine the glide slope angle of an aircraft.

2.5.5.2 Literature on ILS Measurement Models

No literature was found on measurement error models for the ILS glide slope angle. The closest model that was found in literature was by Jarama et al. [20] who investigated an error model for secondary surveillance radar systematic errors used in air traffic control networks. While there are obvious differences between ILS and radar systems, there is a link between the azimuth and the glide slope angle. The azimuth is the angle between the antenna boresight and target altitude [20]. The model proposed by Jarana et al. for the azimuth combined and grouped errors into deterministic and stochastic categories.

$$\theta_{meas}[k] = \theta_{true}[k] + b + \eta_k \tag{2.5.36}$$

where θ is the azimuth angle.

The deterministic errors are attributed to the misalignment of the antennas, the non-orthogonality between axes and the calibration error of the azimuth decoder [20]. The stochastic measurement noise is due to quantization errors and erroneous reflections.

The literature survey did however uncover the maximum allowable deviation in the glide slope angle measurement at the three marker beacons as given by Maybeck [5], and summarised in table 2.5 below.

Beacon Location	**Glide-slope deviation**
Middle to runway threshold	0.06°
Between Outer and Middle	0.1° decreasing to 0.06°
Beyond Outer	0.1°

Table 2.5: The maximum allowable measurement error for the glide slope sensor [5].

2.5.5.3 ILS Relative Altitude Sensor Model

Three important pieces of information are necessary for the ILS to give an aircraft's relative altitude. Firstly, knowledge of the exact location of the runway's glide slope measurement station (x_{st}, z_{st}) is required. This location is actually beyond the runway threshold to ensure that an aircraft does not touch down short of the runway. Secondly, information on the aircraft's location along the X_R axis is obtained from the GPS or IRS measurement. This is used to determine the horizontal displacement Δx. Lastly a glide slope angle is required to determine the relative altitude of the aircraft using trigonometry.

Airbus supplied the X_R location of the glide slope measurement station, but the Z_R position was unknown and therefore had to be derived from actual landing data supplied by Airbus. The Z_R location was chosen as the value that minimised a cost function defined as the sum of square residuals between the Z_{GPS} and Z_{ILS} over the entire dataset as illustrated in figure 2.25. For the purpose of determining the relative vertical altitude of the glide slope measurement station, the GPS altitude measurement Z_{GPS} was treated as the reference altitude, since it was considered to be the most accurate relative altitude sensor. Using this approach, the relative altitude of the glide slope measurement station was calculated to be $z_{st} = -8\,\text{m}$.

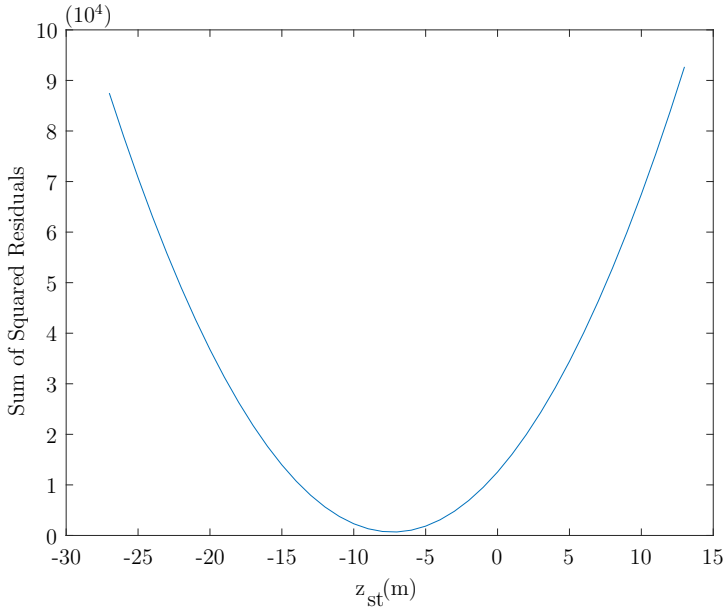


Figure 2.25: The results for the experiment performed to determine the optimal z_{st} location.

Although a relative vertical altitude of $-8\,\text{m}$ seems like an improbable location for the glide slope measurement station, it fits the actual flight data the best, and will therefore be treated as correct. Given that the location of the glide slope measurement station has been established, we can proceed to establish an ILS sensor model. The ILS model consists of two separate parts as shown in figure 2.26.

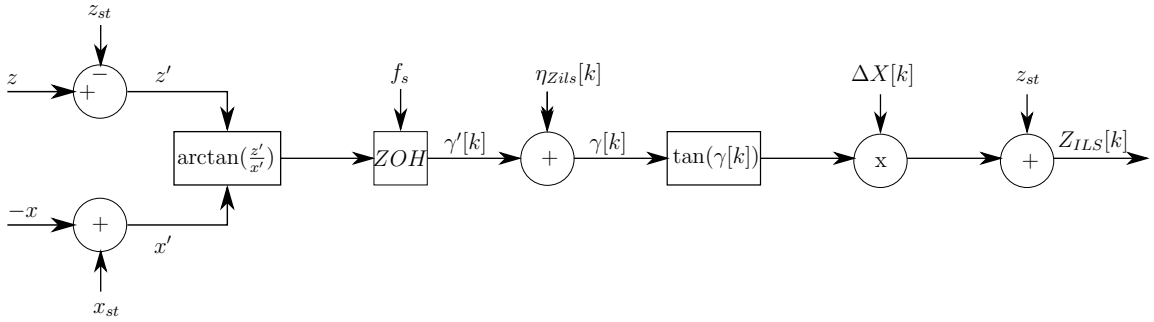


Figure 2.26: The stochastic ILS sensor model for a relative altitude measurement.

The first part establishes a realistic measurement for the glide slope angle using the true location from the aircraft model. The second part establishes a realistic glide slope angle measurement. The true angle is sampled at f_s Hz and is represented by $\gamma'[k]$. Due to the lack of literature and information on glide slope measurement errors, the assumption was made that any fixed errors are negated through calibration and that the remaining unknown stochastic components are additive and modelled by Gaussian white noise to

give a realistic glide slope measurement $\gamma[k]$.

$$\gamma' = \arctan(\frac{z - z_{st}}{x_{st} - x}) \tag{2.5.37}$$

$$\gamma[k] = \gamma'[k] + \eta_\gamma[k] \tag{2.5.38}$$

where $\eta_\gamma[k]$ is discrete-time Gaussian white noise.

The Gaussian white noise is drawn from a Normal distribution where $\eta_{Zils}[k] \sim \mathcal{N}(0,{\sigma_\gamma}^2)$. The effect of $\eta_\gamma[k]$ on the aircraft altitude is more apparent for larger ΔX values. The $\Delta X[k]$ is not directly available, but can be determined from either the GPS or IRS sensor measurement.

$$\Delta X[k] = x_{st} - X_{GPS}[k] \text{ OR } x_{st} - X_{IRS}[k] \tag{2.5.39}$$

It must be noted that the negative of the X_{GPS} and X_{IRS} is taken as the aircraft approaches the runway along the negative X_R axis. The $Z_{ILS}[k]$ measurement can be calculated via trigonometry and adjusted to account for the altitude difference between the X_R axis and glide slope station.

$$Z_{ILS}[k] = \Delta X[k] \cdot \tan(\gamma[k]) + z_{st} \tag{2.5.40}$$

$$Z_{ILS}[k] = (x_{st} - X_{GPS}[k]) \cdot \tan(\gamma[k]) + z_{st} \tag{2.5.41}$$

2.5.5.4 ILS Accuracy

From equation 2.5.41 it is apparent that the theoretical accuracy of the ILS sensor is dependent on both the noise associated with the glide slope angle and the accuracy of the X_{GPS} measurement. The theoretical standard deviation was obtained by linearizing around a nominal x and γ. For notation sake it is assumed that in the continuous domain the relative altitude of the aircraft is reduced to:

$$z = x \cdot \tan(\gamma) + z_{st} \tag{2.5.42}$$

where x is equivalent to ΔX.

Taking the partial derivatives with respect to x and γ yields:

$$\frac{\delta z}{\delta x} = \tan(\gamma) \tag{2.5.43}$$

$$\frac{\delta z}{\delta \gamma} = x \sec^2(\gamma) \tag{2.5.44}$$

The partial derivatives are combined to give the standard deviation of the ILS relative altitude measurement error as follows:

$$\Delta z = \frac{\delta z}{\delta x}\Delta x + \frac{\delta z}{\delta \gamma}\Delta\gamma \tag{2.5.45}$$

$$\Rightarrow \sigma_{Zils} = \tan(\gamma)\sigma_x + x\sec^2(\gamma)\sigma_\gamma \tag{2.5.46}$$

where σ_x is the standard deviation of the GPS horizontal position measurement error and σ_γ is the standard deviation of the ILS glide slope angle measurement.

A parameter value for σ_γ was supplied by Airbus, but was a poor fit of the actual data and was significantly smaller than the values from table 2.5 that were found in literature. A reverse engineering approach was therefore used to derive the parameter value for the standard deviation of the ILS glide slope angle measurement. The noise characteristics were derived by regarding the GPS sensor measurements as the truth and using trigonometry to calculate the true glide slope angle measurement as follows:

$$\gamma_{true}[k] = \arctan\left(\frac{Z_{GPS}[k] - z_s t}{x_{st} - X_{GPS}[k]}\right) \tag{2.5.47}$$

The flight data for all the aircraft approaches were analyzed for $Z_{GPS}[k] > 200$. This would minimise the effect of the X_{GPS} error in accuracy that has a notable influence for smaller $(x_{st} - X_{GPS}[k])$ values. What remains after subtracting $\gamma_{true}[k] - \gamma_{[}k]$ is a zero-mean random noise signal. The standard deviation was extracted from this random noise signal for individual runs and was then averaged across all aircraft approaches.

$$\sigma_\gamma = \frac{1}{J}\sum_{j=1}^{J}\left(\frac{1}{N-1}\sum_{i=1}^{N}(\gamma_{true}[k] - \gamma_{[}k])\right) \tag{2.5.48}$$

where j is the index of the flight data for a single aircraft approach, J is the total number of approaches and N is the number of data points from the j^{th} approach for which $Z_{GPS} > 200$.

An additional observation and concern when using the ILS sensor is that its measurement becomes distorted closer to the glide slope measurement station. In reality it is not located in the X_R and Z_R plane, but rather off to the side of the runway. An analysis of the example sensor data showed that ILS measurement becomes distorted beyond the runway threshold and should therefore not be used for $X_{GPS} > 0$. The theoretical accuracy for the ILS sensor is dependent on the aircraft location. The definition for accuracy used in this thesis is a boundary that should include all or 99.65 % of no-fault normally distributed measurements. This corresponds to a boundary of 2.695 standard deviations for normally distributed data.

$$\Delta_{Zils} = 2.695 \cdot \sigma_{Zils} \tag{2.5.49}$$

2.5.5.5 Validation of ILS Model

The theoretical ILS sensor accuracy was validated on actual data as shown in figure 2.27. Figure 2.27 (a) plots the accuracy bounds for all the sensor data, while figure 2.27 (b) focuses on the accuracy bounds as the aircraft approaches touch down. It is observed from figure 2.27 that the majority of the data points are closely distributed around the glide path. However there are two obvious non-linearities in the data as indicated by the regions within the red circles in figure 2.27 (a). At these points the ILS measurement data comes very close to exceeding the ILS accuracy bounds. Other than these two locations the ILS sensor data appears to be a poor fit of the measurement data, but is left unchanged to account for the worst case scenario. The ILS sensor model parameters and assumptions should be verified against a larger set of measurement data.

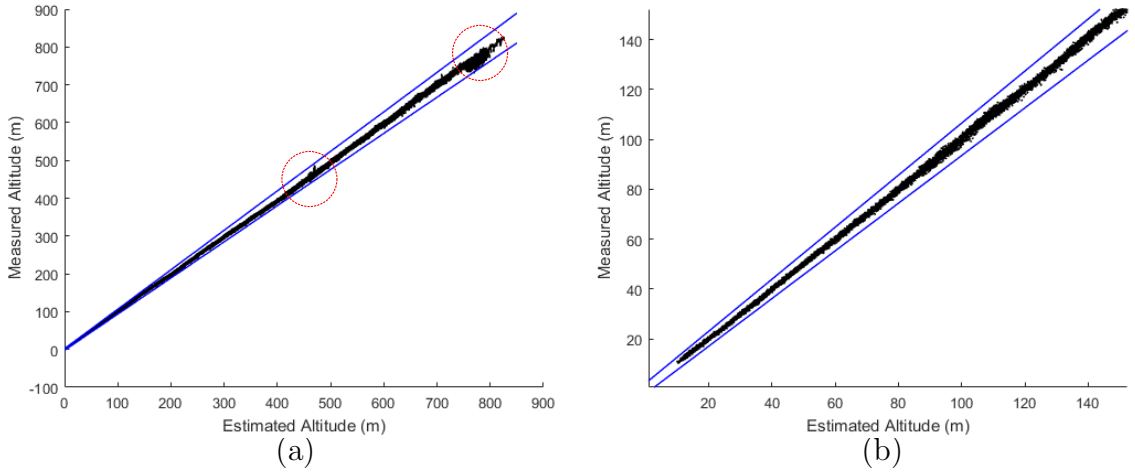


Figure 2.27: The ILS accuracy as a function of the estimated relative altitude for actual aircraft measurement data. The blue line shows the theoretical accuracy limits for the ILS data and the black data points is a scatter plot of actual ILS measurement data.

2.5.6 Radio Altimeter

2.5.6.1 Introduction

A radio altimeter operates on similar principles to radar except that microwaves are used instead of radio waves. The radio altimeter is the only sensor that directly measures the aircraft height. Commercial aircrafts use Frequency Modulated Continuous Wave (FMCW) altimeters as these measure both the height and its instantaneous rate of change.

A radio altimeter measures the height of the aircraft above the earth's surface with reference to the lowest point of an aircraft's landing gear. In other words the radio altimeter measurement will be zero when the landing gear makes contact with the runway. Typically radio altimeters are installed on the fuselage between the wings of commercial aircraft and the height difference between the fuselage and the lowest wheel is compensated for electronically [21]. Height information is displayed to the pilot and used by the Automatic Flight Control System (AFCS) during controlled approaches and landings. The radio altimeter serves as a sensory input to the Ground Proximity Warning System (GPWS) that raises an alarm when the aircraft is flying too close to the ground or is descending too rapidly. Radio altimeters work on the same principle as radar, where distance is measured based on the time of flight. There are two categories of waveforms used in radio altimetry, namely continuous and pulsed waves. Pulsed wave altimeters are usually employed at higher altitudes above 5000ft, while continuous wave are preferred for lower altitudes [22]. The 4.2-4.4 GHz band has been internationally reserved for the use of radio altimeters installed on commercial aircraft [23].

Doppler

A brief review of the Doppler effect is necessary to understand the inner workings of a FMCW radio altimeter. Austrian scientist Christian Doppler coined the term Doppler effect after observing the audible change in sound waves with relative motion [24]. It has since been discovered that the Doppler effect is applicable to the full spectrum of electro-

magnetic radiation, where the formula for the frequency of a reflected signal observed at the radar is given by:

$$f_r = \frac{c + v_r}{c - v_r} f_t \tag{2.5.50}$$

where f_r is the frequency of the received signal, c is the speed of light, f_t is the frequency of the transmitted signal and v_r is the relative velocity between the target and radar.

The Doppler shift is the change in frequency given by the difference between the received and transmitted frequencies [21]:

$$f_d = f_r - f_t = f_t(\frac{2v_r}{c - v_r}) \tag{2.5.51}$$

Under the assumption that $c >> v_r$, the change in frequency can be simplified as follows:

$$f_d \approxeq \frac{2v_r}{c} f_t \tag{2.5.52}$$

Frequency Modulated Continuous Wave

Traditional radar technologies determine distance based on the time difference between an emitted pulse and its returning echo.

$$\Delta t = \frac{2h}{c} \tag{2.5.53}$$

Frequency modulated continuous wave altimeters have the additional capability of measuring the rate of change in height. This is extracted from the Doppler shift in the reflected signal. Modulating the frequency of the emitted signal gives it a distinct mark or "time stamp" that reflects the range to the ground as the frequency difference known as the "beat frequency". FMCW altimeters therefore extract both the Doppler shift and "time stamp" from the reflected signal. The signal depicted in figure 2.28 is known as a chirp and depicts the effects of a linear change in frequency.

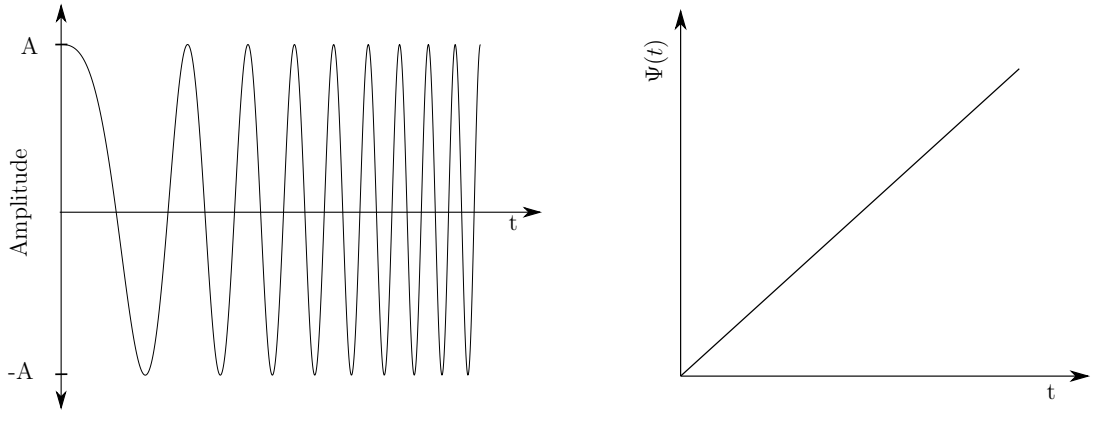


Figure 2.28: Chirp signal whose frequency varies linearly with time.

Mathematically, a chirp signal is described with the following equations:

$$F_{chirp}(t) = A\cos(\Psi(t) \cdot t) \tag{2.5.54}$$

$$\Psi(t) = f_o \cdot t \tag{2.5.55}$$

where A is the amplitude and f_o is the carrier frequency.

Frequency modulation is achieved in a variety of different ways. Figure 2.29 shows popular techniques like saw-tooth, triangular and sinusoidal modulation.

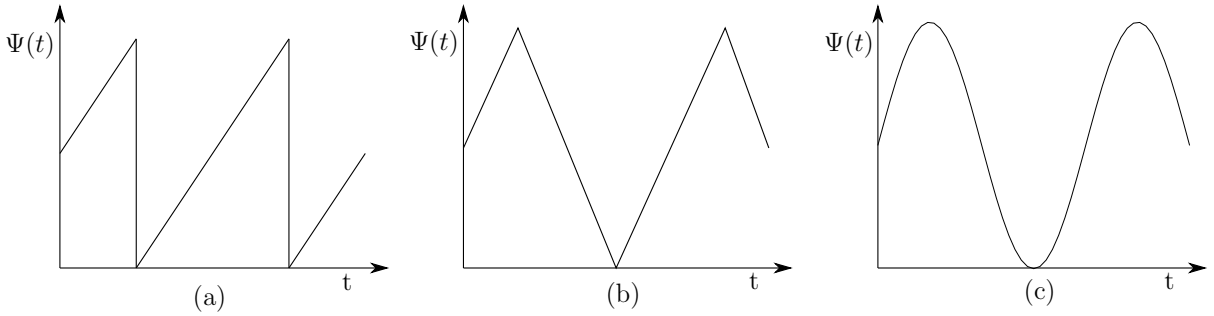


Figure 2.29: Commonly used modulation techniques: (a) saw-tooth (b) triangular (c) sinusoidal modulation.

Due to the delay between the transmitted and received signal there will be a difference in their respective frequencies. The height of the aircraft is proportional to this frequency difference denoted by f_h [25]. The mixing of the received and transmitted signals results in the difference or beat frequency, Ψ_b [26]. A triangular waveform is used for the explanation of how a FMCW radio altimeter works. For this explanation it is assumed that the rate of change in height of the aircraft is zero. Figure 2.30 shows how the frequency is linearly varied over a single modulation period, T, by bandwidth Δf. The signal is modulated at a frequency, f_m, which is also known as the pulse repetition frequency.

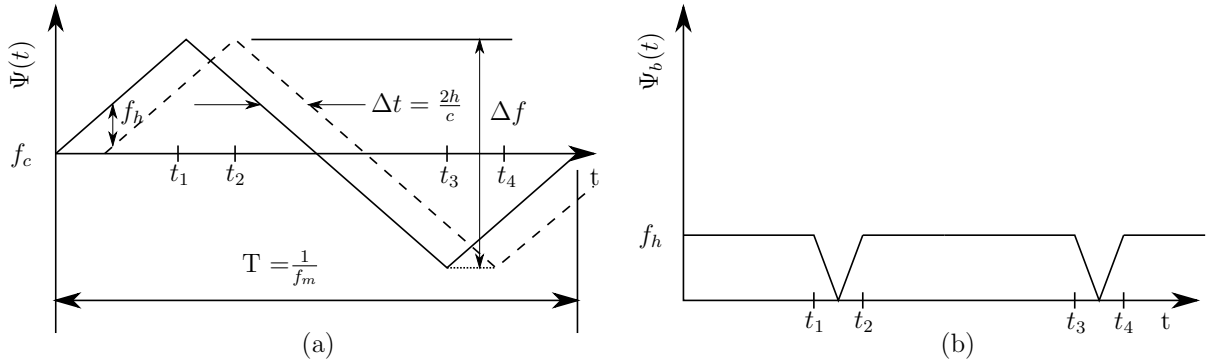


Figure 2.30: The solid line represents the transmitted signal and the dashed line the reflection from ground. (a) triangular modulation (b) the resultant beat signal associated with the triangular modulation [22].

The rate of change in frequency over a linear up-sweep is given by [25]:

$$\frac{df}{dt} = \frac{\Delta f}{T/2} = 2f_m\Delta f \tag{2.5.56}$$

From figure 2.30 the difference in frequency is given by:

$$f_h = \Delta t \cdot \frac{df}{dt} \tag{2.5.57}$$

Substituting in equation 2.5.53 and 2.5.56 into the expression above yields:

$$f_h = \frac{2h}{c} \cdot 2f_m\Delta f \tag{2.5.58}$$

$$\Rightarrow h = \frac{f_h c}{4f_m\Delta f} \tag{2.5.59}$$

The height of the aircraft is extracted from the beat frequency $\Psi_b(t)$ given in figure 2.30 (b). This signal is a constant f_h, except at the turnaround region. The case is now considered where relative motion between the aircraft and the ground below it is introduced as illustrated in figure 2.31. This results in a Doppler shift due to the introduction of a rate of change in height.

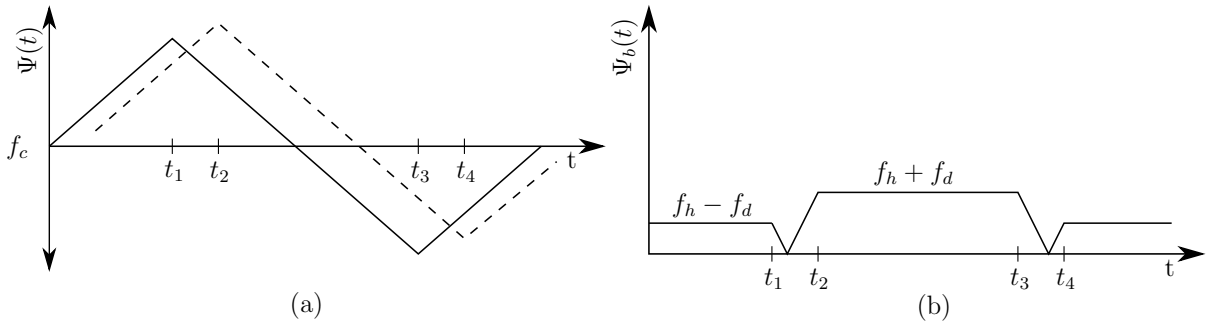


Figure 2.31: The corresponding beat frequency for a triangular modulation with a Doppler shift [22].

The resultant beat signal is $f_h + f_d$ and $f_h - f_d$ for equal periods. The beat signal is a function of two components. The first is a function of height, denoted f_h, while the second is a function of Doppler shift, denoted f_d. The average beat signal over a modulation period is equal to $\frac{1}{2}(f_h + f_d + f_h - f_d) = f_h$ [25].

2.5.6.2 Literature on Radio Altimeter Models

There was very limited literature available on mathematical models that can be used for a radio altimeter. Thomas et al. [2] state that all uncertainties associated with the radio altimeter measurement are modelled by assuming normally distributed random variables. No mention is made of the sensor model that is used to simulate realistic measurements, although it would appear that a first-order Gauss-Markov model was used to simulate measurements that are a mixture between random noise and random walk. The only

radio altimeter model that was found in literature was proposed by Hajiyev et al. [27]. Their height measurement error model for a radio altimeter is given by:

$$\Delta H_{error}[k] = \Delta H_{error}[k-1] - \beta \Delta H_{error}[k-1] + \eta[k] \tag{2.5.60}$$

where ΔH_{error} is the height error and β is the correlation time constant.

This error model has two components, namely a time-correlated and a random error. The time-correlated component decreases the radio altimeter error to model sensor measurements that become more accurate with time. The rate of increase in accuracy is dependent on the β value. This is not technically correct as the radio altimeter accuracy is a function of height rather than time as suggested by the theoretical accuracy values supplied by Airbus. The random error is modelled by discrete-time Gaussian white noise. Hajiyev et al. did not disclose the motivation for this error model or the parameter values. There was however other literature that discussed the sources of error and the theoretical accuracy of the radio altimeter. James Powell discusses the sources of error associated with radio altimetry in his book Aircraft Radio Systems [25]. These error sources include timing errors, fluctuations in signal strength, aircraft installation delay, leakage between the transmitter and receiver, reflections off the landing gear, multipath signals, and the effect of the aircraft orientation.

FMCW altimeters use a counter to determine the number of cycles or half-cycles in one period of modulation. This is achieved by counting the number of positive zero going crossing points. This varies between N and $N+1$ depending on the phase shift. The height equation 2.5.59 can be rewritten as a function of N:

$$h = \frac{f_b c}{4 f_m \Delta f} = \frac{cN}{4\Delta f} \tag{2.5.61}$$

where $N = f_b/f_m$ rounded down to the nearest integer.

Since N is an integer, the resultant height could be out by a factor of $c/(4\Delta f)$ [25]. This results in a quantization error called the step or fixed error that is roughly equal to 0.75 m with a $\Delta f = 100$ MHz. The step error can be mitigated by using a frequency discriminator since the measurements are continuous rather than discrete.

The second source of error is the received signal strength that varies as a function of aircraft height. The radar equation gives the received signal strength as a fourth power of range, while for radio altimeters it is a square function of the range [25] as a greater target area is illuminated with an increase in height, resulting in a stronger reflection.

Part of the two way travel time is due to the electronics and the height of the antennas above the ground. The antennas are mounted on the fuselage, while the reference point for height is given from the lowest point of the landing gear. This inherent delay is built into the radio altimeter functioning that accounts for signal propagation along cables and antenna height. A residual height error forms part of the Aircraft Installation Delay (AID) and is accounted for during calibration such that the altimeter reads a height of zero at touch down.

Transmitter-receiver leakage is a common source of error that affects the receiver sensitivity and could lead to erroneous readings [25]. Separate transmitter and receiver

antennas will reduce leakage and should provide space attenuation in the region of 75 dB [25].

Multipath signals are another common error source. These arise from objects within the illuminated target area returning stronger signals than the area directly below the aircraft, or reflections of the first-time-around signal off the airframe that return to the ground and register as a second-time-around echo. The return echo should be significantly weaker than the required signal strength and should therefore be ignored.

The aircraft's orientation with respect to the inertial axis could possibly also effect the radio altimeter measurement. The effect of pitch and roll depends on the beam width of the radio altimeter set up. A wider beam width covers a larger surface area and therefore mitigates the effects of small pitch and roll perturbations as the shortest route will always be registered by the receiver.

The accuracy of a radio altimeter as given by Nebylov et al. [22]:

0-500 ft	$\pm$ 2 ft or 2 % of height depending on which is greater
$>$ 500 ft	5 % of height

2.5.6.3 Radio Altimeter Sensor Models

A simplified radio altimeter model is proposed that is based on the combination of information obtained from literature and the theoretical sensor characteristics supplied by Airbus. The majority of the error sources from literature can be neglected under normal operating conditions, but will form part of the fault profiles defined later in this chapter. The assumption is made that the effect of the aircraft's orientation is negligible as this thesis deals with a landing scenario. The aircraft's pitch is assumed to be a small enough such that it has no significant effect on the correct functioning of the radio altimeter. The assumption is also made that any roll perturbations are small, and insignificant as the aircraft's heading does not change significantly as it approaches the runway.

Without the true height of the aircraft and accurate terrain information it is very difficult to derive error statistics. A simplified model is therefore proposed in figure 2.32 that is based on the theoretical accuracy of the radio altimeter as a function of height. The mathematical model to describe the aircraft height is given by:

$$H_{RA}[k] = h[k] + b_{Hra}(h) \tag{2.5.62}$$

where h is the true height and b_{Hra} is a bias drawn from a uniform distribution that is scaled as a function of aircraft height.

The bias is drawn from a uniform distribution described by $\mathcal{U}(-\delta_{Hra}, \delta_{Hra})$. δ_{Hra} is given by the theoretical limits of radio altimeter accuracy. This value is scaled according to the change in the theoretical accuracy of the sensor as a function of height. The resultant sensor model is illustrated in figure 2.32.

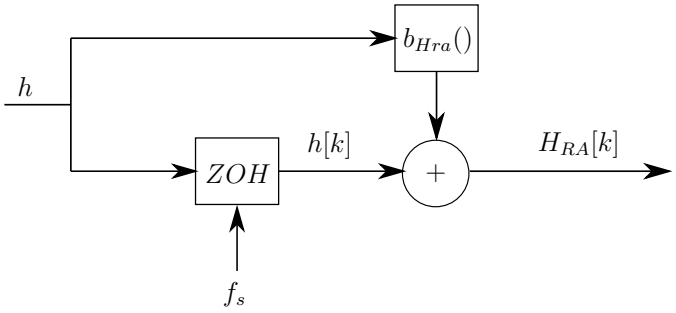


Figure 2.32: The stochastic radio altimeter model.

2.5.6.4 Radio Altimeter Accuracy

The theoretical accuracy of the radio altimeter is given by a piece-wise function that varies with height.

$$\Delta_{Hra} = \begin{cases} 0.9144\,\text{m} & \text{for } h < 30.5\,\text{m} \\ 0.03 \times h\ \text{m} & \text{for } 30.5\,\text{m} \leq h < 152\,\text{m} \\ 0.05 \times h\ \text{m} & \text{for } 152\,\text{m} \leq h < 1676\,\text{m} \end{cases} \tag{2.5.63}$$

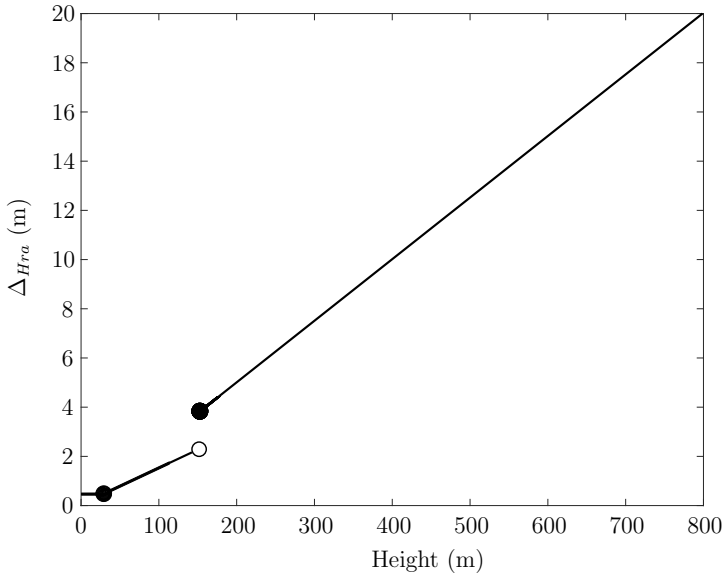


Figure 2.33: Piece-wise function showing the radio altimeter accuracy versus height.

2.5.6.5 Validation of Radio Altimeter Model

The validation of the radio altimeter accuracy in the height domain proved to be challenging as there were many uncertainties surrounding the accuracy of the terrain databases that made it difficult to accurately estimate the aircraft height. The decision was therefore taken to convert the height measurements to the relative altitude domain. The aircraft height can also be converted to relative altitude using the terrain database Z_{T1} that is loaded on the flight computer and with knowledge of the aircraft's X_R position.

$$Z_{RA}[k] = H_{RA}[k] + Z_{T1}(x) \tag{2.5.64}$$

This complicates matters as the accuracy function in the relative altitude domain is dependent on both the accuracy of the radio altimeter and the terrain database. To simplify matters the naive assumption is made that the accuracy for the radio altimeter and terrain database can be combined by assuming that they are each modelled by independent Gaussian random variables. As presented earlier in equations 2.5.9-2.5.10 the sum of two normally distributed random variables $(X + Y)$ is also a normally distributed random variable (Z), that is given by:

$$Z = X + Y \sim \mathcal{N}(\mu_x + \mu_y, {\sigma_x}^2 + {\sigma_y}^2) \tag{2.5.65}$$

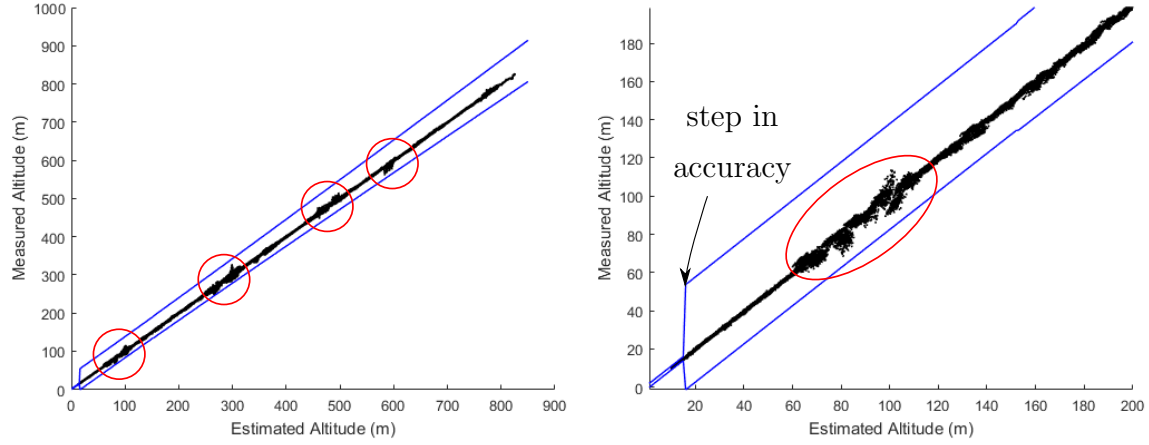
The terrain database accuracy can be approximated by the normal distribution $X \sim \mathcal{N}(\mu_x, {\sigma_x}^2) = \mathcal{N}(\mu_{terr}, {\sigma_{terr}}^2)$. The radio altimeter accuracy in height however needs to be transformed from a uniform distribution to a normal distribution. The variance of a zero mean uniform distribution, $\mathcal{U} \sim (-\delta, \delta)$, is given by $\sigma^2 = \frac{1}{3}\delta^2$. Therefore, the radio altimeter accuracy in height is approximated by $Y \sim \mathcal{N}(\mu_y, {\sigma_y}^2) = \mathcal{N}(0, \frac{1}{3}\Delta^2_{Hra})$. These two distributions are now combined as follows:

$$\Rightarrow Z \sim \mathcal{N}(\mu_{terr}, \frac{1}{3}\Delta^2_{Hra} + \sigma^2_{terr}) \tag{2.5.66}$$

The definition for accuracy used in this thesis is a boundary that should incorporate 99.65 % of no-fault measurements. This corresponds to a boundary of 2.695 standard deviations for normally distributed data. The theoretical function for the radio altimeter accuracy in relative altitude is given by:

$$\Delta_{Zra} = \begin{cases} 2.695 \cdot \sqrt{\frac{1}{3}\delta^2_{Hra} + \sigma^2_{terr}} + \mu_{terr} & \text{for } X_{GPS} < 0\,\text{m} \\ 2.695 \cdot \frac{1}{\sqrt{3}}\delta_{Hra} & \text{for } X_{GPS} \geq 0\,\text{m} \end{cases} \tag{2.5.67}$$

For $X_{GPS} \geq 0\,\text{m}$ the terrain database accuracy is no longer valid as the aircraft is flying over the runway and therefore the height and relative altitude measurement are identical. The theoretical radio altimeter sensor accuracy was validated on actual data as shown in figure 2.34.



(a) (b)

Figure 2.34: The radio altimeter accuracy is shown as a function of the estimated relative altitude of an aircraft for actual measurement data. The blue line shows the theoretical accuracy limits for the radio altimeter data and the black data points are a scatter plot of actual radio altimeter measurement data.

There are a few notable observations that can be made from figure 2.34. Firstly, the radio altimeter accuracy bounds in 2.34 (a) are a tight fit on the bottom bounds of the data, but appear to poorly fit the upper bounds. This is attributed to the offset introduced by the average terrain database error, μ_{terr}, and the conservative assumptions made when modelling. The second observation is that the radio altimeter data is densely distributed around the estimated relative altitude of the aircraft, except for the occasional non-linearity as indicated in figure 2.34 by the red circles and oval. These non-linearities are consistent across all simulation runs, therefore suggesting that they are linked to the non-linearities of the terrain database. This suggests that the radio altimeter is in fact much more accurate than the theoretical characteristics suggest. It should be noted that the theoretical accuracy supplied by Airbus is most likely the absolute maximum deviation that accounts for adverse weather conditions, different types of terrain etc. Should a larger and more diverse dataset be used, then the data might be a better fit. The last notable observation is the sharp increase in accuracy at a relative altitude of less than 15 m in figure 2.34 (b). This point coincides with the location of runway threshold, in other words $X_{GPS} = 0\,\text{m}$, and therefore beyond this point the terrain database errors can be ignored as the relative altitude and height are both with reference to the runway. Along a glide slope of 3° the aircraft would be at a true relative altitude of 14.98 m when $X_{GPS} = 0\,\text{m}$.

2.6 Sensor fault profiles

Airbus supplied a list of six recommended fault profiles:

- Oscillation : Sensor measurement oscillates sinusoidally around the relative altitude at which the fault occurred

$$z_i(t) = A \cdot \sin(2\pi\omega\Delta t) + z_{fault} \tag{2.6.1}$$

  where z_i is the sensor measurement associated with the i^{th} sensor, Δt is the time difference since fault occurred, z_{fault} is relative altitude at which fault occurs, A and ω are a random variables that vary per simulation.

- Jamming : sensor measurement remains at relative altitude at which fault occurred. Examples of real life failures that result in a jamming error are damage to the radio altimeter installation or water covering the fuselage around its antennas

$$z_i(t) = z_{fault} \tag{2.6.2}$$

- Bias : A fixed offset is imposed on a sensor measurement. A bias error on a radio altimeter, also known as bright spot, occurs when there is a stronger reflection from an object that is not necessarily directly below the aircraft. For example: vegetation below the aircraft returns a weak echo signal due to the diffused reflection. A house that is not directly below an aircraft, but still within the radio altimeter's area of illumination might return a much stronger reflection and therefore the distance to the house would register as the perceived height by the sensor

$$z_i(t) = z_{true}(t) + b \tag{2.6.3}$$

  where b is a random variable that varies per simulation.

- Runaway : Sensor measurement increases exponentially after the fault has occurred

$$z_i(t) = z_{true}(t) + e^{c\Delta t} \tag{2.6.4}$$

  where c is constant that determines the gradient of runaway.

- Non-Return-to-Zero : Sensor measurement will oscillate between some fixed positive and negative value after the fault has occurred. An example of a NRZ fault is an encoding error in the sensor electronics that results in a square wave oscillation

$$z_i(t) = A \cdot \text{sgn}(\sin(2\pi\omega\Delta t) + z_{fault}) \tag{2.6.5}$$

  where sgn() equals +1 or -1 based on the sign.

- Increased Noise : Sensor measurement gets an additive white noise component that noticeably increases the noise variance associated with a sensor. This could be due to extreme weather conditions such as hail or sensor failure that results in erroneous readings

$$z_i(t) = z_{true}(t) + \eta(t) \tag{2.6.6}$$

  where η is $\mathcal{N}(\text{o},\sigma^2)$.

Each sensor technology type can experience any of the faults profiles defined above. While certain fault profiles are more prevalent than others, it is possible that any of these fault profiles could occur and therefore need to be simulated.

2.7 Conclusion

In this chapter, the robust height estimation problem was conceptualised, and mathematical models were established for the aircraft motion, the sensors, and the terrain. The structure and nominal parameters for the sensor models were based on information sourced from literature, and the sensor parameters were then identified to fit a real dataset provided by Airbus. Fault models for six types of sensor faults were also defined. A simulation model based on the mathematical models established in this chapter will be used to generate a large dataset of representative sensor measurements containing both “no fault” and “fault” conditions. The next chapter will proceed to discuss the fault detection and isolation theory.

Chapter 3

Fault Detection and Isolation Theory

The necessary theory and literature on fault detection and isolation techniques is presented in this chapter. It is important that the reader understands all the necessary background information on fault diagnostic systems before these techniques can be applied to the problem of robust height estimation.

This chapter starts off with an introduction to the concept of fault diagnostics. An overview of the approaches to fault detection and isolation is presented with the focus being on investigating a variety of model-based and data-driven methods. A fault detection and isolation system automates the decision making process surrounding whether a fault has occurred and, if so, then identifies the source. Decision theory is presented next and discusses how to make optimal choices when it comes to identifying faults. The remainder and bulk of this chapter is dedicated to presenting the theory and literature behind the operation of model-based and data-driven methods.

3.1 Fault Diagnosis

Faults are defined as any undesired change that degrades the performance of a system [28]. These undesired changes are addressed by incorporating fault diagnostic algorithms in the system design that are capable of dealing with sensor, actuator or plant faults using a two staged process known as fault detection and isolation. Fault detection involves monitoring the system variables to determine whether there is a fault or the system is running normally [29]. When a sensor fault is detected, fault isolation is executed to identify the fault type and location. The approaches to fault detection and isolation can be divided into two separate streams, namely hardware redundancy and analytical redundancy [30], as illustrated in figure 3.1.

The traditional approach to fault diagnosis is hardware redundancy [30] which involves the use of numerous sensors that all measure the same critical variable. Faults are isolated through consistency checking and majority voting. The Airbus A380 relative altimetry system is an example of hardware redundancy, where three radio altimeters are used to estimate the aircraft height [31]. The median of the three sensor measurements is regarded as the truth. Taking the mean of the three sensors would be skewed by the presence of an outlier, while the median is a robust measure of central tendency. Hardware redundancy is advantageous as it is simple and robust, but is not always practically feasible due to price and space constraints [30].

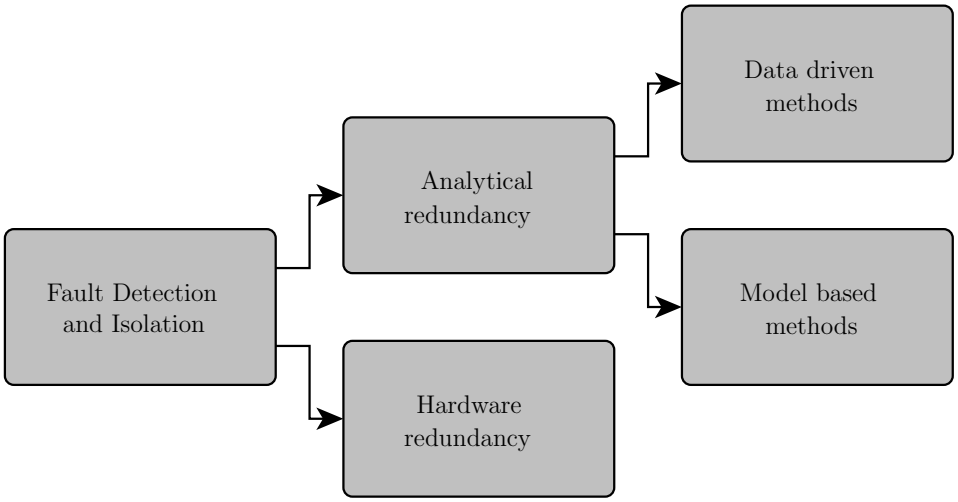


Figure 3.1: Overview of the approaches to fault detection and isolation.

Analytical redundancy replaces redundant sensors with methods that use expert knowledge or data analysis to perform fault diagnosis. This is achieved by either deriving an explicit mathematical model of the target system (model-based methods) or classifying sensor failures based on historic datasets (data-driven methods).

3.1.1 Model-Based

The implementation of a model-based method focuses on the formation of residuals that are given by difference between the model estimate and plant output as seen in figure 3.2.

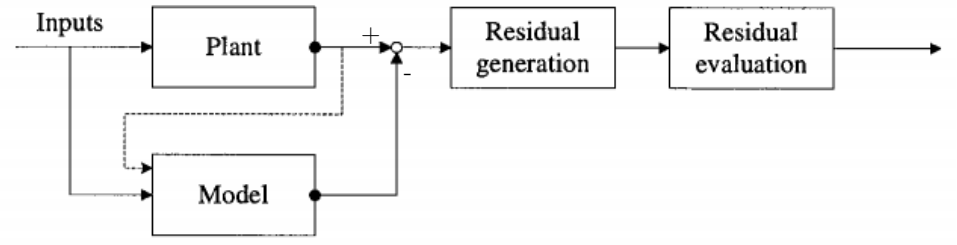


Figure 3.2: Model based approach to fault detection and isolation [29].

Under normal conditions the model will accurately mimic plant behavior and the residuals will be close to zero. Conversely when a fault is introduced the model is a poor representation and results in large residuals. Fault detection and isolation will therefore be performed by evaluating the residuals and alarms generated accordingly. Model based techniques are advantageous as they are capable of accurately and quickly detecting faults [29]. However they do require prior knowledge about the system dynamics, which is not always readily available.

3.1.2 Data-Driven

Data-driven methods form the second group of analytical redundancy approaches. These methods do not require any information on system dynamics, but rather rely on historic process data to derive structure that is used to determine the category (class) of a new observation (data point) [32]. The process of assigning a data point to a specific category is called classification. Data driven methods cover a broad range of approaches that can be divided into one of two categories, namely supervised and unsupervised machine learning techniques. Supervised techniques provide the learning algorithm with a labeled dataset and it is expected to determine a means of classifying unseen data points accordingly [33]. It uses training and prior knowledge to derive the decision boundaries that can be used to optimally separate the respective classes. Conversely unsupervised techniques expect the learning algorithm to form natural clusters when presented with an unlabeled dataset [34].

There are three different categories of supervised learning, namely novelty detection, binary and multi-class classification [7]. Novelty detectors are trained with a dataset consisting of a single class. Unseen data points are therefore categorised by their similarity to the known class. Binary classification considers data with two classes, where new data points are categorised based on their similarity with one of the two classes. Similarly, multi-class classification algorithms assign new data points to one of the many classes based on some similarity measure. Novelty detection and binary classification algorithms are used for fault detection, while multi-class classifiers have the capability to identify which sensor is faulty.

3.2 Decision Theory

The purpose of fault detection and isolation systems is to automate the decision making process surrounding which operating mode or discrete status is most likely active. The operating modes are represented by K discrete output classes (C_k) that each describe a different system state for the collective group of sensors, actuators or plant. For example a discrete status (C_1) could represent a healthy system with no faulty sensors, while (C_2) represents a failure with a specific sensor. Each discrete class has a conditional probability, $p(C_k|x)$, associated with it that reflects the likelihood of it being correct. Decision theory ensures that the optimal decisions are made given the respective probabilities. Bayes' theorem is one manner used to quantify the effects of observed data (x) on the conditional probability for each class, posterior, as follows:

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} \tag{3.2.1}$$

$$\propto p(x|C_k)p(C_k) \tag{3.2.2}$$

where $p(C_k)$ (prior) is the prior probability for class C_k, $p(x)$ (evidence) is constant for a specific x and $p(x|C_k)$ (likelihood) is quantified by assuming some likelihood model for the observed data [35].

The aim of maximizing the probability of assigning x to the correct class is achieved by selecting the highest posterior probability such that:

$$C_k = \arg\max p(x|C_k)p(C_k) \tag{3.2.3}$$

If the prior is equivalent for all classes ($p(C_k) = \frac{1}{K}$), then the class assignment equation 3.2.3 reduces to finding the maximum of the likelihood:

$$C_k = \arg\max p(x|C_k) \tag{3.2.4}$$

Another approach to the decision making process is through the use of decision boundaries or surfaces to differentiate between classes. Decision boundaries are chosen such that they minimise the percentage of misclassifications as illustrated in figure 3.3. Without prior knowledge of the distribution of the data a means is needed to determine the optimal decision boundary. This is achieved through the use of a cost function that is a singular measure of the overall loss incurred when using a specific decision boundary [36]. The optimal location is determined during training by selecting the decision boundary that minimises the total loss and thereby maximises the percentage of correct classifications.

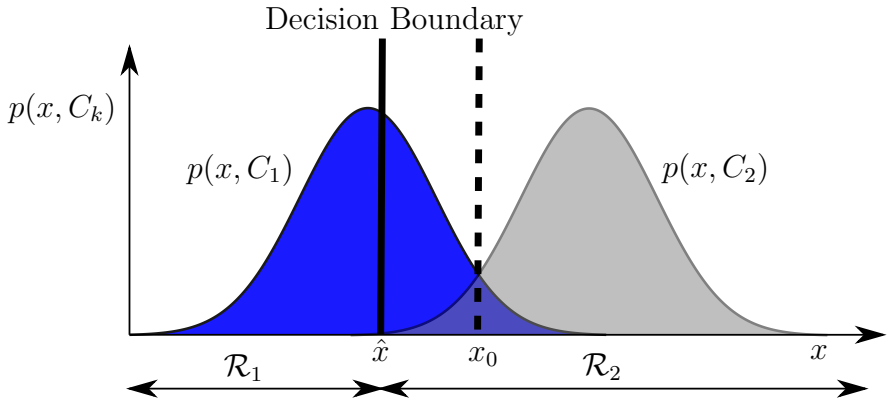


Figure 3.3: Illustration depicting the conditional probabilities $p(x|C_k)$, shown as a function of x, and the decision boundary $x{=}\hat{x}$. Values for $x \geq \hat{x}$ belong to region $\mathcal{R}_2$ and are assigned to class C_2, while values $x < \hat{x}$ belong to region $\mathcal{R}_1$ and are assigned to C_1. The location of the optimal decision boundary that will minimise the percentage of misclassifications corresponds to location x_0 [36].

3.3 Model Based Methods

The theory and literature behind the model-based fault detection and isolation techniques is discussed in this section. Table 3.1 provides a summary of the approaches that are investigated.

Model Based Method	**Approach Type**
Robust Kalman filter	Robust statistics and optimal estimation
Bank of Kalman filters	Probabilistic combination of optimal estimators

Table 3.1: Summary of model-based approaches.

The Kalman Filter (KF) is a classical optimal estimation method and will form the foundation of both the Robust KF and Bank of KF approaches. The section will therefore

start off by introducing the necessary background and notation for a traditional Kalman filter, before the Robust KF and Bank of KF approaches are investigated.

3.3.1 Traditional Kalman Filter

The Kalman filter was invented by Kalman and Bucy in 1960. It is a mathematical model that filters signals with a certain degree of noise and uncertainty to give the most likely parameter estimates. The KF is a type of Bayes' filter that is applicable to time-varying linear systems [37] that have been reduced to an elegant set of matrix equations based on two key assumptions. The first is that the system whose state is being estimated can be modelled by a linear system that is described by the following state space equations [37]:

$$\vec{X}_k = F\vec{X}_{k-1} + G\vec{U}_{k-1} + W_k \tag{3.3.1}$$

$$\vec{Y}_k = H\vec{X}_k + V_k \tag{3.3.2}$$

where $\vec{X}_k$ and $\vec{Y}_k$ is the system state and measurement vector at time k, F is the state transition matrix, G is the input matrix, $\vec{U}_k$ is the control vector, H is the output matrix, W_k is system noise and V_k is the measurement noise.

The second assumption is that the process noise can be modelled as zero-mean Gaussian white noise with a probability distribution given by $W_k \sim \mathcal{N}(0, Q_k)$, where Q_k is the covariance matrix. Similarly, the measurement noise can be modelled as zero-mean Gaussian white noise given by probability distribution $V_k \sim \mathcal{N}(0, R_k)$, where R_k is the covariance matrix.

The Kalman filter is a two step recursive process. The first step is the control update in which the state estimate at time step k is propagated from the previous state using the linear state space model. This is also known as the "prediction" as the system model is used to estimate the state, ${X_k}^-$. The error covariance matrix, P_k, is also updated as there is more uncertainty in the state estimate. The second step is the measurement update in which the state estimate is corrected based on evidence presented by actual measurements, $\vec{Y}_k$. The correction is determined by the Kalman gain, L_k, which determines whether the filter is going to place more emphasis on the measurements or the state prediction. The Kalman gain is controlled by the ratio between the process error and measurement noise covariance. A larger measurement noise will result in smaller gains and the Kalman filter placing more emphasis in the model. A larger process noise will result in larger gains and the Kalman filter placing more emphasis on the measurements. The Kalman filter iteratively updates the state belief distribution with the state estimate being given by the mean of the distribution. The equations that govern the operation of a Kalman filter are given by [37]:

Control update:

$$\vec{X_k}^- = F\vec{X}^+_{k-1} + G\vec{U}_{k-1} \tag{3.3.3}$$

$$P^-_k = Q_k + FP^+_{k-1}F^T \tag{3.3.4}$$

Measurement update:

$$S_k = R_k + HP_k^- H^T \tag{3.3.5}$$
$$L_k = P_k^- H^T (S_k)^{-1} \tag{3.3.6}$$
$$P_k^+ = (I - L_k H) P_k^- \tag{3.3.7}$$
$$\vec{X_k}^+ = \vec{X_k}^- + L_k(\vec{Y_k} - H\vec{X_k}^-) \tag{3.3.8}$$

3.3.2 Robust Kalman Filter

3.3.2.1 Overview

The traditional KF is used to fuse redundant sensor measurements based on the assumption that all sensors are faultless and available. The Robust KF is an extension of a traditional KF that is designed to be insensitive to outliers. This is achieved through the assistance of robust statistics that establishes a gating region around the state estimate. The residual analysis step compares the measurement residual against the theoretical limits of sensor accuracy at each iteration.

$$Y_k^{(i)} - H\vec{X}_k \leftarrow \begin{cases} Y_k^{(i)} - H\vec{X}_k & \text{for } |Y_k^{(i)} - H\vec{X}_k| \leq \Delta_i \\ 0 & \text{for } |Y_k^{(i)} - H\vec{X}_k| > \Delta_i \end{cases} \tag{3.3.9}$$

where Δ_i is the limit of accuracy for the i^{th} sensor.

Any sensor measurement that falls outside of this gating region is considered to be an outlier and its measurement residual is excluded from the correction of the state estimate given by equation 3.3.8. The estimated state update is therefore isolated from the fault in this manner. The Robust KF therefore has an identical structure to that of the traditional KF, except for the additional logic checks performed before state estimate correction.

3.3.2.2 Advantages and Disadvantages

The advantage of the Robust KF is its simplicity of implementation. The disadvantage of the Robust KF is that it requires prior knowledge in the form of sensor noise, theoretical limits of sensor accuracy and a model of vehicle dynamics.

3.3.2.3 Configurable Parameters

The following are configurable parameters for the Robust KF framework:

- σ_w - process noise variance
- Δ_i - theoretical limit of sensor accuracy

The process noise covariance matrix, Q_k, is a tuning parameter that is associated with the noise of the process and will determine the rate of convergence and the Kalman gains. Δ_i is the theoretical sensor accuracy that acts as a gating region around the state estimate by establishing a decision boundary that determines whether the measurement is faulty or not.

3.3.3 Bank of Kalman Filters

3.3.3.1 Overview

The Bank of KF approach to fault detection and isolation (FDI) establishes $m+1$ independent Kalman filters in parallel, where m is the number of sensors being monitored as seen in figure 3.4.

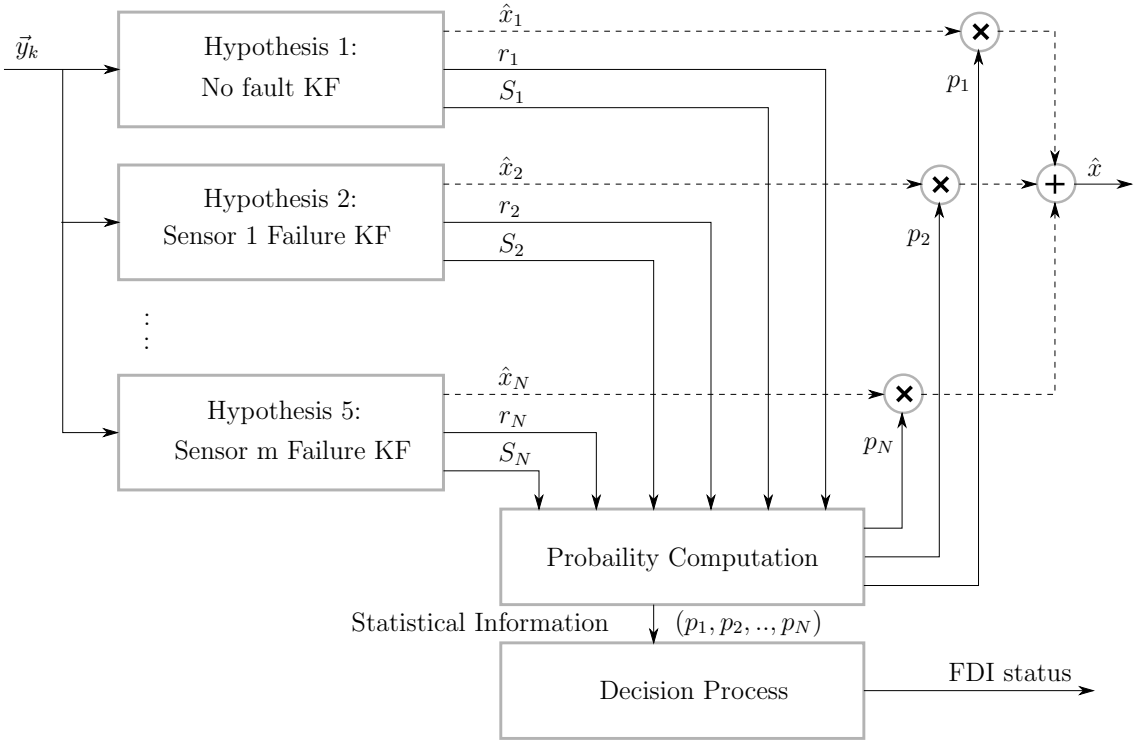


Figure 3.4: The FDI architecture for a Bank of Kalman filters approach to fault detection and isolation.

The design of individual filters is based on different hypotheses. The first KF assumes that all the sensors are functioning correctly and are therefore incorporated in the KF design. The remaining Kalman filters each assume the failure of a different sensor that is then excluded from the KF design. When a failure occurs the output of the KF that excludes that sensor will be robust to that specific technology failure.

An automated decision making process is needed to determine which of the hypotheses best represents the current observations. A Gaussian Bayes Classifier is used to determine the likelihood associated with respective Kalman filters [38]. This is a suitable technique when dealing with Kalman filters as the measurement residual (r_i) and its covariance matrix (S_i) give useful information regarding the accuracy of the model. A likelihood function is constructed from the above parameters and a probability can be coupled to each hypothesis denoted by θ. This approach is adaptive in the sense that the final state estimate is weighted according to the probability associated with each scenario. Bayes' theorem is used to determine the probability of an outcome being true given the evidence. The possible outcomes are the list of hypotheses, while the sequence of previous sensor values, $\vec{Y}_k = [\vec{y}_k, \vec{y}_{k-1}, ..., \vec{y}_0]$, forms part of the evidence. The derivation for the probability

associated with θ_i is given by:

$$p(\theta = \theta_i \mid \vec{Y}_k) = \frac{p(\theta = \theta_i)p(\vec{Y}_k \mid \theta = \theta_i)}{p(\vec{Y}_k)} \tag{3.3.10}$$

It is assumed that the hypotheses are mutually exclusive as only one sensor technology can fail at a given instant. The chain rule is used to expand the denominator of equation 3.3.10 as follows:

$$p(\vec{Y}_k) = p(\vec{Y}_k \mid \theta = \theta_1)p(\theta_1) + ... + p(\vec{Y}_k \mid \theta = \theta_N)p(\theta_N) \tag{3.3.11}$$

$$= \sum_{j=1}^{N} p(\vec{Y}_k \mid \theta = \theta_j)p(\theta = \theta_j) \tag{3.3.12}$$

where N is the total number of hypotheses.

The probability of the i^{th} hypothesis occurring at time step k is now given by the fraction:

$$p(\theta = \theta_i \mid \vec{Y}_k) = \frac{p(\vec{Y}_k \mid \theta = \theta_i)p(\theta = \theta_i)}{\sum_{i=1}^{N} p(\vec{Y}_k \mid \theta = \theta_j)p(\theta = \theta_j)} \tag{3.3.13}$$

The current expression requires values and covariances for a sequence of measurements to be known. Transforming this expression into a recursive equation will simplify the computational complexity as the probabilities can be updated at each time step based on current measurements and previous probabilities. Separating the sequence of measurements such that $[\vec{y}_k, \vec{Y}_{k-1}]^T$ and using the chain rule, the probability for $\vec{Y}_k$ is given by [39]:

$$p(\vec{Y}_k \mid \theta = \theta_i) = p([\vec{y}_k, \vec{Y}_{k-1}]^T \mid \theta = \theta_i) \tag{3.3.14}$$

$$= p(\vec{y}_k \mid \vec{Y}_{k-1}, \theta = \theta_i)p(\vec{Y}_{k-1} \mid \theta = \theta_i) \tag{3.3.15}$$

Equation 3.3.15 can be further simplified by assuming that $p(\vec{y}_k \mid \vec{Y}_{k-1}, \theta = \theta_i) = p(\vec{y}_k \mid \theta = \theta_i)$ and $\alpha_i(k) = p(\vec{Y}_k \mid \theta = \theta_i)$. For the simplicity of notation equation 3.3.15 is rewritten:

$$\alpha_i(k) = p(\vec{y}_k \mid \theta = \theta_i)\alpha_i(k-1) \tag{3.3.16}$$

Now equation 3.3.13 can be updated by substituting equation 3.3.16 giving:

$$p(\theta = \theta_i \mid \vec{Y}_k) = \frac{\alpha_i(k)p(\theta = \theta_i)}{\sum_{i=1}^{N} \alpha_j(k)p(\theta = \theta_j)} \tag{3.3.17}$$

$$= \frac{p(\vec{y}_k \mid \theta = \theta_i)\alpha_i(k-1)p(\theta = \theta_i)}{\sum_{j=1}^{N} p(\vec{y}_k \mid \theta = \theta_j)\alpha_j(k-1)p(\theta = \theta_j)} \tag{3.3.18}$$

A formula is now needed to determine a value for $p(\vec{y}_k \mid \theta = \theta_i)$. Knowing that one of the fundamental assumptions associated with the Kalman filter is that measurements are normally distributed, a probability density function for a Gaussian distribution can be used to quantify $p(\vec{y}_k \mid \theta = \theta_i)$. The equation for a Gaussian distribution is given by equation 3.3.19.

$$f(\vec{y}_k \mid \theta = \theta_i) = \frac{1}{(2\pi)^{\frac{m_i}{2}} \mid \Sigma \mid^{1/2}} \exp(-\frac{1}{2}(\vec{y}_k - \mu_k)^T(\Sigma^{-1})(\vec{y}_k - \mu_k)) \tag{3.3.19}$$

where m_i is the dimension of the measurement vector for the i^{th} hypothesis, Σ is the covariance matrix, and μ_k the mean.

By substituting the mean value for the expected measurement value $H\vec{X}_k^-$ equation 3.3.19 can be rewritten in terms of the measurement residual and its covariance as follows:

$$\vec{r}_k = \vec{y_k} - H\vec{X}_k^- \tag{3.3.20}$$

$$S_k = R_k + HP_k^-H^T \tag{3.3.21}$$

$$f(\vec{y}_k \mid \theta = \theta_i) = \frac{1}{(2\pi)^{\frac{m_i}{2}} \mid S_k \mid^{1/2}} \exp(-\frac{1}{2}(\vec{r}_k)^T(S_k^{-1})(\vec{r}_k)) \tag{3.3.22}$$

where S_k is the measurement residual covariance.

Equation 3.3.18 is now updated using the distribution function to quantify the probability for the i^{th} hypothesis as follows:

$$p(\theta = \theta_i \mid \vec{Y}_k) = \frac{f(\vec{y}_k \mid \theta = \theta_i)\alpha_i(k-1)p(\theta = \theta_i)}{\sum_{i=1}^{N} f(\vec{y}_k \mid \theta = \theta_j)\alpha_j(k-1)p(\theta = \theta_j)} \tag{3.3.23}$$

The FDI status is updated based on the hypothesis that maximises the probability function given by equation 3.3.23 above, while the estimated states can be calculated as a weighted sum of the probabilities associated with each hypothesis. The more likely a hypothesis is, the larger its probability will be and therefore it will provide a greater contribution towards the overall state update.

$$\hat{x} = \sum_{i=1}^{N} x_i(k)p_i(k) \tag{3.3.24}$$

The final decision process to determine the FDI status makes use of a fixed threshold to improve the classifier's performance. The Gaussian Bayes Classifier is based on many assumptions and the use of a probability and consistency threshold brings stability to the output of the classifier [38]. Firstly, a hypothesis is only classified as being active once its probability has exceeded a predetermined probability threshold. Secondly, the consistency of the decision block is monitored and only updates the sensor status once a hypothesis has been active for longer than 2 s.

3.3.3.2 Advantages and Disadvantages

The drawbacks of the Bank of Kalman filters approach is that it is computationally more intensive than the Robust KF as it requires multiple Kalman filters in parallel as well as prior knowledge in the form of sensor noise and prior probabilities associated with each hypothesis.

3.3.3.3 Configurable Parameters

The following are configurable parameters for the Bank of KF framework:

- σ_w - process noise variance
- $P(\theta = \theta_i)$ - prior probability associated with i^{th} sensor status
- P_{thresh} - probability threshold to identify active hypothesis

The process noise covariance matrix Q_k is a tuning parameter that is associated with the noise of the process and will effect the rate of convergence and the Kalman gains. The prior probability $P(\theta = \theta_i)$ is set to $\frac{1}{N}$ without any prior knowledge of the probability associated with each individual hypothesis.

3.4 Overview of Data Driven Methods

This section gives an overview of the data-driven techniques investigated for fault detection and isolation. The breakdown of the various approaches is shown in figure 3.5.

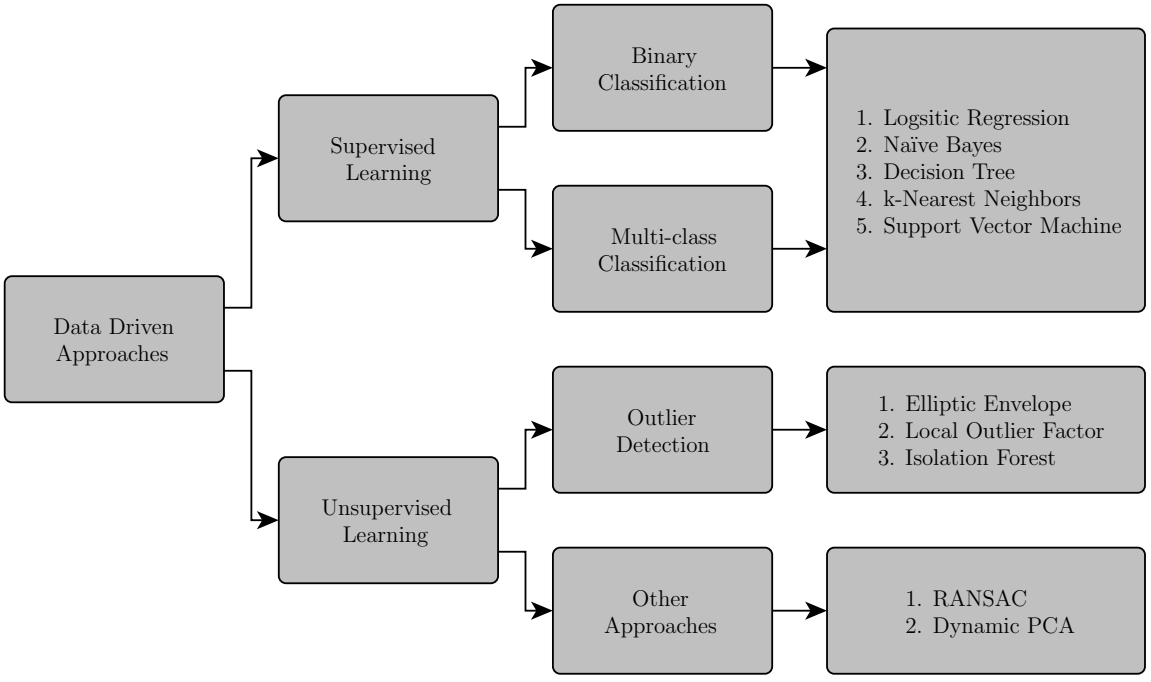


Figure 3.5: Overview of the data-driven approaches to fault detection and isolation.

The data-driven approaches can be separated into two separate streams, namely supervised and unsupervised machine learning techniques. The supervised approaches consist of both binary and multi-class classifiers, while the unsupervised approaches consist of data-driven outlier detection algorithms and other novel approaches found in literature that are applicable to fault detection and isolation.

The binary and multi-class algorithms are trained offline on a wide variety of measurement

data and can then be applied to an online stream of measurements. A similar approach is applied to the outlier detection methods, except that they are trained on unlabeled datasets that consist mostly of "normal" or "no-fault" data points with a few scattered "faults" or "outliers". They are expected to identify these outliers based on their similarity to the majority of the data. The binary classification and outlier detection techniques are used for fault detection. They are capable of differentiating between one of two possible classes, namely fault or no-fault. The multi-class classifier is used for fault isolation and is responsible for identifying which sensor is faulty.

3.5 Binary Classification

Binary classifiers are investigated as a means of detecting when a fault occurs. Table 3.2 provides a summary of the binary classification approaches that are investigated. A wide range of algorithms are considered that vary both with approach and complexity.

Classifier Name	**Approach Type**
Logistic Regression	Linear classification
Naïve Bayes	Probabilistic classification
Decision Tree	Rules-based classification
k-Nearest Neighbors	Non-parametric instance based classification
Support Vector Machine	Distance-based classification

Table 3.2: Summary of binary classifiers.

A machine learning library can be used to implement the algorithms listed in table 3.2. The parameters and implementation of the algorithms are specific to the scikit-learn machine learning library. The binary classification algorithms are trained on equal percentages of data from both the fault and no-fault class to ensure that the optimal unbiased decision boundary is found.

3.5.1 Logistic Regression

3.5.1.1 Overview

Logistic Regression fits an optimal hyperplane that forms a decision boundary and is used to differentiate between two classes based on their location relative to this hyperplane as illustrated in figure 3.6. A hypothesis or sigmoid function predicts the binary label assigned to a data point. The independent features that describe a data point are combined using a weighted vector ($\vec{w}$), before a sigmoid function manipulates the sum of the linearly combined independent variables to give an estimated probability value between zero and one. This hypothesis function is defined by [36]:

$$h(\vec{x}, \vec{w}, b) = P(y = 1|\vec{x}, \vec{w}, b) = \frac{1}{1 + e^{-(\vec{w}^T \vec{x} + b)}} \tag{3.5.1}$$

where $\vec{w}$ is the a vector of regression coefficients for each feature with bias variable b.

The decision boundary for the class prediction is given by the set of variables for which $P(y = 1|\vec{x}, \vec{w}, b) = P(y = 0|\vec{x}, \vec{w}, b) = 0.5$. This corresponds to the optimal hyperplane:

$$\vec{w}^T\vec{x} + b = 0 \tag{3.5.2}$$

The logistic regression algorithm attempts to find the optimal regression coefficients such that the hypothesis function is a good fit for the data.

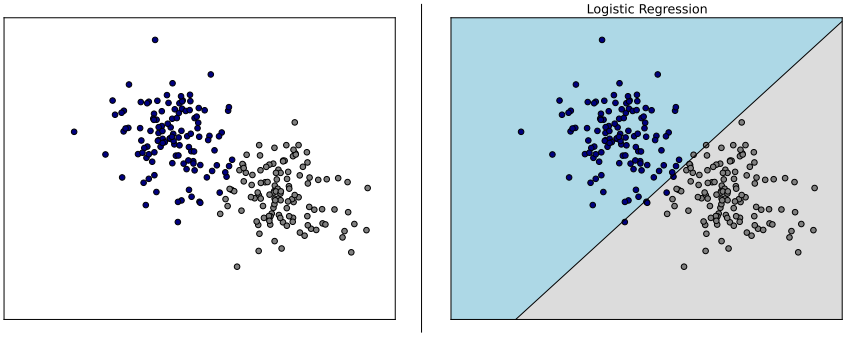


Figure 3.6: The decision boundary for two-dimensional example dataset classified using a Logistic Regression classifier.

3.5.1.2 Training

No closed form solution exists to estimate the optimal model parameters [32]. Instead a maximum likelihood approach is used to determine optimal regression coefficients using a method called gradient descent. The optimal parameters are those that minimise a convex cost function over the set of training data points. The algorithm is summarised as an iterative process in which small "steps" are taken along the direction of the gradient vector. With each iteration the model parameters are updated until the global optimum is reached. The maximum likelihood cost function for the training dataset with N data points is given by:

$$j(\vec{w}) = \prod_{i=1}^{N} h(\vec{x}_i, \vec{w}, b)^{y_i}[1 - h(\vec{x}_i, \vec{w}, b)]^{(1-y_i)} \tag{3.5.3}$$

where y_i is the label associated with the i^{th} data point.

An error function is defined by taking the negative logarithm of the likelihood function 3.5.3 above.

$$J(\vec{w}) = -\log[j(\vec{w})] = -\sum_{i=1}^{N}\Big[y_i \log[h(\vec{x}_i, \vec{w}, b)] + (1 - y_i)\log[1 - h(\vec{x}_i, \vec{w}, b)]\Big] \tag{3.5.4}$$

The gradient of the error function with respect to $\vec{w}$ is given by [36]:

$$\nabla L(\vec{w}) = \sum_{i=1}^{N}(y_i - h(\vec{x}_i, \vec{w}, b))\vec{x}_i \tag{3.5.5}$$

Regularization of the regression coefficients $\vec{w}$ is necessary to prevent over fitting. A penalty parameter λ is imposed on the regression coefficients that limits them from becoming too large. Cost function 3.5.4 is adapted to account for regularization as follows [36]:

$$J(\vec{w}) = -\sum_{i=1}^{N} \left[y_i \log(h(\vec{x}_i, \vec{w}, b)) + (1 - y_i) \log(1 - h(\vec{x}_i, \vec{w}, b)) \right] + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2 \qquad (3.5.6)$$

where d is the dimension corresponding to the number of features.

3.5.1.3 Classification

New data points are classified based on a probability threshold. Data points that belong to the positive class correspond with an estimated probability $P(y = 1|\vec{x}, \vec{w}, b) \geq 0.5$ and find themselves above the decision hyperplane such that $\vec{w}^T\vec{x} + b \geq 0$. Similarly, for the negative class $P(y = 0|\vec{x}, \vec{w}, b) > 0.5$ and these data points find themselves below the decision hyperplane.

3.5.1.4 Advantages and Disadvantages

Logistic Regression is easy to train and implement with a limited computational complexity required to classify new data points. It also gives the conditional probabilities which is a useful property to assess the confidence associated with a classification. The drawback of Logistic Regression is that it is prone to over-fitting if regularization is not used correctly [7] and can be a poor fit for the data due to the linear decision hyperplane that is unable to adapt to non-linear decision boundaries without prior preprocessing or the use of kernel functions.

3.5.1.5 Configurable Parameters

The configurable parameters for Logistic Regression are as follows:

- **penalty** - regularization algorithms used for regression coefficients
- **c** - inverse regularization parameter
- **tol** - set tolerance for stopping criteria
- **max_iter** - maximum number of iterations used
- **solver** - set optimization algorithm in training

The scikit-learn library supports L1 and L2 regularization. The L1 regularization takes the sum of the absolute regression coefficients, while L2 penalises by using the squared sum of regression coefficients. The effect of the squared term is a harsher penalty that shrinks the regression coefficients to small non-zero values in comparison to the L1 regularization that shrinks most of the coefficients, while the most important ones remain unchanged [40]. The default L2 optimization was chosen.

The tolerance for the stopping criteria is set to stop the training algorithm and thereby ensure that unnecessary computational power is not wasted for a small improvement in accuracy. Similarly, the training algorithm can be halted by setting the maximum number

of iterations.

The are a variety of gradient descent optimization algorithms available such as the newton-cg (conjugate gradient), lbfgs (Limited memory Broyden-Fletcher-Goldfarb-Shanno), liblinear and sag (stochastic average gradient). The default optimization algorithm is liblinear.

3.5.2 Naïve Bayes

3.5.2.1 Overview

The Naïve Bayes classifier is a probabilistic approach that is based on Bayes' theorem [32]. The class conditional probabilities are assumed to be modelled by a known distribution and are determined for each feature axis during training. These conditional class distributions are used to calculate the probability of a data point belonging to a specific class. The Naïve Bayes algorithm derives its name from the naïve assumption of feature independence. Figure 3.7 depicts the conditional probabilities for the class labels ($Y = [y1 = +1, y2 = -1]$) along the feature axes.

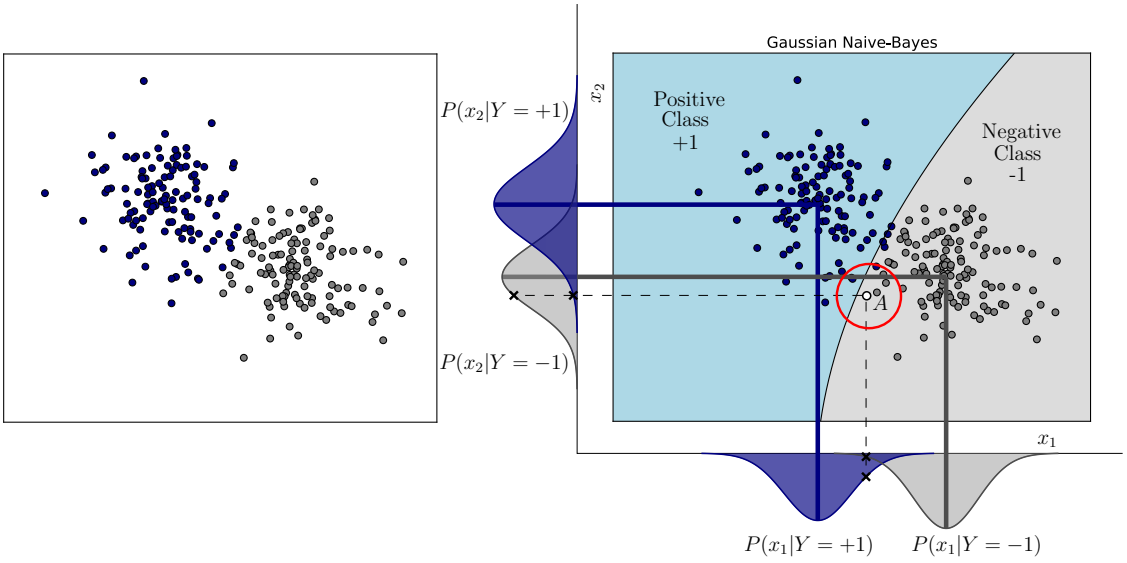


Figure 3.7: The Naive Bayes decision boundary for an example two-dimensional dataset. The conditional probabilities are assumed to be drawn from a Gaussian distribution.

3.5.2.2 Training

The training of a Gaussian Naïve Bayes classifier involves determining the mean and standard deviation of the axes-independent Gaussian distributions that describe the spread of the two classes as follows:

$$\mu_{ij} = E[X_i|Y = y_j] \tag{3.5.7}$$

$$\sigma_{ij} = E[(X_i - \mu_{ij})^2|Y = y_j] \tag{3.5.8}$$

The training process fits the conditional probabilities to the dataset given the class [7]. Figure 3.7 illustrates the resultant conditional probability distributions [$p(x_1|Y = +1)$, $p(x_1|Y = -1)$, $p(x_2|Y = +1)$, $p(x_2|Y = -1)$] after training.

3.5.2.3 Classification

New data points are assigned to the class which yields the maximum conditional probability. Bayes' theorem is used to determine the maximum conditional probability. This theorem states that the posterior probability of belonging to a class y_j is given by:

$$P(Y = y_j|\vec{X}) = \frac{P(Y = y_j)P(\vec{X}|Y = y_j)}{P(\vec{X})} = \frac{P(Y = y_j)P(x_1, x_2, ..., x_d|Y = y_j)}{P(x_1, x_2, ..., x_d)} \tag{3.5.9}$$

Through the use of the chain rule and under the naïve assumption that the features that make up a dataset are independent, equation 3.5.9 can be reduced to:

$$P(Y = y_j|x_1, x_2, ...x_d) = \frac{P(Y = y_j)\prod_{i=1}^{d} P(x_i|Y = y_j)}{P(x_1, x_2, ..., x_d)} \tag{3.5.10}$$

$$= \frac{1}{\eta}P(Y = y_j)\prod_{i=1}^{d} P(x_i|Y = y_j) \tag{3.5.11}$$

where $P(x_1, x_2, ..., x_d)$ is constant for a specific feature vector $\vec{X}$ and can be represented by scaling factor η.

The use of Bayes' theorem is illustrated through the classification of an unlabeled data point, A, in figure 3.7. The conditional probability is calculated for both classes:

$$P(X = A|Y = +1) = P(x_1 = A|Y = +1)P(x_2 = A|Y = +1) \tag{3.5.12}$$

$$P(X = A|Y = -1) = P(x_1 = A|Y = -1)P(x_2 = A|Y = -1) \tag{3.5.13}$$

The probability of belonging to a class given the data point A can be determined using equation 3.5.11:

$$P(Y = +1|X = A) = \frac{1}{\eta}P(X = A|Y = +1)P(Y = +1) \tag{3.5.14}$$

$$P(Y = -1|X = A) = \frac{1}{\eta}P(X = A|Y = -1)P(Y = -1) \tag{3.5.15}$$

The unlabeled data point is assigned to the class that results in the maximum conditional probability between equation 3.5.14 and 3.5.15. The unlabeled data point is therefore assigned to the negative class. Equation 3.5.16 is expanded to determine the class for a multidimensional and multi-class problem as follows:

$$Y = \arg\max_{y_j} \prod_{i=1}^{d} P(x_i|Y = y_j)P(Y = y_j) \tag{3.5.16}$$

3.5.2.4 Advantages and Disadvantages

The advantage of the Naïve Bayes algorithm is its simplicity to train and implement. The drawback is that the naïve approach of feature independence is not a valid assumption in most applications and will therefore result in poor performance.

3.5.2.5 Configurable Parameters

There are no configurable parameters for the Gaussian Naïve Bayes algorithm.

3.5.3 Decision Tree

3.5.3.1 Overview

Decision Trees provide a structured approach in the form of a sequential decision process. A Decision Tree consists of nodes and branches, where each node represents an if-else test applied to a feature and the branches representing the possible outcomes. A decision tree is best understood visually as illustrated in figure 3.8.

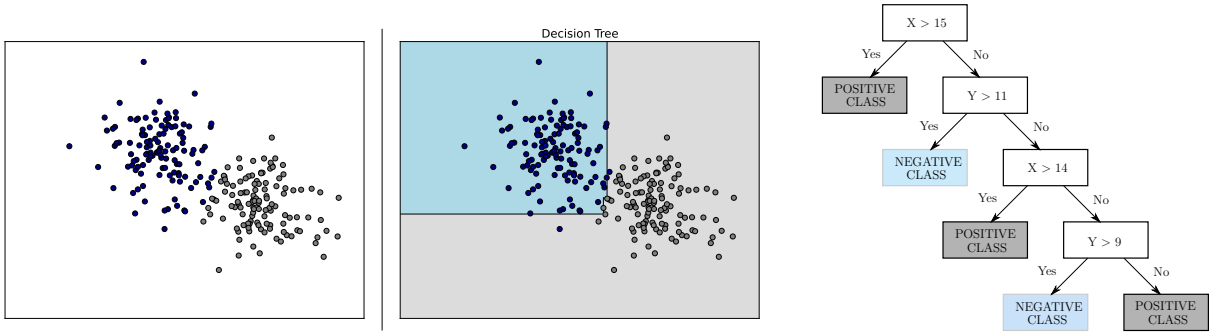


Figure 3.8: The Decision Tree boundary for an example two-dimensional dataset. The sequential decision tree shows how the classification boundary is formed. Each rectangle represents a node and the branches are represented by arrows.

3.5.3.2 Training

The Classification and Regression Tree (CART) algorithm uses the training data to grow a binary decision tree. This "tree growing" methodology will recursively split the data presented at a parent node into two children nodes that will eventually terminate in a leaf node and be assigned to a class. The fundamental aim of the training algorithm is to create a compact tree with as few nodes as possible. This is achieved by evaluating the information gain and choosing the splitting feature and value that will maximise the quality of split [41]. Before the information gain equation can be defined, two impurity metrics need to be explained. These are the Gini and Entropy impurity metrics. The Gini impurity is the expected error rate at a node if the class label is randomly selected from the distribution at that node [34]. The Gini index H_G for the positive class (P) and the negative class, (N), with class probability $p(\mathrm{P})$ and $p(\mathrm{N})$ respectively is defined as

$$H_G(\mathrm{P,N}) = 1 - \left(p(\mathrm{P})^2 + p(\mathrm{N})^2\right) \tag{3.5.17}$$

Similarly, the entropy impurity H_E is defined as

$$H_E(\mathrm{P,N}) = -p(\mathrm{P}) \log p(\mathrm{P}) - p(\mathrm{N}) \log p(\mathrm{N}) \tag{3.5.18}$$

The information gain or change in impurity ΔIG between the parent, S, and the left/right children nodes, S_L and S_R, is defined as:

$$\Delta IG(S) = H_{[]}(\mathrm{P,N}) - p(S_L)H_{[]}(\mathrm{P}_L, \mathrm{N}_L) - p(S_R)H_{[]}(\mathrm{P}_R, N_R) \tag{3.5.19}$$

where $H_{[]}$ is either the Gini or Entropy impurity, and $p(S_L)$ and $p(S_R)$ are the fraction of data points at parent S that will be assigned to the children nodes respectively.

The CART algorithm selects a random feature and examines the information gain by varying the split value over the feature range [42]. The split value that maximises the information gain is chosen and CART algorithm is repeated on the next layer in the tree. Once the tree is fully grown, each branch will terminate on a pure leaf node. To prevent over-fitting, a grown tree is pruned to allow it to generalise better to new examples. This is achieved using predefined parameters discussed in the parameters section.

3.5.3.3 Classification

New data points are classified by following the binary tree of if-else statements until a leaf node is reached. The new data point is assigned the same label as that of the leaf node.

3.5.3.4 Advantages and Disadvantages

The benefits of a Decision Tree is its robustness, interpretability and ease of understanding. Once trained, it requires very little memory and is computationally efficient at evaluating new data points. Drawbacks are that it is prone to over-fitting and will require a deep tree to learn the complex decision boundaries.

3.5.3.5 Configurable Parameters

The following are configurable parameters for the Decision Tree classifier:

- **criterion** - choice of impurity measure
- **max_depth** - maximum depth of the tree
- **min_samples_split** - minimum number of samples required to split a node
- **max_samples_leaf** - maximum number of samples required to be at a leaf
- **min_impurity_decrease** - minimum required change in impurity between parent and children nodes to ensure that a split will occur

The choice of impurity measure is between the Gini (default) and Entropy metric. The remainder of parameters are used to prune the tree. These parameters can be set individually or a combination of them can be used. The default for each of these parameters is set to produce a fully grown tree.

3.5.4 k-Nearest Neighbors

3.5.4.1 Overview

The k-Nearest Neighbor (k-NN) algorithm is a conceptually simple approach to classification. A new unlabeled data point is classified by assigning it the majority label associated with its k nearest neighbors. These nearest neighbors are the k closest data points based on Euclidean distance. The resultant decision boundary is illustrated on an example dataset in figure 3.9.

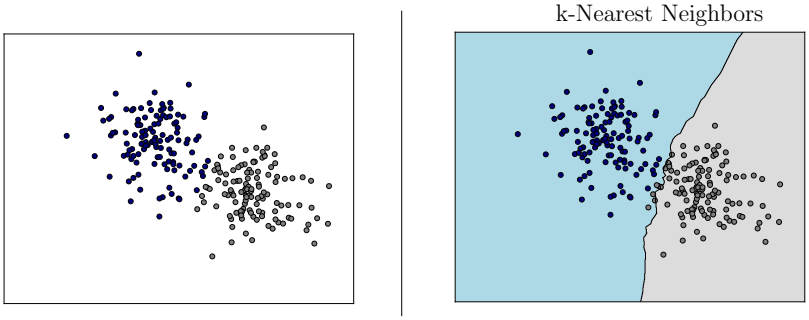


Figure 3.9: The decision boundary for the example two-dimensional data classified with a k-Nearest Neighbor classifier.

3.5.4.2 Training

The k-Nearest Neighbors algorithm is not trained, as it is instance-based. The algorithm will use the labeled training dataset to classify new data points instead of forming its own model of the data. The training stage therefore involves storing of the training dataset [7]. Some implementations simplify finding the k nearest neighbors by establishing a tree structure during training. This reduces the computational complexity involved with classifying an unseen data point [42].

3.5.4.3 Classification

The circle in figure 3.10 contains $k = 3$ data points from the training dataset that are the closest to the unknown new data point. The new data point will be classified according to the majority vote of the predefined k shortest distances. The unlabeled data point will therefore be assigned to the positive class. An uneven integer is chosen for k to ensure that there can never be an equal vote.

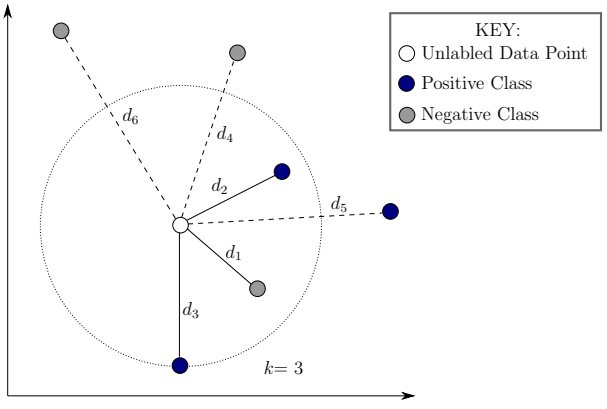


Figure 3.10: Visual explanation for classification using k nearest neighbors algorithm. The distance is calculated from the unlabeled data point to each training data point.

3.5.4.4 Advantages and Disadvantages

The advantages of the nearest neighbor algorithm are its conceptual simplicity and that there is no prior knowledge required about the dataset. One obvious disadvantage is memory requirement and computational intractability when dealing with large datasets.

3.5.4.5 Configurable Parameters

The following are configurable parameters for the k-Nearest Neighbors classifier:

- **n_neighbors** - the number of neighbors
- **algorithm** - the type of algorithm used to determine the nearest neighbors
- **metric** - distance metric used

The default number of neighbors used for classification is five. The choice of k will determine the smoothness of the decision boundary. A larger k is more immune to the effects of noise, but will result in a less distinct decision boundary. There are three different options for an algorithm that can be used to perform distance calculations, namely Brute Force, K-D Tree and Ball Tree. The Brute Force algorithm is effective for smaller datasets, but becomes infeasible for larger N. The scikit-learn library offers two tree-based data structures in the K-D and Ball tree to reduce the number of required iterations by summarizing distance information. The central idea is illustrated using three data points. If point A is distant from point B, and point B is in close proximity to point C, then points A and C must also be distant. The K-Dimensional (K-D) Tree is a binary tree structure that recursively partitions the data space along its axes. This strategy reduces the nearest neighbor computations to $\mathcal{O}(\log(N))$. The Ball Tree is an extension of the KD-Tree that partitions the data using a series of nested hyper-spheres instead of along Cartesian axes. This makes the tree construction more costly, but improves efficiency in higher-dimensional spaces.

3.5.5 Support Vector Machine

3.5.5.1 Overview

A Support Vector Machine (SVM) is a model-based classification method. It is also known as a Large Margin Classifier as it attempts to find the optimal hyperplane in multidimensional space that separates the binary classes. This corresponds to the maximum margin or distance between the support vectors and decision boundary [41]. The support vectors are those training data points nearest the decision boundary from either class. Support Vector Machines are powerful classifiers as they have the ability to separate linearly inseparable data. A training dataset is preprocessed to represent patterns in a different subspace where they become linearly separable using a non-linear mapping or kernel function as illustrated on the example dataset in figure 3.11.

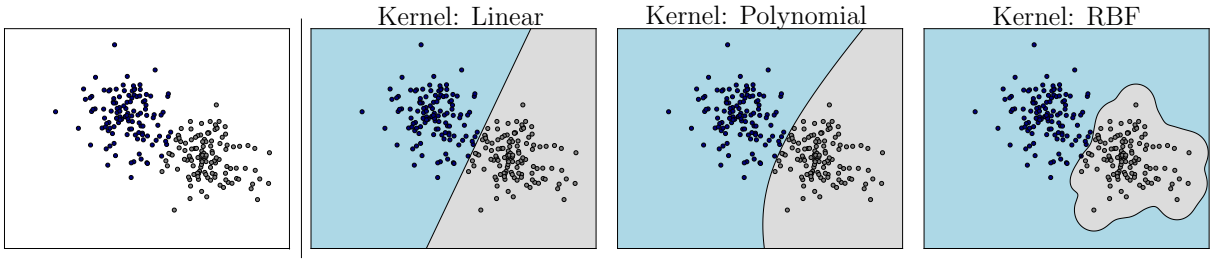


Figure 3.11: The SVM decision boundary for a two-dimensional dataset classified using three different Kernel functions, namely linear, polynomial and Radial Basis Function (RBF) kernels.

3.5.5.2 Training

A linearly separable dataset is considered for explanatory purposes. The equation that governs the operation of a linear classifier is given by:

$$y(x_i) = \vec{w}^T \vec{x}_i + b \tag{3.5.20}$$

where a hyperplane is defined by weight vector $\vec{w}$ and bias variable b.

New data points are classified according to the sign of $y(\vec{x}_i)$:

$$\vec{w}^T \vec{x}_i + b = \begin{cases} \geq 0 & \text{positive class} \\ < 0 & \text{negative class} \end{cases} \tag{3.5.21}$$

For new data points close to the decision boundary $\vec{w}^T \vec{x}_i + b = 0$ a small change in $\vec{x}_i$ can result in a misclassification. Support Vector Machines address this shortcoming through the use of a decision boundary that is separated from the training data by a finite threshold ϵ^2.

$$\vec{w}^T \vec{x}_i + b = \begin{cases} \geq \epsilon^2 & \text{positive class} \\ < -\epsilon^2 & \text{negative class} \end{cases} \tag{3.5.22}$$

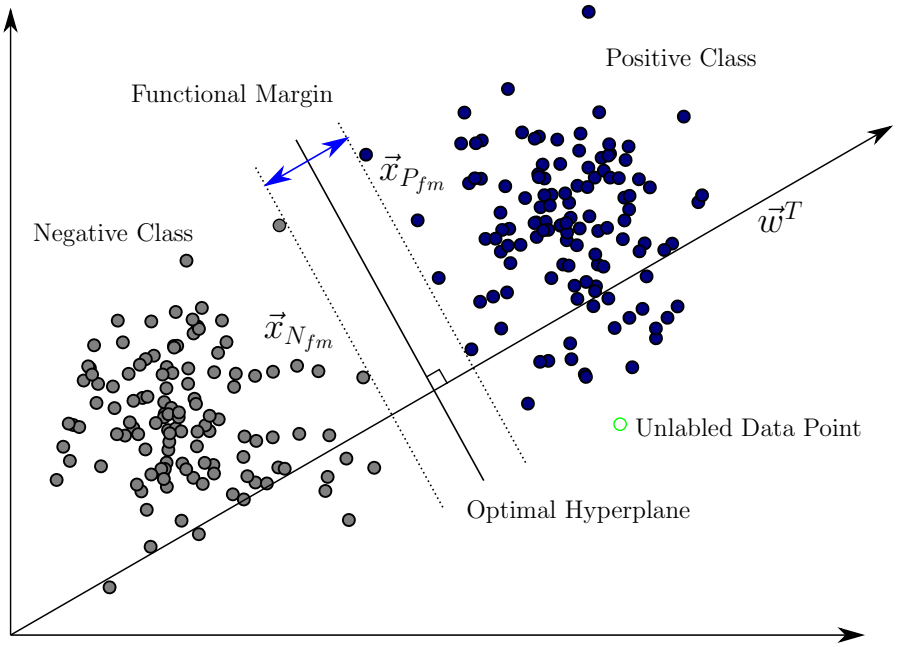


Figure 3.12: The optimal SVM decision boundary is represented by the solid line. The largest functional margin given by the dotted lines separates the two classes.

The SVM training aims to find the largest functional margin between the nearest data points of either class. The optimal location for the linear hyperplane that will maximise the width of the functional margin is to place it in the middle. The training process is reduced to an optimisation problem to find the largest functional margin. A Support

Vector Machine is trained by projecting the data points on a vector perpendicular to the hyperplane, $\vec{w}^T$ such that:

$$\vec{w}^T\vec{x}_P + b \geq 1 \tag{3.5.23}$$

$$\vec{w}^T\vec{x}_N + b \leq -1 \tag{3.5.24}$$

where $\epsilon = 1$ for convenience, $\vec{x}_P$ and $\vec{x}_N$ are training data points belonging to the positive and negative class respectively.

The distance between any data point, $\vec{x}_i$, from the origin along the vector $\vec{w}$ is $\frac{\vec{w}^T\cdot\vec{x}_i}{||\vec{w}||}$ [35]. The width of the functional margin is given by the distance along the direction of the vector $\vec{w}^T$ between data points from the positive class ($\vec{x}_{P_{fm}}$) and negative class ($\vec{x}_{N_{fm}}$) situated on the functional margin.

$$\frac{\vec{w}^T\cdot(\vec{x}_{P_{fm}} - \vec{x}_{N_{fm}})}{||\vec{w}||} = \frac{2}{||\vec{w}||} \tag{3.5.25}$$

The maximum width of the functional margin is a maximization of $\frac{2}{||\vec{w}||}$ or equivalently the minimization of $||\vec{w}||$. For each training data point $\vec{x}_i$ there is a corresponding label y_i that provides necessary information to maximise the functional margin, where the optimization problem is defined by [35]:

$$\arg\min_{\vec{w}} \frac{1}{2}||\vec{w}||^2 \text{ subject to } y_i(\vec{w}^T\vec{x}_i + b) - 1 \geq 0 \tag{3.5.26}$$

For training data points that lie on the boundary of the functional margin they satisfy the constraint that $y_i(\vec{w}^T\vec{x}_{i_{fm}} + b) - 1 = 0$. This becomes a constrained optimization problem using the constraint stated above. Lagrange multipliers can be used to determine the global minimum.

$$\mathcal{L}(\vec{w}, b) = \frac{1}{2} \mid\mid \vec{w} \mid\mid^2 - \sum_{i=1}^{N} \alpha_i[y_i(\vec{w}^T\vec{x}_{i_{fm}} + b) - 1] \tag{3.5.27}$$

where $\mathcal{L}()$ is the Lagrange function and α_i is the multiplier such that $\alpha_i \geq 0$.

The global minimum is determined by taking the partial derivatives of equation 3.5.27 and setting them equal to zero:

$$\nabla_{(\vec{w},b)}\mathcal{L}(\vec{w}, b) = 0 \Leftrightarrow \begin{cases} \vec{w} - \sum_{i=1}^{N} \alpha_i y_i \vec{x}_{i_{fm}} = 0 \\ -\sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{3.5.28}$$

The only definite solution presented above gives a global minimum at:

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_{i_{fm}} \tag{3.5.29}$$

Substituting equation 3.5.29 into equation 3.5.27 and then simplifying yields:

$$\mathcal{L}(\vec{w}, b) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \vec{x}_{i_{fm}} \cdot \vec{x}_{j_{fm}} \tag{3.5.30}$$

The problem is reduced to finding the maximum $\vec{x}_{i_{fm}} \cdot \vec{x}_{j_{fm}}$ [7].

For datasets that are not linearly separable it will not be possible to find the maximum $\vec{x}_{i_{fm}} \cdot \vec{x}_{j_{fm}}$. Kernel functions are used to project the d dimensional input space into a different subspace where the data becomes linearly separable using a kernel function $K(\vec{x}_i, \vec{x}_j)$. This allows for non-linear decision boundaries when the optimal hyperplane is projected back to the original input space as seen in figure 3.11. A higher-dimensional feature space does not actually have to be constructed as the kernel trick allows all necessary operations to be performed in the input space [41].

3.5.5.3 Classification

New data points are classified according to their location with respect to the optimal hyperplane. If the projection of new data point onto the vector $\vec{w}$ is greater than zero it will assigned to the positive class as seen in figure 3.12. The sign of $y(\vec{x}_i)$ therefore determines which class a new training example is assigned to [36].

$$y(\vec{x}_i) = \vec{w}^T \vec{x}_i + b = \begin{cases} \geq 0 & \text{positive class} \\ < 0 & \text{negative class} \end{cases} \tag{3.5.31}$$

3.5.5.4 Advantages and Disadvantages

The advantages of a Support Vector Machine classifier is that the use of kernels give it the ability to separate linearly inseparable data, and that the optimisation function is of a convex form that ensures that the algorithm will converge on the global minimum. The disadvantages of a Support Vector Machine classifier is the use of a kernel function increases the computational complexity and may also require fine tuning which would be time consuming.

3.5.5.5 Configurable Parameters

The following are configurable parameters for the Support Vector Machine classifier:

- **C** - penalty parameter
- **kernel** - the type of kernel to be used

The penalty parameter controls the degree of regularisation that prevents over fitting. A well-trained algorithm should generalise well to new training data points. C is the inverse regularization parameter and its purpose is to prevent over-fitting.

Kernel Type	Function
Linear	$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i^T \vec{x}_j$
Polynomial	$K(\vec{x}_i, \vec{x}_j) = \gamma(\vec{x}_i^T \vec{x}_j + r)^d$
Radial Basis Function	$K(\vec{x}_i, \vec{x}_j) = e^{-\gamma \|\vec{x}_i - \vec{x}_j\|^2}$
Sigmoid	$K(\vec{x}_i, \vec{x}_j) = \tanh[\gamma(\vec{x}_i^T \vec{x}_j + r)]$

Table 3.3: Summary of kernel functions offered by the SVC library.

The choice of kernel depends on the intended application. The linear kernel is the simplest kernel function whereby it returns the inner product $\vec{x}_i \cdot \vec{x}_j$. The polynomial kernel is non-stationary and well suited for training data that has been normalised. The Radial Basis Function (RBF) kernel measures similarity between $\vec{x}_i$ and $\vec{x}_j$. Euclidean distance is used as the measure of similarity. A value that tends towards 1 is an indication of $\vec{x}_i$ and $\vec{x}_j$ that are similar, whilst dissimilarity a value of 0. The sigmoid kernel stems from neural networks and is usually used in text classification applications [34]. The different configurable parameters for various kernels are γ, d and r.

3.6 Multi-class Classification

Multi-class refers to more than two classes. These classification algorithms are investigated as a means to identify a faulty sensor once a fault has been detected. Whilst some of the binary classifiers discussed in section 3.5 are inherently multi-class, others need assistance to expand to $k > 2$ classes. There are two distinct methodologies by which binary classification algorithms are expanded for multi-class applications, namely the One-Versus-All and One-Versus-One transformation. The binary classifiers from section 3.5 are all capable of multi-class classification when using the scikit-learn machine learning library [43]. This section will present and discuss the various strategies used to expand the binary classifiers to a multi-class setting.

Classifier Name	**Transformation Strategy**
Naïve Bayes	Inherently Multi-class
Decision Tree	Inherently Multi-class
k-Nearest Neighbors	Inherently Multi-class
Logistic Regression	One-Vs-All
Support Vector Machine	One-Vs-One

Table 3.4: Summary of multi-class classifiers.

3.6.1 Inherently Multi-class

3.6.1.1 Overview

The Naïve Bayes, Decision Tree and k-Nearest Neighbors classifiers are all inherently multi-class. The section below will give a brief overview of their multi-class operation and show how the equations that govern them apply to a multi-class setting.

Naïve Bayes
The multi-class Naïve Bayes algorithm is simply an extension of its binary counterpart by extending the projection of conditional probability distributions on respective axes to each of the k classes. New data points are assigned to the class that maximises the conditional probability as given by equation 3.5.16.

Decision Tree
A Decision Tree is trained to determine the feature and value that maximises the quality

of the split across all the classes. The Gini impurity equation 3.5.17 can be expanded to account for k classes denoted by subset $C = \{C_1, C_2, ..., C_k\}$ as follows:

$$H_G(C) = 1 - \sum_{i=1}^{k} p(C_i) \tag{3.6.1}$$

where $p(C_i)$ is the class probability.

Similarly, the entropy impurity equation 3.5.18 can be rewritten as:

$$H_E(C) = -\sum_{i=1}^{k} p(C_i) \log(p(C_i)) \tag{3.6.2}$$

Using the impurity equation the information gain 3.5.19 that is used to assess the quality of the split between parent S and children (S_L, S_R) in a multi-class problem is given by:

$$\Delta IG(S) = H_{[]}(C) - p(S_L)H_{[]}(C_L) - p(S_R)H_{[]}(C_R) \tag{3.6.3}$$

Once the tree is fully grown each leaf node represents one of the k classes.

k-Nearest Neighbors
The k-Nearest Neighbors is an intuitive algorithm whose functioning remains the same regardless of the number of classes. An unlabeled data point is still assigned to the class with the majority vote.

3.6.1.2 Advantages and Disadvantages

The advantages of applying inherent multi-class algorithms are that they only require the establishment of a single classifier with comparable performance to their binary counterparts.

3.6.2 One-Versus-All

3.6.2.1 Overview

The One-Versus-All approach establishes k binary classifiers. Each individual classifier is trained to identify a distinct class. This is achieved by assigning the training data belonging to the distinct class a positive label and grouping the remaining k-1 classes together under a separate negative label. The target labels assigned to the i^{th} classifier are given by:

$$y_i = \begin{cases} +1 & \text{if } y = i \\ -1 & \text{if } y \neq i \end{cases} \tag{3.6.4}$$

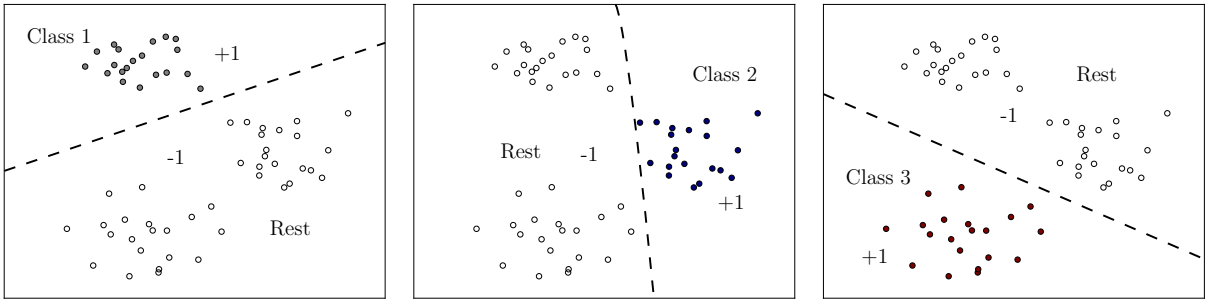


Figure 3.13: The One-Versus-All approach to multi-class classification for the example dataset where k=3 establishes separate Logistic Regression binary classifiers with decision boundaries indicated by the dashed line. The binary classifier is trained to identify one class from the remainder of the dataset.

The separate binary classifiers are combined by assigning the new data point a label that corresponds to the class that maximises the probability function:

$$P(y = i|\vec{x}, \vec{w}_i, b_i) = \frac{1}{1 + e^{-(\vec{w}_i^T \vec{x} + b_i)}} \tag{3.6.5}$$

where $\vec{w}_i$ and b_i are model parameters for the i^{th} classifier.

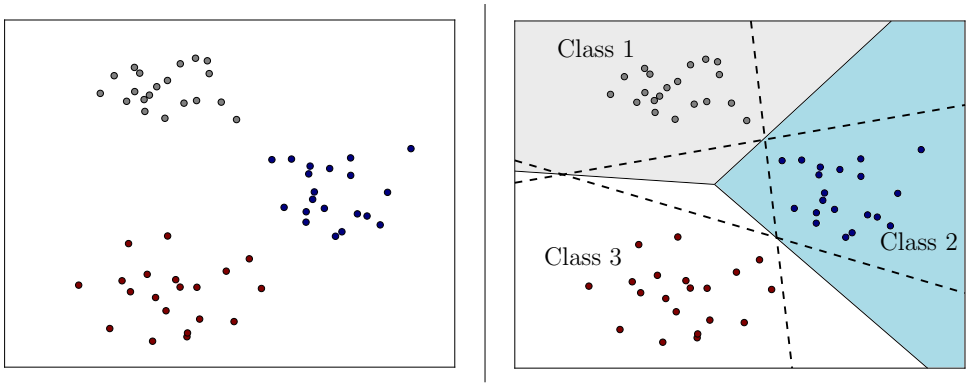


Figure 3.14: The separate Logistic Regression binary classifiers indicated by the dashed lines are combined using equation 3.6.5 to establish the multi-class classifier decision boundary as indicated by the solid lines and shaded regions.

3.6.2.2 Advantages and Disadvantages

The one-versus-all approach is generic in the sense that it can be applied to any binary classification algorithm. The drawbacks are that the computational and memory requirements are more intensive as it requires k binary classifiers. The second distinct disadvantage is that the individual binary classifiers are biased as they are trained on unbalanced datasets consisting of more negative class training data points than positive class training data points.

3.6.3 One-Versus-One

3.6.3.1 Overview

The One-Versus-One approach establishes $\frac{1}{2}k(k\text{-}1)$ binary classifiers where each individual classifier is trained on a different pair of classes from the original training set. These individual classifiers are combined to form a single classifier. This is an ensemble approach to multi-class classification where the majority vote is used to classify new data points.

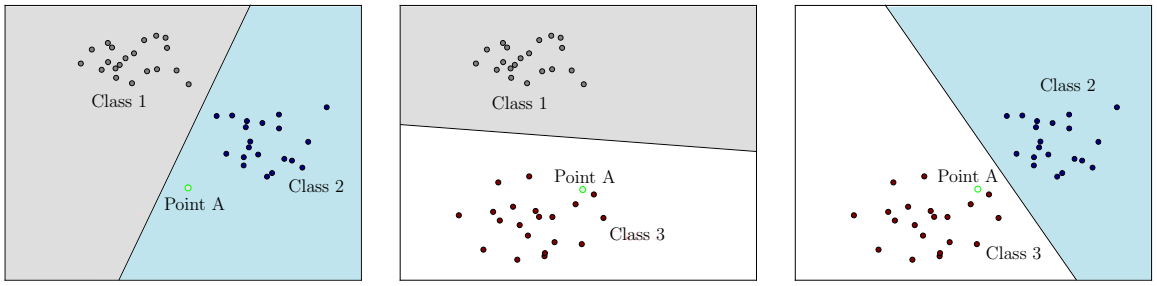


Figure 3.15: The One-Versus-One approach to multi-class classification for the example dataset where k=3 establishes separate SVM binary classifiers using a linear kernel. Point A is a new data point that will be assigned to Class 3 using a majority vote.

3.6.3.2 Advantages and Disadvantages

The number of binary classifiers required for the One-Versus-One approach increases as a squared function of the number of classes k. This will make the One-Versus-One approach computationally intensive. The advantage over the One-Versus-All approach is that the binary classifiers are trained on an unbiased training set.

3.7 Outlier Detection

Faults are rare occurrences and can be viewed as outliers as they will produce measurements which greatly differ from the majority of the data. A variety of outlier detection algorithms are investigated in this section that are based on different fundamental assumptions and approaches. Data driven outlier detection algorithms are intended for offline application on large, static datasets with few outliers. An unlabeled dataset and the percentage of outliers is given as input and it is the responsibility of the detection algorithm to determine which data points are the most likely outliers. The outlier detection algorithms are similar to novelty detectors in that they are trained to identify faults based on their similarity to the majority class (no-fault data). Table 3.5 provides a summary of the outlier detection approaches that are investigated.

Outlier Approach Name	Approach Type
Elliptic Envelope	Robust distribution estimation
Local Outlier Factor	Density-based outlier detection
Isolation Forest	Rules-based detection

Table 3.5: Summary of outlier detection methods.

3.7.1 Elliptic Envelope

3.7.1.1 Overview

The Elliptic Envelope assumes that the "inlier" or "no-fault" data points come from a multivariate Gaussian distribution. Robust estimation techniques such as the minimum covariance determinant (MCD) are used to estimate the mean and variance of a multivariate dataset containing outliers. The elliptic envelope algorithm firstly estimates the distribution of normal data using the MCD estimator, thereafter an elliptical classification boundary is fitted. The Elliptic Envelope is fitted to an example dataset as shown in figure 3.16.

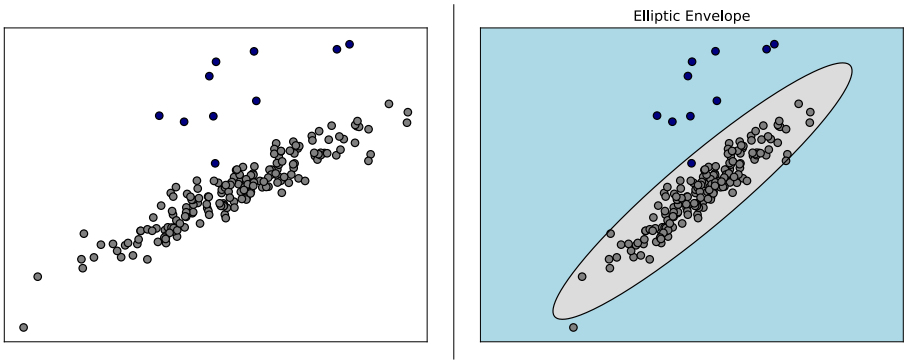


Figure 3.16: The decision boundary for two-dimensional example dataset classified using a Elliptic Envelope outlier detector. The navy data points are the outliers, while the grey data points are inliers.

3.7.1.2 Training

The training of the Elliptic Envelope iteratively searches for the best representative subset of inlier data points using the MCD algorithm as shown in figure 3.17 and explained in greater detail below.

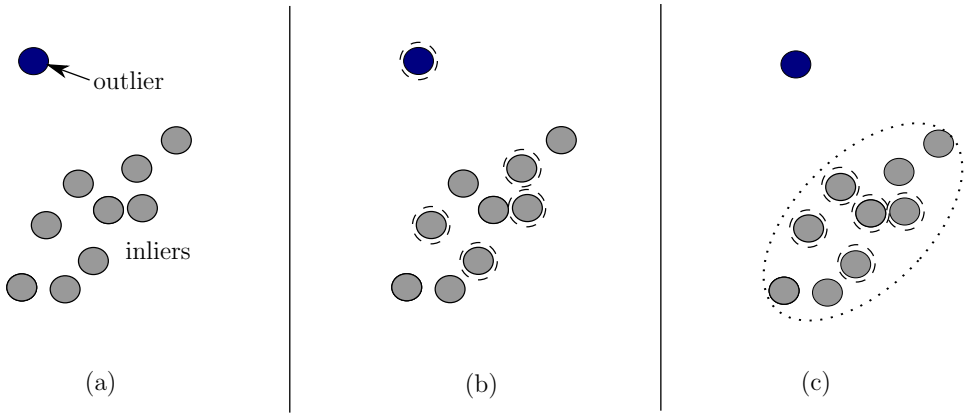


Figure 3.17: Visual demonstration of the Minimum Covariance Determinant algorithm. Figure (a) shows the original dataset with a single outlier. A subset of $h = 5$ random data points are selected as indicated by the dashed circles in figure (b). Their mean and variance is calculated and the h data points from the larger dataset that minimise the Mahalanobis distance are selected. This process is repeated until the subset of data points remains unchanged. An elliptic envelope classification boundary is fitted and used to correctly identify the outlier as illustrated in figure (c).

The MCD estimator attempts to find h inlier data points from a larger set of N data points by recursively selecting combinations that minimise the determinant of the covariance matrix. Analyzing all possible combinations is computationally expensive as it requires that the covariance determinant of $\binom{N}{h}$ subsets be calculated. The fast MCD algorithm as developed by Rousseeuw et al. [44] addresses this shortfall. The algorithm is initiated by selecting a random subset of h data samples, $H_1 \subset \{x_1, ...x_N\}$. The sample mean ($\vec{\mu}_1$) and covariance ($\Sigma_1$) of subset H_1 are calculated under the assumption that $\det(\Sigma_1) \neq 0$. The Mahalanobis distance, defined as the number of standard deviations between a data point and a distribution, is calculated for each of the N data points as follows:

$$d(i) = \sqrt{(\vec{x}_i - \vec{\mu}_1)^T \Sigma_1^{-1} (\vec{x}_i - \vec{\mu}_1)} \tag{3.7.1}$$

The Mahalanobis distances are arranged in an increasing order of magnitude. The process is repeated for a new subset, H_2, that contains the h data points with the smallest Mahalanobis distances. The property $\det(\Sigma_2) \leq \det(\Sigma_1)$ will achieve equality when $\vec{\mu}_1{=}\vec{\mu}_2$, $\Sigma_1{=}\Sigma_2$ and the process is terminated. This repetitive process is known as the C or concentration step as the algorithm attempts to locate the combination of h samples that are most densely concentrated.

1. Compute $\vec{\mu}_j$ and Σ_j for subset H_j
2. Terminate process if $\Sigma_j = \Sigma_{j-1}$
3. Compute Mahalanobis distances $d(i)$ for the entire dataset
4. Rearrange the Mahalanobis distances in increasing order of magnitude
5. Create new subset H_{j+1} consisting of h data points with the smallest Mahalanobis distances

The initial subset of h training examples may lead to a sub-optimal solution, therefore an approximate MCD solution is found by taking many initial subsets H_1 and applying C-step to each one. The solution that yields the lowest covariance determinant is regarded as being correct.

The parameter h is governed by the following restriction:

$$\frac{(N+d+1)}{2} \leq h \leq N \tag{3.7.2}$$

where d is the dimensionality of the dataset.

3.7.1.3 Classification

New data points are classified based on their Mahalanobis distance from the inlier distribution as given by equation 3.7.1. A threshold (Δ) is used to classify new data points $\vec{x}_i$ as follows:

$$y(\vec{x}_i) = \begin{cases} d(i) \geq \Delta & \text{outlier} \\ d(i) < \Delta & \text{inlier} \end{cases} \tag{3.7.3}$$

The threshold Δ is based on a Chi-Square distribution with the same number of degrees of freedom as the dimensionality [44].

$$\Delta = \sqrt{\chi^2_{d,p}} \tag{3.7.4}$$

where p is the percentage of outlier data points.

3.7.1.4 Advantages and Disadvantages

The advantages of the Elliptic Envelope is its simplicity of use and that there is no need for prior knowledge other than the percentage of outliers in the training dataset. It can also be used as a novelty detection algorithm, by setting the percentage of outliers to zero. The disadvantage with the approach is that it is based on the assumption of an inlier multivariate Gaussian distribution that might not be valid for all datasets.

3.7.1.5 Configurable Parameters

The following are configurable parameters for the Elliptic Envelope outlier detector:

- **contamination** - percentage outliers in the dataset
- **support_fraction** - percentage data points to be included in MCD calculation

3.7.2 Local Outlier Factor

3.7.2.1 Overview

Local Outlier Factor (LOF) is a density-based algorithm developed by Kriegel at al [45] that expands on standard outlier detection algorithms by indicating the degree of outlierness. Most outlier detection algorithms regard being an outlier as a binary property. The distance between a data point and its k nearest neighbors is used to estimate its density.

The deviation in density between the observed data point and its k nearest neighbors is reflected in a measure called the LOF score. A unity value indicates that the data point is comparable with its neighbors and is most likely an inlier, while larger LOF scores are associated with outliers. The Local Outlier Factor is fitted to an example dataset as shown in figure 3.18.

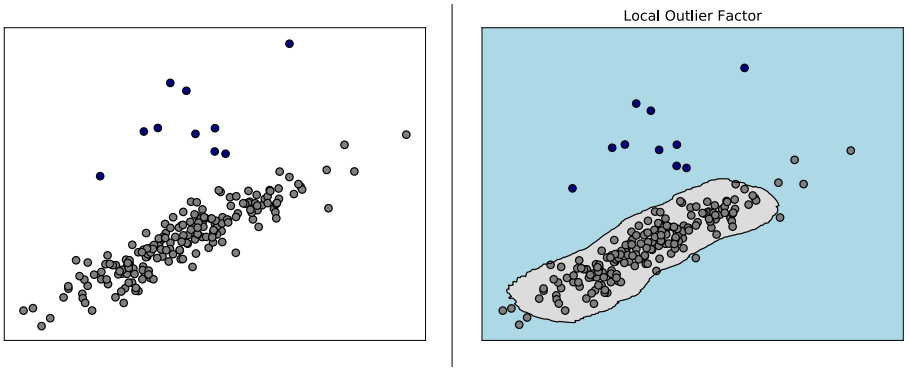


Figure 3.18: The decision boundary for two-dimensional example dataset classified using a LOF outlier detector. The navy data points are the outliers, while the grey data points are inliers.

3.7.2.2 Training

The training entails determining the Local Outlier Factor for each data point in the training dataset. This requires that the concepts of k-distance (k_{dist}), reachability distance (r_{dist}) and the local reachability density (lrd) be defined as they are all necessary to calculate the LOF score. For explanatory purposes it is assumed that the LOF algorithm uses three of the nearest neighbors ($k = 3$) to determine the LOF for each data point. The k_{dist} is given as the Euclidean distance between the data point under consideration, $\vec{x}_i$, and its k^{th} nearest neighbor. Breunig et al. [46] proceed to define the concept of reachability distance, r_{dist}, between data point $\vec{x}_i$ and $\vec{x}_o$. $\vec{x}_o$ is representative of any one of the neighbors of data point $\vec{x}_i$. The reachability distance as explained in figure 3.19 is defined as follows:

$$r_{dist}(i, o) = \max\left\{k_{dist}(o), d(\vec{x}_i, \vec{x}_o)\}\right\} \tag{3.7.5}$$

The reachability distance acts as a smoothing function, where the choice of k controls the degree of smoothing. In typical density-based clustering algorithms, there will be a minimum number of data points that need to fall within a specific volume surrounding the test point for it to be considered an inlier [46]. The LOF algorithm replaces the specific volume with a local reachability density that is given by the inverse of the average reachability distance. The local reachability density (lrd) of data point $\vec{x}_i$ is given by:

$$lrd(i) = \left(\frac{1}{k}\sum_{o\in\{a,b,c\}} r_{dist}(i, o)\right)^{-1} \tag{3.7.6}$$

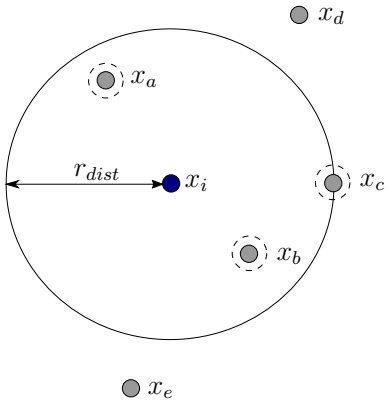


Figure 3.19: Visual demonstration of the concept of reachability distance. The $k = 3$ nearest neighbors of data point $\vec{x}_i$ are shown with dashed circles. The reachability distance of data point $\vec{x}_a$, $\vec{x}_b$ and $\vec{x}_c$ are the same. This is given by the distance between data point x_i and the k^{th} nearest neighbor. The reachability distance to $\vec{x}_d$ and $\vec{x}_e$ are given by the Euclidean distance, which is greater than the k_{dist}.

The LOF determines the average ratio of local reachability density of data point $\vec{x}_i$ in comparison with its k nearest neighbors. The LOF of data point $\vec{x}_i$ is given by:

$$LOF(i) = \frac{1}{k} \sum_{o \in \{a,b,c\}} \frac{lrd(o)}{lrd(i)} \tag{3.7.7}$$

When data point $\vec{x}_i$ is an outlier, it will have a lower local reachability density and a larger LOF score.

3.7.2.3 Classification

New data points are classified based on their LOF score. The user sets the percentage of training data to be regarded as outliers through a predefined input parameter that will adjust the cut-off LOF threshold accordingly.

3.7.2.4 Advantages and Disadvantages

The advantage of Local Outlier Factor is that it considers both the local and global properties of datasets and can therefore adapt well to abnormal distributions [46]. The drawback is that it is a complex method that is computationally intensive.

3.7.2.5 Configurable Parameters

The following are configurable parameters for the Local Outlier Factor:

- **contamination** - percentage outliers in the dataset
- **n_neighbors** - number of nearest neighbors to be considered

3.7.3 Isolation Forest

3.7.3.1 Overview

An isolation forest combines numerous isolation trees to form a single classier or model [7]. This approach is based on the assumption that outliers are easily distinguishable from

normal data points and therefore when establishing tree structures all outliers should terminate at nodes situated close to the root of a tree. Normal data points are harder to distinguish and therefore are situated much deeper within. The Isolation Forest is fitted to an example dataset as shown in figure 3.20.

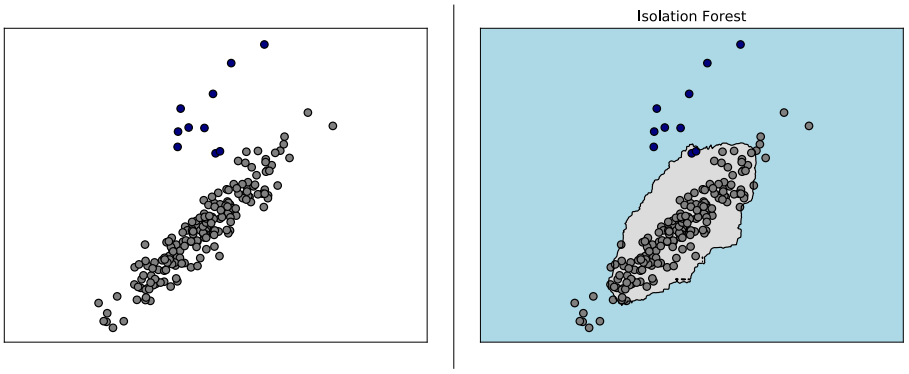


Figure 3.20: The decision boundary for two-dimensional example dataset classified using a Isolation Forest outlier detector. The navy data points are the outliers, while the grey data points are inliers.

3.7.3.2 Training

The individual isolation trees are established using a modified Classification and Regression Tree (CART) methodology for unlabled datasets. Given that CART training methodology was extensively covered in Chapter 3.5.3.2, this section will rather elaborate on the modifications to the algorithm.

Firstly, the isolation forest is an ensemble approach that establishes and combines numerous partial models of the data. This is achieved by training individual trees on smaller subsets of data as opposed to establishing a complete binary tree of all data points. It is this property that makes the Isolation Forest effective at dealing with high-dimensional and large quantities of data. The second modification is around the manner in which an isolation tree is trained. Binary trees are recursively partitioned until each data point terminates on a distinct leaf node. For labeled datasets the information gain is used to assess and ensure that the best quality split is chosen. Isolation Forests are trained on an unlabeled dataset and therefore the partitioning methodology must be adapted. A random partitioning methodology is applied that randomly selects a feature and split value until the tree is fully grown or it reaches some predefined height limit [47]. Individual trees therefore have unique partitions and data on which they are trained.

3.7.3.3 Classification

New data points are classified based on the path length between root and termination node averaged over a forest of random trees. This is the average number of edges that must be transversed when following the binary tree of if-else statements. The user sets the percentage of training data to be regarded as outliers through a predefined input parameter that will adjust the cut-off threshold for average path length accordingly.

3.7.3.4 Advantages and Disadvantages

Most conventional outlier detection algorithms are constrained to a low-dimensional feature space and small datasets due to the computational complexity [48]. This is not the case for an Isolation Forest that scales to larger datasets and higher dimensions with relative ease. The drawback of an Isolation Forest is that the results may vary due to the random partitioning strategy.

3.7.3.5 Configurable Parameters

The following are configurable parameters for the Isolation Forest:

- **contamination** - percentage outliers in the dataset
- **n_estimators** - number of base estimators that are to be combined
- **max_samples** - number of data points in smaller subsets

3.8 Other Data Driven Approaches

Table 3.6 provides a summary of additional data-driven techniques found in literature that have relevance to fault detection and isolation. RANdom SAmple Consensus (RANSAC) is a robust estimator of model parameters from a dataset polluted by outliers. Dynamic Principal Component Analysis (D-PCA) is used for fault detection by monitoring changes in statistical measures of a dynamic data stream.

Approach Name	Approach Type
RANSAC	Robust model estimation
Dynamic PCA	Monitoring of statistical measures

Table 3.6: Summary of other data-driven methods found in literature.

These techniques require no prior training, but rather exhibit potential to derive structure from a window of consecutive measurements. An additional approach in Dynamic Clustering for evolving environments (DyClee) was also considered and is documented in Appendix B.2 as additional information for the reader.

3.8.1 RANSAC

RANdom SAmple Consensus (RANSAC) is a widely used outlier removal technique often applied in the field of visual odometry. It was first introduced by Fischler and Bolles as a means to estimate model parameters from a dataset contaminated by outliers.

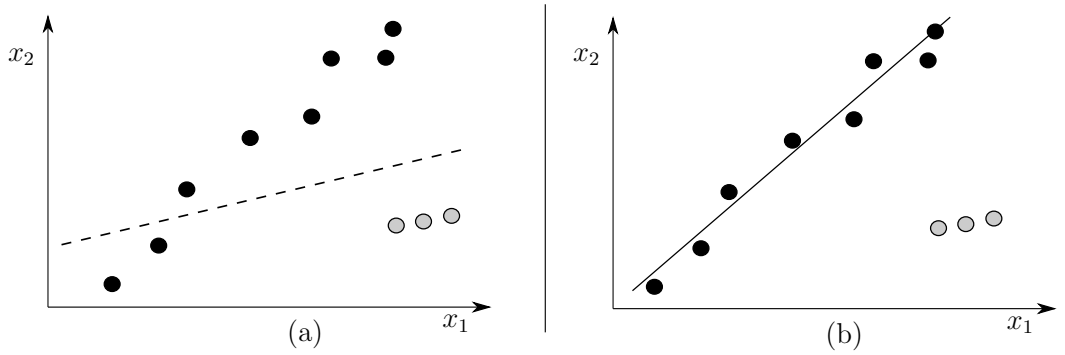


Figure 3.21: The black data points represent inliers whilst the grey are outliers. The least squares estimation of the 2-D dataset returns the model represented by the dashed line in (a), whilst the true inlier model is shown by the solid line in (b).

The RANSAC algorithm consists of an iterative hypothesise-and-verify approach with the aim of determining a representative model for the inlier data points. During the first stage a model is instantiated from a minimalistic subset of random data points. For the example of a straight line only two data points are required to hypothesise a possible model. The second stage evaluates the effective representativeness of the proposed model by determining the degree of support for the hypothesised model. Some predefined error threshold such as Euclidean distance is used to evaluate the consistency of the model. This two stage process is repeated k times and the model with the maximum support is selected.

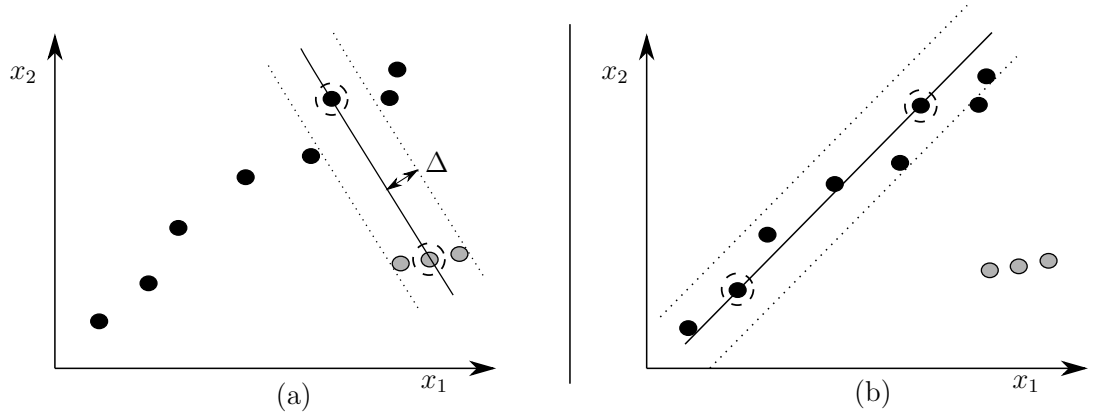


Figure 3.22: Two examples of possible hypotheses generated by the RANSAC algorithm. The dashed circle shows the random data points initially selected. The solid line shows the model fitted to these points, whilst the dashed line depicts the boundary to determine the degree of support for the model. Example (a) shows a model that is poorly supported, whilst (b) shows a model with good support.

The standard RANSAC framework does not guarantee finding the correct or optimal model due to the nature of random sampling. The probability of finding a representative model is equivalent to randomly selecting a minimalistic subset of m inlier data points from the dataset $\mathcal{Z}$ [38]. Assuming the data points are sampled independently, the optimal number of iterations required to satisfy η can be determined using the derivation of

Fischer and Bolles [49]:

$$k = \frac{log(1-\eta)}{log(1-\epsilon^m)} \tag{3.8.1}$$

where η the probability of success and ϵ is the percentage of inliers.

Algorithm 1 Pseudocode for RANSAC algorithm.

Input:

$\mathcal{Z}$	dataset of measurements
Δ	threshold for determining degree of support
η	probability of success
ϵ	percentage inliers
m	minimum number of points to instantiate model

Output:

H(θ)	model representing inliers
$\mathcal{I}$	inlier dataset

```
function RANSAC(Z,Δ,η,ϵ,m)
    k ← Required Iterations(η,ϵ,m)
    for j = 1:k do
        S ← Random Sample(Z,m)
        H(θ_j)←Fit Model(S)
        I_j ← Model Evaluation(H(θ_j),Δ, Z)
        if |I_j| > I then
            H(θ) = H(θ_j)
            I = I_j
        end if
    end for
return(I,H(θ))
```

There are many variations to the standard RANSAC framework that have been documented in Appendix B for the interested reader. These variations were investigated, but were not used.

3.8.2 Dynamic PCA

Principle Component Analysis (PCA) is a data analytics technique that is used to extract the principal components that give structure to a stationary dataset. PCA is a linear transform of the correlated data space such that the new representation of the features becomes uncorrelated. The transformed dataset is given by:

$$Y = V^T X \text{ or } V^T(X - \vec{\mu}) \tag{3.8.2}$$

where V is the transformation matrix, X the original dataset and $\vec{\mu}$ is the mean vector.

The transformed data space is either centered around the mean of the distribution (centered rotation) as illustrated in figure 3.23 or around the origin (uncentered rotation).

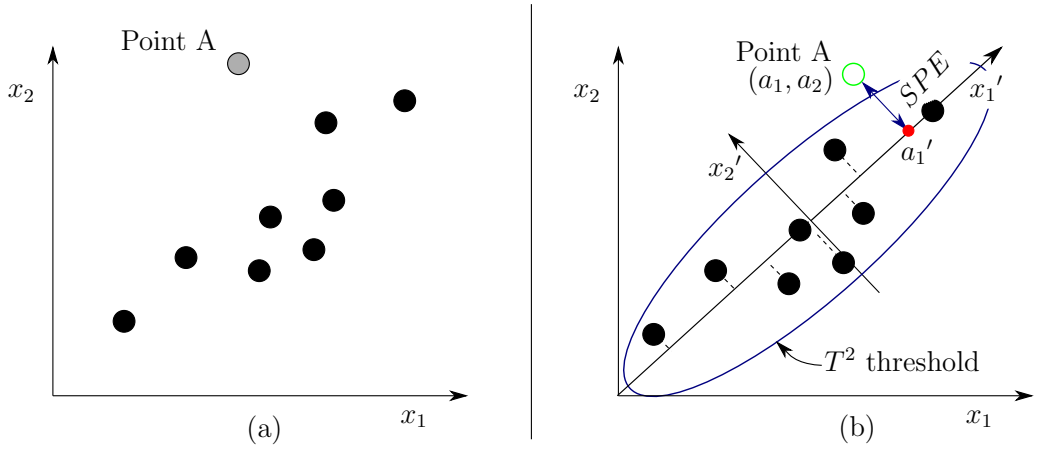


Figure 3.23: The correlated input data space is shown in figure (a). PCA is applied to the dataset to demonstrate how the best representative linear model is chosen to coincide with the first principal component. The original dataset is projected on to the first principal component as indicated by the dashed lines. Outliers such as point A can be identified using statistical measures such as the Square Projection Error (SPE) or Hotelling's T^2 statistic.

The PCA algorithm is explained using the projection of the data points onto the first principal component that is given by $\vec{v_1} = \frac{x_1'}{||x_1'||}$. The mean vector ($\vec{\mu}$) and covariance matrix (Σ) for the dataset consisting of N data points is given by:

$$\vec{\mu} = \frac{1}{N}\sum_{i=1}^{N}\vec{x_i} \tag{3.8.3}$$

$$\Sigma = \frac{1}{N-1}\sum_{i=1}^{N}(\vec{x_i} - \vec{\mu})(\vec{x_i} - \vec{\mu})^T \tag{3.8.4}$$

For the convenience of the derivation it is assumed that $\vec{v_1}$ is a unit vector. The mean of the projected data is given by $\vec{v_1}^T\vec{\mu}$. The variance of the projected data is given by:

$$\frac{1}{N}\sum_{i=1}^{N}(\vec{v_1}^T\vec{x_i} - \vec{v_1}^T\vec{\mu})^2 = \vec{v_1}^T\Sigma\vec{v_1} \tag{3.8.5}$$

In order to retain the maximum variance of the dataset, the projected variance $\vec{v_1}^T\Sigma\vec{v_1}$ is maximised with respect to $\vec{v_1}$. A Lagrange multiplier with constraint $\vec{v_1}^T\vec{v_1} = 1$ is used to maximise the following equation [36]:

$$\mathcal{L}(\vec{v_1}) = \vec{v_1}^T\Sigma\vec{v_1} + \alpha_1(1 - \vec{v_1}^T\vec{v_1}) \tag{3.8.6}$$

where $\mathcal{L}()$ is the Lagrange function and α_1 is the Lagrange multiplier.

The global minimum is determined by taking the partial derivative of equation 3.8.6 and setting it equal to zero:

$$\nabla_{(\vec{v_1})}\mathcal{L}(\vec{v_1}) = 0 \tag{3.8.7}$$

$$\Rightarrow \Sigma\vec{v_1} = \alpha_1\vec{v_1} \tag{3.8.8}$$

The variance of the projected data will be a maximum when $\vec{v_1}$ is set to be the eigenvector that has the largest eigenvalue α_1. This eigenvector is known as the first principal component [36]. Additional principal components are given by the remaining eigenvectors. These are orthogonal to the principal component axis and are considered in an incremental order that maximises the projected variance of the data. The orthogonal transform is therefore reduced to finding the eigenvectors and eigenvalues for the given dataset. Algorithms such as eigenvalue decomposition that determine the eigenvalues and eigenvectors will not be discussed here, but are available in Golub and Van Loan [50]. An eigenvalue decomposition is given by:

$$\Sigma = V \Lambda V^T \tag{3.8.9}$$

where V is a matrix that consists of individual eigenvectors arranged in columns such that $V = [\vec{v_1}^T, \vec{v_2}^T, .., \vec{v_d}^T]$ and Λ is a diagonal matrix of individual eigenvalues $\Lambda = \text{diag}(\alpha_i)$, where $i = 1$ to d (dimensionality).

The transformation matrix (V) is applied to input dataset using equation 3.8.2 to create a transformed feature space where data points are uncorrelated. The original dataset X is transformed to an equivalent dataset Y coordinated in the principal axes, using equation 3.8.2. PCA can also be used to represent the original dataset with fewer dimensions by projecting the data points into a reduced subspace described by the largest principal components. The eigenvectors corresponding to the k largest non-zero eigenvalues are used to form the transformation matrix [51]:

$$Y_k = V_k^T X \tag{3.8.10}$$

The reconstruction of the transformed data points in the original feature space given by:

$$X' = V_k Y_k \tag{3.8.11}$$

When applying PCA to a dynamic data stream using a large, sliding window the dynamic information between the features will be lost [51]. Dynamic PCA is therefore an extension of this approach that adapts to the series correlation of a dynamic data stream by incorporating a time history of data points. Let X be a set of n data points from a d dimensional dynamic data stream:

$$X = [X_1, X_2, ..., X_d]_{(\text{n} \times \text{p})} \tag{3.8.12}$$

Standard PCA assumes that each of the d features are stationary over a window of n sequential measurements. This assumption does not hold for a dynamic data stream. Therefore a trajectory matrix is established for the i^{th} feature by applying a time lag shift of order w as follows:

$$X_i^w = \begin{bmatrix} X_i(1) & X_i(2) & \cdots & X_i(w) \\ X_i(2) & X_i(3) & \cdots & X_i(w+1) \\ : & & \ddots & : \\ X_i(n-w+1) & \cdots & \cdots & X_i(n) \end{bmatrix} \tag{3.8.13}$$

The trajectory matrix is formed by combining the trajectory matrices for individual features:

$$X^w = \begin{bmatrix} X_1^w & X_2^w & \cdots & X_d^w \end{bmatrix} \tag{3.8.14}$$

Equation 3.8.10 is used to establish the transformed feature space using an appropriate selection of eigenvectors derived from the correlation of the trajectory matrix. This application of PCA on the trajectory matrix constitutes the key difference between standard and Dynamic PCA.

Fault detection and isolation is performed through the monitoring of statistical measures associated with individual time lag shift vectors. When a significant fault occurs, the time lag shift vector that incorporates that measurement will drastically differ. Statistical measures are used to perform fault detection as illustrated in figure 3.23. Two common measures are Hotelling's T^2 statistic and square projection error (SPE), where T^2 measures the major variation of the data and SPE is a measure of goodness of fit. The T^2 statistic is the square of the Mahalanobis distance defined as a measure of the number of standard deviations between a data point and the mean of the distribution. The T^2 is calculated for individual data points and is given by:

$$T^2 = \vec{y}^T \Sigma^{-1} \vec{y} \tag{3.8.15}$$

where $\vec{y}$ is a transformed data point.

The SPE is a scalar value that establishes the representativeness of a data point to the model defined by the lower dimensional eigenvectors. The SPE is the square of the residual error between a data point and its projection into the lower dimensional subspace. Figure 3.23 depicts the lower dimensional projection of Point A using a red circle. The square projection error is given by:

$$SPE = (\vec{x} - \vec{x}\,')^T (\vec{x} - \vec{x}\,') \tag{3.8.16}$$

where $\vec{x}$ is the original data point and $\vec{x}\,'$ is the reconstructed data point.

3.9 Conclusion

This chapter has presented the necessary background theory for a variety of fault detection and isolation techniques. Analytical redundancy approaches to FDI are investigated as an alternative to the currently used hardware redundancy in relative altimetry systems as this method is not robust to the failure of an entire technology. The model-based techniques establish explicit mathematical models of the system and detect faults using residual analysis. The model-based techniques investigated are variants of the Kalman filter. The data-driven techniques do not require any prior knowledge of system dynamics, but rather rely on historic data to derive structure or "learn" ways of identifying faults. The data-driven techniques that were presented include outlier and binary classification algorithms for fault detection and multi-class classifiers for fault identification. Other novel data-driven approaches that have potential for performing fault detection and isolation are also investigated. Now that a variety of fault detection and isolation techniques have been investigated, the next chapter will apply these techniques to the problem of aircraft height verification.

Chapter 4

Fault Diagnostic System

This chapter presents and discusses all the necessary information required to design and evaluate the fault detection and isolation (FDI) approaches. An accurate and reliable FDI system will ensure that the aircraft height estimate is robust to sensor failure.

This chapter initially gives an overview of the fault detection and isolation system. The discrete operating modes are presented and the important design decisions are explained. The strategy to generate representative sensor measurement data for the training of data-driven approaches and establishment of test datasets is described. Thereafter a variety of FDI evaluation tools that will be necessary to evaluate the performance of all FDI approaches is discussed. The machine learning library selection and classifier training tools are also presented. Lastly the chapter ends with a discussion on how the FDI system will be demonstrated using aircraft landings where a sensor failure is experienced.

4.1 System Overview

The fault detection and isolation system ensures that the aircraft height estimate is robust to sensor failure by correctly identifying a faulty sensor so that it can be excluded. An overview of the available input sensor measurements and database values is illustrated in figure 4.1. The fault detection and isolation (FDI) system gives a discrete output called the sensor status which corresponds to the operating mode for the group of height and altitude sensors.

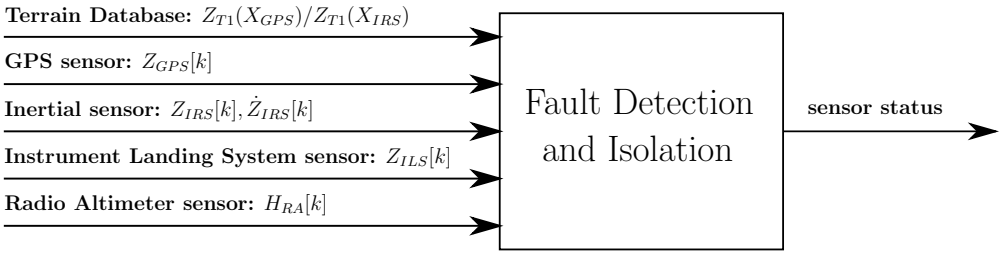


Figure 4.1: An overview of the available inputs and outputs for fault detection and isolation.

The fault detection and isolation approaches will each have their own unique architectures that can make use of any of the listed inputs and are required to give a single discrete output. There are five different sensor statuses as illustrated in table 4.1 that

account for all possible outputs based on the assumption that only one sensor technology failure can occur at a time. For the sake of notation in this work a sensor status can be referred to as a hypothesis and is reflected using the notation θ_i.

Discrete Status	Hypothesis	Explanation
1	$\theta_{\text{no\_fault}}$	All sensors normal
2	$\theta_{\text{GPS\_fault}}$	Global Positioning System fault
3	$\theta_{\text{IRS\_fault}}$	Inertial Reference System fault
4	$\theta_{\text{ILS\_fault}}$	Instrument Landing System fault
5	$\theta_{\text{RA\_fault}}$	Radio Altimeter fault

Table 4.1: Summary of sensor statuses.

4.2 Design Decisions

Two design decisions were made with regards to the approach towards fault detection and isolation. The first design decision was to convert all sensor measurements to relative altitude and to perform fault detection and isolation in the relative altitude domain, instead of in the height domain. The second design decision was to consider two general approaches: fault diagnosis using sensor measurements from a single time instant, and fault diagnosis using sensor measurements from a window of consecutive time instants. The motivation for these respective decisions is discussed below.

4.2.1 Height versus Relative Altitude

This section motivates the design decision to convert all sensor measurements to relative altitude, and to perform fault detection and isolation in the relative altitude domain, instead of in the height domain. When converting all sensor measurements to the relative altitude domain, only the radio altimeter measurement will be affected by the errors in the onboard terrain database. Likewise when converting all sensor measurements to the height domain, all the other sensors (GPS, IRS, and ILS) will be affected by the errors in the onboard terrain database. To convert from relative altitude to height, the relative altitude of the terrain must be subtracted from a relative altitude measurement.

$$H_{GPS}[k] = Z_{GPS}[k] - Z_{T1}(X_{GPS}[k]) \tag{4.2.1}$$
$$H_{IRS}[k] = Z_{IRS}[k] - Z_{T1}(X_{GPS}[k]) \tag{4.2.2}$$
$$H_{ILS}[k] = Z_{ILS}[k] - Z_{T1}(X_{GPS}[k]) \tag{4.2.3}$$

When converting from height to relative altitude, the height measurement must be added to the relative altitude of the terrain.

$$Z_{RA}[k] = H_{RA}[k] + Z_{T1}(X_{GPS}[k]) \tag{4.2.4}$$

When working in the relative altitude domain, only the radio altimeter height measurement must be converted to a relative altitude measurement, while all the other sensors (GPS, IRS, and ILS) already provide measurements in the relative altitude domain. When

working in the height domain, only the radio altimeter already provides height measurements, and all the other sensor measurements must be converted from relative altitude measurements to height measurements. Furthermore, when model-based approaches such as the Kalman filter are used, the aircraft state is more naturally propagated in relative altitude than in height. The relative altitude domain is aligned with inertial space and therefore obeys the equations of motion, and changes "smoothly", while the height is a function of the terrain and may therefore change abruptly.

4.2.2 Single versus Consecutive Time Instants

The distinction was made between using measurements from a single time instant or a window of consecutive measurements to perform fault diagnosis. This distinction was not intentional, but rather naturally occurred as the project progressed. An initial thought was to define a region in hyperspace where no-fault data points lie and identify faulty sensor measurements based on their location relative to this no-fault region using machine learning algorithms. A data point is the term used to describe measurements from a single time instant, where each dimension represents a different sensor measurement. Both outlier and binary classification techniques were investigated for fault detection. The outlier techniques require no prior knowledge in the form of labels, but rather derive their own structure from the data. The binary classifiers exploit the prior knowledge provided by labels to determine the "optimal" decision boundary. Naturally the knowledge of a fault being active is not beneficial without knowing exactly which sensor is faulty and therefore the use of multi-class classifiers is necessary.

The next focus of this project was to investigate a variety of classical and novel fault detection and isolation methods found in literature. These approaches seem more natural as they look at a window of consecutive measurements when attempting to detect and isolate a faulty sensor. Deviations are much easier to identify when incorporating a time history of measurement and trajectory data. The use of a window or consecutive time instants also allows for the incorporation of detection and isolation memory that can be used to improve performance. The best performing single time instant method was also considered under the consecutive time instant approach by giving it the ability to remember previous detections and isolations. This allows for an objective comparison in performance as there are distinct properties to both the single and consecutive time instant approaches. Considering the measurements from each time instant as unique allows for flexibility and ensures that there is no bias towards a previous prediction, while using a window of consecutive measurements aims to improve performance by incorporating prior knowledge in the form of previous predictions, trajectory data, or measurement history.

4.3 Data Generation

The sensor measurement dataset as supplied by Airbus was recorded during a limited number of aircraft landings for a specific runway at Toulouse-Blagnac airport. This dataset only contains no-fault aircraft landings. The real dataset was not deemed rich enough to serve as training data for the data-driven approaches or diverse enough to serve as comprehensive test data for all approaches. The simulation set-up as discussed in Chapter 2.2 was therefore required to generate more landing runs that also contained examples of all the different types of sensor faults. The synthetic dataset was split into a training set

and a test set. The performance of the fault detection and isolation system was verified on the simulated dataset and validated on the real sensor measurement dataset supplied by Airbus. Similar to the simulated approach, synthetic faults were injected on top of the real dataset to create a variety of sensor failures. The behaviour of the system is therefore validated on a combination of real landings where no faults are present and real landings where faults are introduced by synthetically injecting faults.

4.3.1 Simulated Data

The data generation process for sensor data can be summarised as follows:

1. The simulation model is initialised with a random initial relative altitude along the glide slope, a random set of sensor configuration parameters and random atmospheric conditions. Additionally when simulating a sensor fault the altitude at which a fault occurs is randomly selected along with the fault parameters, while the fault profile is predetermined before simulation. When the simulation is run, the fault is injected onto the simulated measurement once the aircraft descends below the fault occurrence altitude.

2. Individual instants of measurement data from a landing simulation run are assigned both a binary and multi-class label. The binary label indicates whether that data point is "fault" or "no-fault", while the multi-class label indicates the actual sensor status.

3. The sensor data from the simulation model is manipulated with knowledge from the terrain database and stored in both the relative altitude and height domain.

The simulation start and failure location are considered to be random variables. The assumption is made that simulation starts when the aircraft meets the 3° glide slope. The initial relative altitude of the aircraft is drawn from a uniform distribution between the minimum and maximum relative altitude at which an aircraft can intercept the glide slope. The maximum simulation altitude is limited by the length of the terrain database and the 3° glide slope angle. The minimum initial simulation altitude is a value that is taken from the analysis of actual flight data. The magnitude of the wind gusts is also treated as a random variable to ensure that the model accounts for realistic sensor measurements in adverse weather conditions.

A systematic approach was taken to the simulation of faults. The assumption is made that a fault can occur at any given time along the glide path to ensure that the test data is representative of a range of possibilities. The location of the fault is drawn from a uniform distribution ranging between zero and the maximum relative altitude. The random variables for respective fault profiles are initialised prior to each simulation. The magnitude of the bias, noise and oscillating fault profiles are designed such that they will always meet the minimum threshold required to be a fault. This minimum threshold must be beyond a sensor's range of accuracy.

4.3.2 Training data

Training data was needed to train the single time instant fault detection and isolation approaches. The initial approach to establishing a training dataset was to randomly extract individual fault and no-fault data points from the simulated landing runs. This

proved problematic for two reasons. Firstly, the manner in which a fault was labeled led to the development of training datasets that had a large number of faulty data points that were situated deep within the no-fault region. All data points from the simulated landing runs that occur after the time of fault occurrence are labeled as such whether they meet the theoretical definition of being a fault or not. Some fault profiles such as a jamming fault or increased noise fault produce measurements that are actually within the no-fault region when considering individual data points. This resulted in an unacceptably high number of false alarms. Secondly, the random sampling strategy did not adequately cover the entire data space. Most fault profiles occupy the data space surrounding the fault boundary, while the regions further away have less data. Generating training data that is deemed diverse enough to adequately cover the data space would be computationally intensive and might still not cover unseen fault profiles.

Instead, the training dataset was constructed from randomly generated sensor measurements. This is a valid approach when classifying data points constructed using measurements from the same sampling instant, because whether or not a sensor is faulty is only dependent on the current measurements and not a time history. Malan [7] proposed the use of a method whereby random data points are generated from the union of a wide uniform distribution over the entire data space, and a narrow normal distribution around the fault boundary. The data points from the wide uniform distribution will ensure the correct identification of faults situated far away from the decision boundary, while the narrow distribution ensures that the classifier can correctly differentiate between data points situated in a close proximity to the theoretical decision boundary.

Separate sets of training data was established for both the outlier, binary and multi-class classifiers. The data-driven outlier detection algorithms were used as novelty detectors by training them on a single class. The training dataset consisted of no-fault data points and faults are identified according to their similarity to the known class. The training dataset for the binary classifier consisted of an equal percentage of fault and no-fault data to ensure an unbiased training dataset. This ensures that classifiers trained on this data would not be biased towards a specific class. A similar approach was used for the establishment of a training dataset for the multi-class classifier. The multi-class classifier is used to differentiate between active fault profiles and therefore its training data only consisted of fault data with equal splits from each sensor technology type. The label assigned to a data point corresponded to the sensor status associated with that failure.

4.3.3 Test data

Five separate test sets were established to ensure that the results obtained are consistent regardless of the initialization values for the simulation model or magnitude of the fault profiles. Each test set contained 30 simulated landing runs with an equal percentage of each sensor status and fault profile. Therefore there were 6 different landing runs for each of five possible sensor statuses. The no-fault sensor status comprised of six different simulation landing runs without any injected faults. The remaining sensor statuses each deal with a separate technology failure. The make up of the simulation runs for each sensor status representing a technology failure was chosen such that all of the six fault profiles were represented. In total the test data contains 150 unique landing runs distributed evenly across 5 smaller test sets. This will ensure fairness in testing and that results are not biased towards fault profiles that are easier to detect than others.

4.3.4 Validation data

The dataset supplied by Airbus contained actual sensor measurement data for a limited number of aircraft landings at a specified runway at Toulouse-Blagnac Airport. This data was reserved for the validation of the fault detection and isolation results obtained using the test data above. The supplied dataset consisted of 26 aircraft landing runs of no-fault sensor data. Synthetic errors from each of the fault profiles were injected onto the measurement data for individual landings. The validation dataset consisted of an 24 faulty runs with each of the four sensors having six separate fault profiles. The remaining two landings runs were used for no-fault testing. It is unfortunately not possible to assign an equal number of landing runs to each sensor status. Each landing run is actually a no-fault landing run until the time of fault occurrence. Therefore the majority of the individual measurement instances will be no-fault data points.

4.4 FDI Evaluation Tools

The sections discusses procedures used to quantify the performance and reliability of the fault detection and isolation architectures.

4.4.1 Detection and Isolation Accuracy

The primary evaluation criteria for a FDI architecture is fault detection and isolation accuracy. There are separate metrics, where detection accuracy is defined as the percentage of data points that the architecture correctly labels as fault or no-fault. There are four possible outcomes when assessing fault detection as illustrated in the table 4.2.

	Actual fault	**Actual no-fault**
Predicted fault	True positive	False positive
Predicted no-fault	False negative	True negative

Table 4.2: Possible outcomes summarised in a contingency table.

where,

- True positive - FDI architecture predicts fault and it actually was a fault
- False positive - FDI architecture predicts fault and actually was a no-fault
- False negative - FDI architecture predicts no-fault and actually was a fault
- True negative - FDI architecture predicts no-fault and actually was a no-fault

Similarly, the fault isolation accuracy is defined as the percentage data points that the FDI architecture assigns the correct sensor status too. These two metrics are evaluated separately as some FDI approaches are designed for fault detection, while others are better suited for isolation.

4.4.2 Statistical Significance Test

Statistical significance tests will be performed on architectures A and B when there is a marginal difference in accuracy results. McNemar's test ensures that the size of the test set is large enough to ensure the difference in classification results are statistically significant.

	Architecture A correct	**Architecture A incorrect**
Architecture B correct	a	b
Architecture B incorrect	c	d

Table 4.3: McNemar's statistical significance test contingency table.

Where McNemar's test statistic is given by:

$$\chi^2 = \frac{(b-c)^2}{b+c} \tag{4.4.1}$$

The χ^2 test statistic is a Chi-squared distribution with one degree of freedom with results considered to be statistically significant for probability threshold of 95 %. Essentially for $\chi^2 > 3.841$ it can be confidently stated that the test set is large enough to prove that the difference in results between the two architectures is statistically significant and not attributed to random chance [52].

4.4.3 Evaluation of Incorrect Classifications

There are three possible outcomes when analysing the incorrect classifications associated with a FDI architecture with numerous possible sensor statuses. A false alarm, also known as false positive, is the first possible outcome that occurs when the FDI architecture predicts a sensor is faulty, while there actually was not a fault. The location of false alarms is an important measure of FDI architecture reliability and it is essential that false alarms are avoided. A missed opportunity or false negative is the second possible outcome and occurs when the system predicts that there is not a fault, while there actually is a fault present. The last possible outcome is incorrect isolation and occurs when the FDI architecture detects an actual fault, but then isolates the incorrect sensor. An analysis is done on the location of the false alarms, missed opportunities and incorrect isolations to ensure that they are located in close proximity to the fault boundary. Minimisation of false alarms, missed opportunities, and incorrect isolations are of utmost importance for a safety critical function such as aircraft height estimation. Normal sensor data points that are well within the domain of normal operation should never generate a false alarm and likewise faulty sensors that exceed the definition of an extreme outlier should not go undetected.

4.5 Machine Learning Tools

A machine learning library was used to train and evaluate numerous of the data-driven approaches. The motivation behind which library was chosen and the procedures used to ensure algorithms are effectively trained is discussed in this section.

4.5.1 Machine Learning Library Selection

An open source machine learning library will be used to evaluate the performance of different classifiers. The following criteria were considered when evaluating a choice of machine learning library:

1. Variety of classifiers
2. Quality of supporting documentation and API for classifiers
3. Classifiers must be configurable
4. Programming language
5. Must be an open source library

Research was done on available machine learning libraries and is summarised in table 4.4. It was decided that scikit learn best met the criteria stated above and would be used to implement the classification algorithms. MATLAB was used to implement all other algorithms that did not require prior training with a machine learning library.

Library	**Programming language**	**Comments**
scikit learn	Python	well documented
TensorFlow	Python	mostly deep learning
PyLearn2	Python	limited documentation
Apache Spark MLIB	Java, Python, Scala, R	scalable framework
Weka	Java	backed by Waikato University
Shogun	C++	few binary classifiers
mlpack	C++	extensive documentation although not easy to navigate
Statistics and Machine Learning Toolbox	MATLAB	not open source

Table 4.4: Machine Learning Libraries

4.5.2 Classifier Training Tools

Learning Curve

A learning curve is used to ensure that a classifier is trained properly. Classifier accuracy is plotted as a function of training dataset size. A classifier is properly trained when its accuracy does not improve with an increase in the number of data points used for training. Smaller training datasets may result in a classifier that is poorly trained and will yield lower classification accuracies, while larger datasets are more computationally intensive to train on.

K-Fold Cross Validation

K-fold cross validation is used to evaluate the representativeness of a training dataset. It is a systematic approach that divides a dataset into K equal partitions. K-1 partitions are used as the training dataset, while the remaining partition forms the cross-validation dataset used to evaluate classifier accuracy. This is repeated K times with a different partition being used as the cross-validation dataset for each repetition. K-fold cross-validation ensures that classifier results are consistent regardless of the validation set chosen. Ideally a representative dataset will yield similar accuracies regardless of which fold is used as the cross validation set.

Receiver Operating Curve

A Receiver Operating Characteristic (ROC) curve is used to choose an optimal decision threshold and evaluate the sensitivity of the classifier's performance to the decision threshold. The optimal unbiased decision boundary is one that produces an equal number of false positives (false alarm) and false negatives (missed opportunities). A ROC curve is established by plotting the True Positive Rate (TPR) versus the False Positive Rate (FPR) for different decision boundaries associated with a classifier. The TPR and FPR are given by:

$$TPR = \frac{\sum \text{True positive}}{\sum \text{Actual fault}}$$
$$FPR = \frac{\sum \text{False positive}}{\sum \text{Actual no-fault}}$$

The shape of a ROC curve is determined by the overlap between the fault and no-fault class distributions. For explanation purposes a Gaussian distribution is considered.

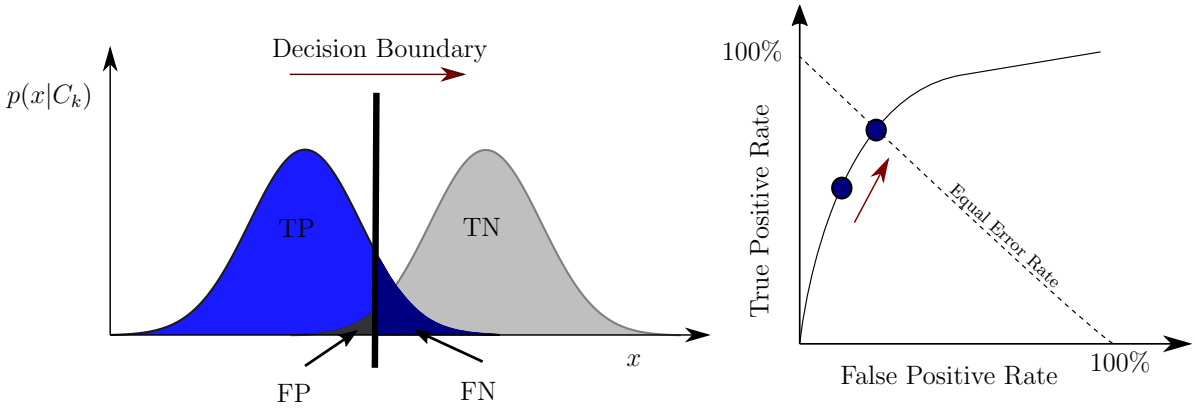


Figure 4.2: Receiver operating characteristic curve showing as the decision boundary is moved to the right, the ROC curve will approach the optimal position along the equal error rate line.

The further the distributions illustrated in figure 4.2 are apart, the better the classifiers performance will be. The vertical line shown between the fault and no-fault distributions indicates the decision boundary for classification. Adjusting this threshold will result in a change in the operating point on the ROC curve. The optimal operating point is a threshold that is unbiased, producing an equal percentage of false alarms (FPR) and

missed detections (100 %-TPR). The intersection between the Equal Error Rate (ERR) and ROC curve coincides with the optimal operating point for a classifier.

4.6 FDI Demonstration

The fault detection and isolation performance of the various architectures will be demonstrated on the same landing run with four different injected fault profiles as illustrated in figure 4.3-4.6. A visual demonstration of how the actual versus predicted sensor status changes with time allows for a better understanding of the architecture's performance and highlights any shortfalls that were raised during training or evaluation. Four of the six fault profiles were chosen that are the hardest to detect and best represent a variety of possible failures. These are a bias fault, a jamming fault, an increased noise fault and an oscillation fault.

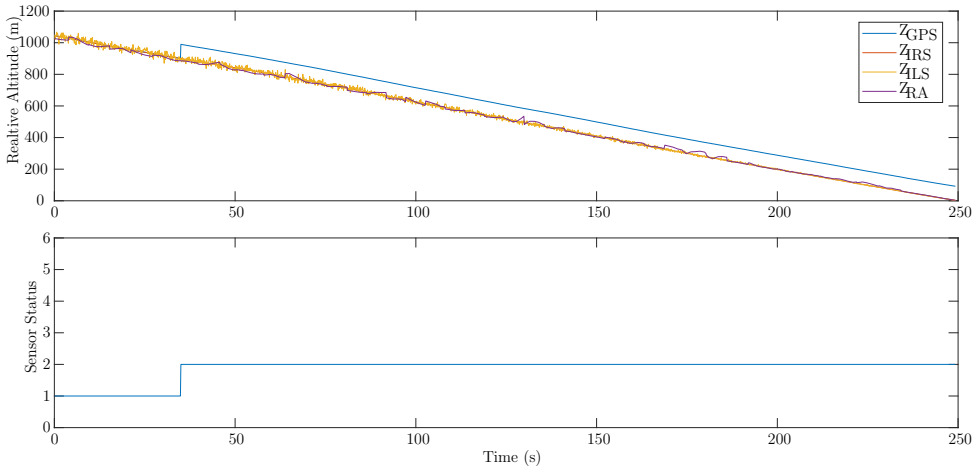


Figure 4.3: The upper plot depicts a landing run with a GPS bias fault. The lower plot shows the actual sensor status.

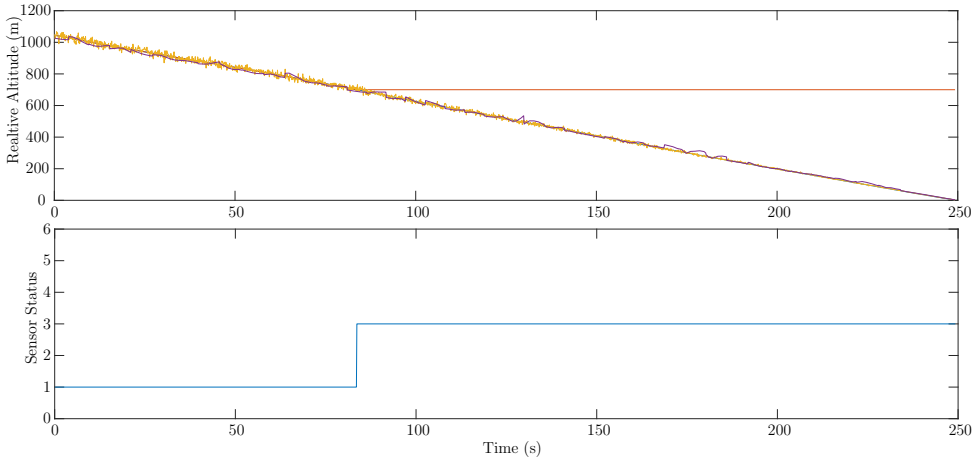


Figure 4.4: The upper plot depicts a landing run with a IRS jamming fault. The lower plot shows the actual sensor status.

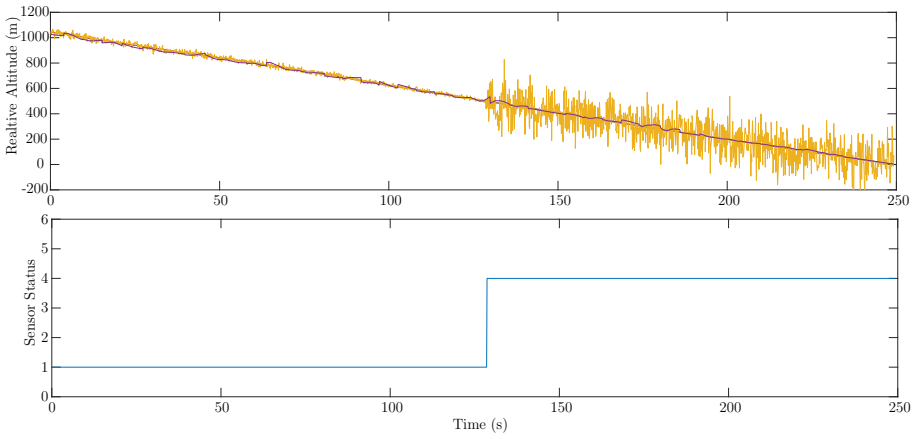


Figure 4.5: The upper plot depicts a landing run with an ILS increased noise fault. The lower plot shows the actual sensor status.

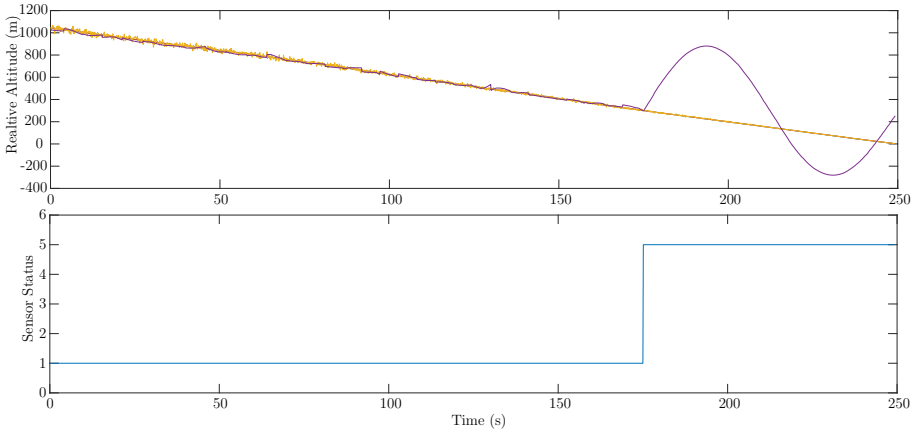


Figure 4.6: The upper plot depicts a landing run with a radio altimeter oscillation fault. The lower plot shows the actual sensor status.

4.7 Conclusion

This chapter has introduced and discussed the necessary information and background knowledge to design and evaluate fault detection and isolation architectures. Fault detection and isolation was performed in the relative altitude domain to mitigate the effect of the non-linearities associated with the terrain database. The simulation model was used to establish the test datasets, while the results are validated on actual sensor measurement data. A novel method for establishing a training dataset from randomly generated data points was explained. The procedures used to quantify the performance and reliability of the fault detection and isolation architectures were discussed. This section ends by discussing the simulated landing runs (with injected faults) that were used to demonstrate the fault detection and isolation performance by plotting the sensor status as a function of time. Now that the necessary background information and assessment tools have been established, the design and evaluation of the respective single and consecutive time instant fault detection and isolation techniques will be presented.

Chapter 5

Single Time Instant Fault Diagnosis

This chapter presents the design and evaluation of a fault diagnostics system for a commercial aircraft using sensor measurements from a single time instant. The system investigates the use of outlier, binary and multi-class classifiers for fault detection and isolation.

The chapter starts off by presenting the conceptual design and architecture. A computational complexity study of the various approaches is then performed. Thereafter, the classifier training tools are used to ensure that the outlier detection methods, binary classifiers and multi-class classifiers are properly trained. An evaluation of the fault detection and isolation performance is then presented. The results are also validated on actual aircraft data. Lastly the top performing fault detection and isolation architectures are demonstrated on simulated landing runs with faults injected on certain sensor measurements to show how the actual and predicted sensor status changes as a function of time.

5.1 Conceptual Design and Architecture

A single time instant fault diagnostic system comprises of two separate stages. The sampled measurements (X) are preprocessed before performing fault detection and isolation. The preprocessing of the sensor data (X) aids the classifier performance by improving the class separability. In the second stage the preprocessed data (X′) is analysed using classification algorithms to detect when a fault occurs. This will activate the fault isolation classifier that determines which sensor measurement is problematic.

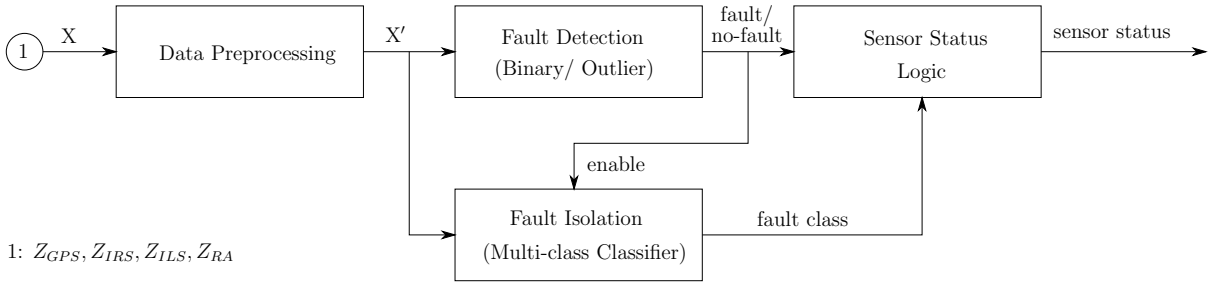


Figure 5.1: The architecture for a single time instant fault detection and isolation.

5.1.1 Data Preprocessing

A variety of preprocessing techniques were investigated with the aim of improving classifier performance as the fault diagnostics requires the learning of complex and often non-linear decision boundaries. Through the manipulation of the training dataset the separability of the respective classes can be improved. The techniques chosen make use of different methods and approaches. Table 5.1 below provides a summary of chosen techniques. Preprocessing was applied to all the test datasets and their performance was compared against a baseline of an unprocessed dataset. Based on the results, the best performing preprocessing technique can be applied online to data stream X.

Technique Name	**Description**
Unprocessed	Original Dataset
Orthogonal Transform	Subspace rotation
Kernel Transform	Projection into higher-dimensional space
Residual Transform	Proposed method that establishes new dataset using measurement residual

Table 5.1: Summary of preprocessing techniques.

Orthogonal Transform

An orthogonal transform is the rigid rotation of the coordinate axis system. This fixed rotation is applied to the data space such that the new representation of the data points becomes uncorrelated. An orthogonal transform is illustrated in figure 5.2 where it is not apparent that Data Point A is an outlier when considering its projection onto the original coordinate axis system. However rotating the coordinate axis system such that the new representation of the features becomes uncorrelated improves class separability when projecting the data points onto the rotated axes. For example in figure 5.2 the projection of Data Point A on axis x_2' will distinguish it from the other data points whose projections are closer to the origin.

The transformed dataset is calculated using an uncentred rotation about the origin:

$$Y = V^T X \tag{5.1.1}$$

where Y is the transformed dataset, V is the transformation matrix, and X is the original dataset.

An orthogonal transform projects the original data points onto the principal axes that describe the underlying structure of the data. These principal axes coincide with the directions of maximum variance [36]. Principal Component Analysis (PCA) as discussed in Chapter 3.8.2 is performed offline on a static dataset of no-fault data points to identify the transformation matrix (V) that consists of individual eigenvectors arranged in columns such that $V = [\vec{v_1}^T, \vec{v_2}^T, .., \vec{v_d}^T]$. The vector ($\vec{v_1}$) that maximises the projection of the data is called the first principle component. For example, the principal component

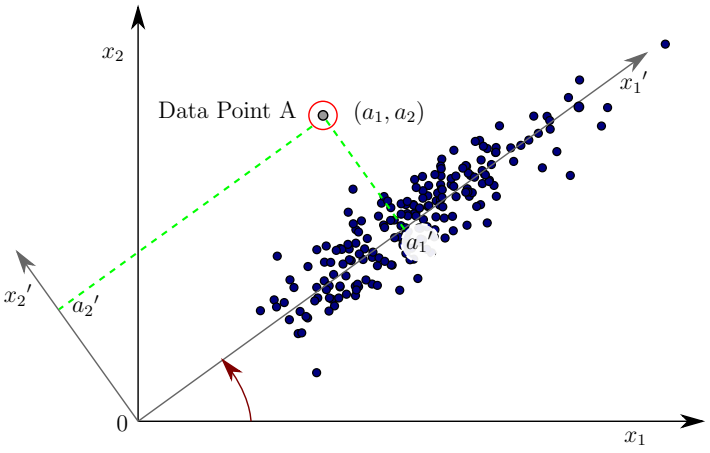


Figure 5.2: The projection of a Data Point A with original coordinates (a_1, a_2) to the rotated axes system with coordinates (a_1', a_2'). Data Point A is an outlier. The first principal axis x_1' is in the direction that maximises the variance of the projected data points.

in figure 5.2 coincides with axis x_1'. Subsequent principal components are chosen such that they maximise the projected variance along all possible orthogonal vectors to those already considered. The orthogonal transform is applied online to new data points such that there rotated coordinates are given by:

$$\vec{Y_k} = V^T \vec{X_k} \tag{5.1.2}$$

where k is the sampling instant.

Kernel Transform

Kernel functions project a d dimensional input space into a different subspace where respective classes become linearly separable as illustrated in figure 5.3. The central concept known as the kernel trick is that the inner product $<x_i, x_j>$ can be substituted with a kernel function $K(x_i, x_j)$. This is best explained using the derivation of Support Vector Machine as proposed by Boser et al [53] and discussed in Chapter 3.5.5.2.

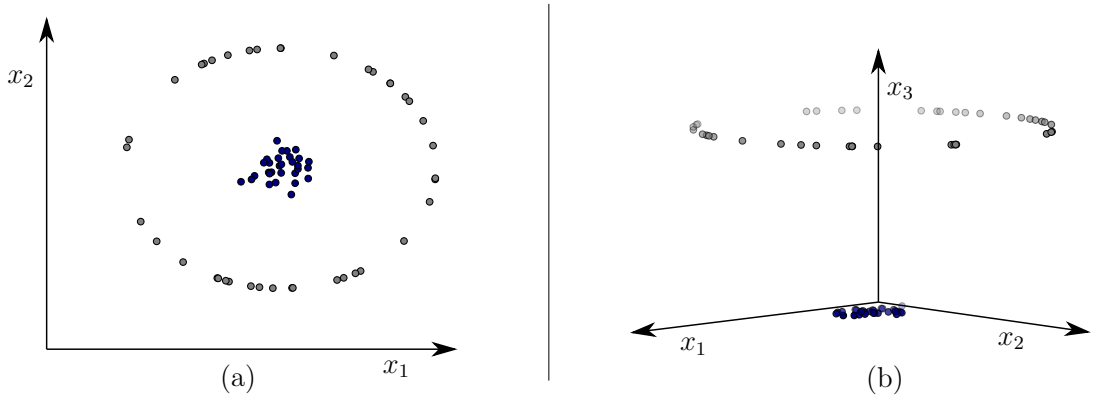


Figure 5.3: The input dataset (a) is linearly inseparable. The kernel function projects the example two dimensional dataset into a different subspace where the different classes become linearly separable as illustrated in (b).

The kernel functions that were considered include linear, polynomial, Radial Basis Function (RBF), and sigmoid. The choice of kernel function used is largely dependent on the specific application and data structure. For the intended application of improving class separability amongst the group of aircraft height and altitude sensors the Radial Basis Function (RBF) kernel is used. New features are introduced that are a measure of similarity between the individual sensor measurements. This is well-suited to the problem of fault detection and isolation where faults can be viewed as outliers that will differ from the majority of the data. This degree of outlierness will be reflected in the measure $K(x_i, x_j)$ and improve class separability.

$$K(x_i, x_j) = e^{\gamma||x_i - x_j||^2} \tag{5.1.3}$$

where γ is the kernel coefficient.

The measure of similarity used by the kernel is based on Euclidean distance. As the Euclidean distance between sensor measurements tends to zero, so $K(x_i, x_j) \to 1$. Preprocessing therefore involves the incorporation of additional features to the dataset, where $x_k = K(x_i, x_j)$ and $i \neq j$ for all combination of features in the input dataset.

Residual Transform

The residual transform is not a standard preprocessing technique, but rather a model-based preprocessing technique that is specifically proposed for the robust height verification problem. The premise is that a single data point consists of several different measurements of the same true variable. The differences between the measurements that constitute this data point therefore reflect a measure of consensus that can be exploited.

The residual transform first calculates an estimate of the true variable (aircraft relative altitude), before calculating the differences (residuals) between the instantaneous measurements and the estimated relative altitude. The residuals represent the instantaneous measurement error for each instantaneous measurement. The estimate of the true variable may be obtained in various ways, for example by taking the weighted average of the measurements, or by taking the median of all the measurements. The residual transform therefore provides preprocessing that allows the classification algorithm to perform its classification based on the estimated relative altitude of the aircraft and the instantaneous sensor errors instead of on instantaneous sensor measurements.

For example, in the robust height estimation problem a single data point consists of the relative altitude measurements obtained from the GPS sensor, the inertial sensor, the instrument landing system (ILS), and the radio altimeter (combined with the terrain database). All four sensors provide an estimated measure of the true relative altitude, and each sensor has a different measurement accuracy. The residual transform first calculates an estimate of the aircraft relative altitude (using the weighted average or median of the sensor measurements), and then calculates the difference (residual) between each sensor measurement and the "true" relative altitude as illustrated in figure 5.4. The residuals then represent the instantaneous relative altitude measurement errors for each of the four sensors.

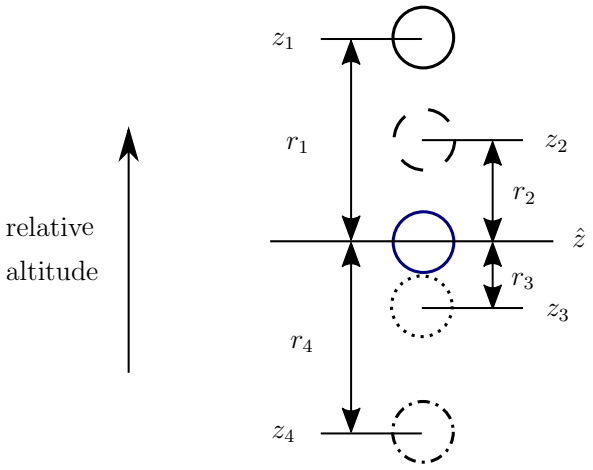


Figure 5.4: The residual transform manipulates the original dataset to output a new feature space consisting of a fused relative altitude estimate for the aircraft and residual features. The different sensor measurements are represented by the solid and dashed black circles and the estimate by the blue circle.

The estimate of the true relative altitude may be calculated using the weighted average of the relative altitude measurements, as follows:

$$\hat{z} = \frac{\sum_{i=1}^{d} \frac{z_i}{\sigma_i{}^2}}{\sum_{i=1}^{d} \frac{1}{\sigma_i{}^2}} \tag{5.1.4}$$

where $\hat{z}$ is the estimated relative altitude, d is the number of sensors, z_i is the instantaneous sensor measurement of the i^{th} sensor, and σ_i is the known standard deviation of the sensor measurements for the i^{th} sensor

However, the disadvantage of this approach is that the weighted average may be significantly skewed by a sensor measurement from a faulty sensor, especially if the sensor is deemed more accurate or usually has a small measurement noise variance. A more robust approach would be to calculate the estimate of the aircraft relative altitude using the median of all the sensor measurements, as follows:

$$\hat{z} = \text{median}\{z_1, z_2, z_3, z_4\} \tag{5.1.5}$$

The advantage of using the median is that the estimate of the aircraft relative altitude will not be skewed by an extreme outlier measurement provided by a faulty sensor, and that there is no need for prior knowledge in the form of theoretical sensor accuracies or measurement noise variances.

Once an estimate of the aircraft relative altitude has been obtained, the residuals are calculated as follows:

$$r_i = ||\hat{z} - z_i|| \tag{5.1.6}$$

A transformed data point is therefore constructed by concatenating the estimated relative altitude with the calculated measurement residual for each sensor, as follows:

$$X_i = [\hat{z}, r_1, r_2, r_3, r_4]^T \tag{5.1.7}$$

The residual transform therefore transforms the original dataset that consists of the instantaneous relative altitude measurements into a new dataset that consists of the estimated relative altitude and the instantaneous sensor measurement residuals. The residual transform increases the dimensionality of the dataset from d to d+1.

5.1.2 Fault Detection and Isolation

The design decision was made to use an architecture that comprised separate fault detection and isolation stages. This allowed for a wider variety of algorithms to be investigated and proved to be more accurate than an architecture that only used a multi-class classifier. The problem of identifying the specific sensor that failed is inherently multi-class and an additional architecture described in Appendix C.1 was considered and evaluated alongside the architecture presented in figure 5.1. The respective architectures were compared on the same sets of testing data as shown by the results in Appendix C.2, but it was found that using a separate binary classifier for fault detection, and a multi-class classifier for fault isolation, yielded better results.

The data-driven outlier detection algorithms that were investigated include the Elliptic Envelope, Local Outlier Factor and Isolation Forest. The binary and multi-class classifiers include Logistic Regression, Naïve Bayes, Decision Tree, k-Nearest Neighbors and Support Vector Machine. A binary classifier or outlier detection technique analyses X' and reflects a binary classification outcome as fault or no-fault. The multi-class classifier is enabled when a fault occurs and then isolates the respective faulty sensor. This is reflected in the fault class: GPS fault, IRS fault, ILS fault or RA fault. The design decision was made that the multi-class classifier would not be activated until a fault occurs to mitigate the need for unnecessary computational resources. The sensor status logic uses the outcomes of the binary classifier to determine when a fault occurs, whilst the fault class reflects which sensor is faulty. The possible sensor statuses for the combination of altitude and height sensors are: no-fault, GPS fault, IRS fault, ILS fault or RA fault.

The sensor status logic is an intuitive logic check based on the outcomes of the fault detection and isolation classifiers that determines the sensor status.

Algorithm 2 Pseudocode to determine the Sensor Status.

Input:

fault/no-fault	output of binary classifier
fault class	output of multi-class classifier

Output:

sensor status	operational status of height/altitude sensors

```
1: function SENSOR_STATUS_LOGIC(fault/no-fault,fault class)
2:     if no-fault then
3:         sensor status = no-fault
4:     else
5:         sensor status = fault class
6:     end if
7: return(sensor status)
```

5.2 Computational Complexity Study

The computational complexity of classifying a new data point is examined for each data-driven technique that will be used in the single time instant fault detection and isolation architecture. The computational complexity is summarised in the table 5.2 below. N is number of training data points, d is the dimensionality of the dataset and k is the number of different classes.

Approach Name	**Detection Complexity**	**Isolation Complexity**
Logistic Regression	$\mathcal{O}(d)$	$k{\cdot}\mathcal{O}(d)$
Naïve Bayes	$\mathcal{O}(d)$	$\mathcal{O}(kd)$
Decision Tree	$\mathcal{O}(\log(2)^{\text{Tree depth}})$	$\mathcal{O}(\log(2)^{\text{Tree depth}})$
Nearest Neighbors	$\mathcal{O}(dN\log(N))$	$\mathcal{O}(dN\log(N))$
Support Vector Machine	$\mathcal{O}(d^2)$	$\frac{1}{2}k(k-1)\cdot\mathcal{O}(d^2)$
Elliptic Envelope	$\mathcal{O}(d)$	-
Local Outlier Factor	$\mathcal{O}(dN\log(N))$	-
Isolation Forest	$\mathcal{O}(\log(2)^{\text{Tree depth}})$	-

Table 5.2: Summary of the computational complexity for numerous classifiers.

The computational complexity to evaluate a new data point for the binary Logistic Regression architecture is $\mathcal{O}(d)$ [7]. The multi-class Logistic Regression uses a one-versus-all approach and establishes k binary classifiers and hence has a computational complexity of $k \cdot \mathcal{O}(d)$. The computational complexity of the Naïve Bayes increases linearly with the dimensionality as the class conditional probabilities need to be determined for each dimension, resulting in a computational complexity $\mathcal{O}(d)$. Its isolation (multi-class) computational complexity is therefore also proportional to the number of classes. The brute force method for the k-Nearest Neighbors requires N Euclidean distance computations that have a time complexity of $\mathcal{O}(dN^2)$. However, when using a K-D Tree structure, the computational complexity is reduced to of $\mathcal{O}(dN\log(N))$ [34]. Its isolation computational complexity is identical to the detection complexity as the same mechanism is used to classify new data points. The computational complexity of the Support Vector Machine depends on the type of kernel used. Linear kernels have a computational complexity of $\mathcal{O}(d)$, while other kernel types have a computational complexity of $\mathcal{O}(dN_{SV})$, where N_{SV} is the number of support vectors used. This can usually be approximated by $\mathcal{O}(d^2)$ [54]. The scikit-learn multi-class Support Vector Machine uses a one-versus-one architecture that establishes $\frac{1}{2}k(k-1)$ binary classifiers. The computational complexity to classify a new data point using an Elliptic Envelope involves a Mahalanobis distance calculation between the data point and the centre of the distribution. The computational complexity is therefore proportional to the dimensionality and is given by $\mathcal{O}(d)$. The Local Outlier Factor's computational complexity is identical to k-Nearest Neighbors as they both use similar mechanisms [45]. Similarly, an Isolation Forest constructs a binary tree, and therefore has a similar computational complexity to that of the Decision Tree.

The Logistic Regression, Naïve Bayes, Decision Tree, Elliptic Envelope and Isolation Forest have favourable detection computational complexities. The k-Nearest Neighbors and Local Outlier Factor are dependent on the number of training data points, N, and will therefore be computationally expensive when using a large training dataset. The inherently multi-class approaches are favorable as they have similar or identical computational complexities to their binary counterparts. The Naïve Bayes and Decision Tree classifiers are therefore the most suitable for use on multi-class problems with large datasets, where the computational complexity is to be minimised.

5.3 Classifier Training

The classifier training tools presented in Chapter 4.5.2 are now applied to ensure the outlier detection, binary and multi-class classification algorithms are properly trained. The following training process was used:

1. The optimal size of training dataset is determined for each of the data preprocessing and classification techniques using the five-fold cross validation. This allocates 80 % of the data to training, whilst the remaining 20 % is used as a validation set.

2. A receiver operating characteristic (ROC) curve is established for each binary and multi-class classifier. A ROC curve ensures that the optimal unbiased decision boundary is used for classification, when using the default algorithm parameters. The sensitivity of each binary and multi-class algorithm to the decision boundary is also analysed.

5.3.1 Learning Curves

The complete set of learning curves for the outlier detection, binary and multi-class classifiers are available in Appendix C.3. The default configuration parameter values were used along with the training dataset as discussed in Chapter 4.3.2. An example of a learning curve for a Logistic Regression binary classifier is illustrated in figure 5.5. The mean accuracy when applying the K-fold methodology usually increases with the number of training data points, whilst the variation decreases. For clarity, the variation was only plotted for the no-preprocessing curve. A classifier is properly trained when its accuracy does not improve significantly with an increase in the number of training data points. The number of training data points was chosen such that each algorithm is well trained, whilst retaining computational feasibility. The conservative design decision was taken to use 30 000 data points for the size of the training dataset. Beyond this point the increase in classification accuracy for the k-Nearest Neighbors and Support Vector Machine become negligible as illustrated in figure C.15 and figure C.16 respectively.

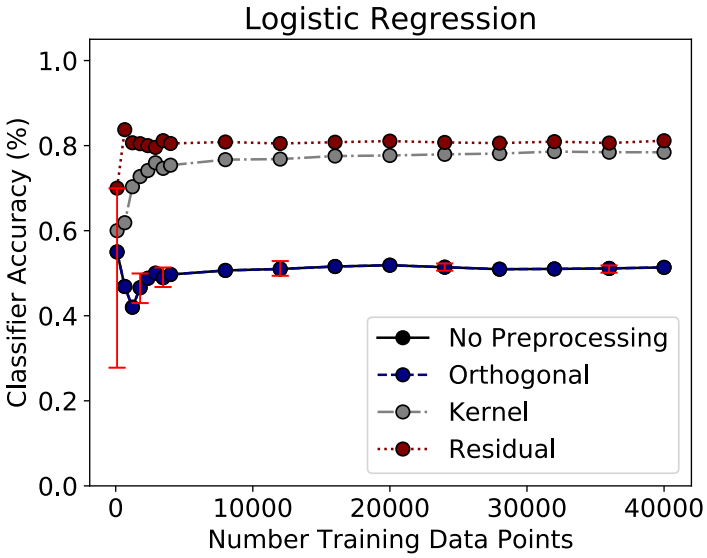


Figure 5.5: The learning curve for a Logistic Regression binary classifier. The variation in results is shown in red. The no-preprocessing learning curve is under the orthogonal learning curve.

5.3.2 ROC Curves

The Receiver Operating Characteristic curves were established for each binary and multi-class classifier using the training dataset. The complete set of ROC curves are available in Appendix C.4. The ROC curves give an early indication of promising techniques. The ROC curve ensures that the optimal decision threshold will be used when evaluating a classification algorithm. The optimal decision threshold is one that produces an Equal Error Rate (ERR) of false alarms and missed opportunities, which coincides with the intersection between the black dashed line and the individual ROC curves as shown in figure 5.6.

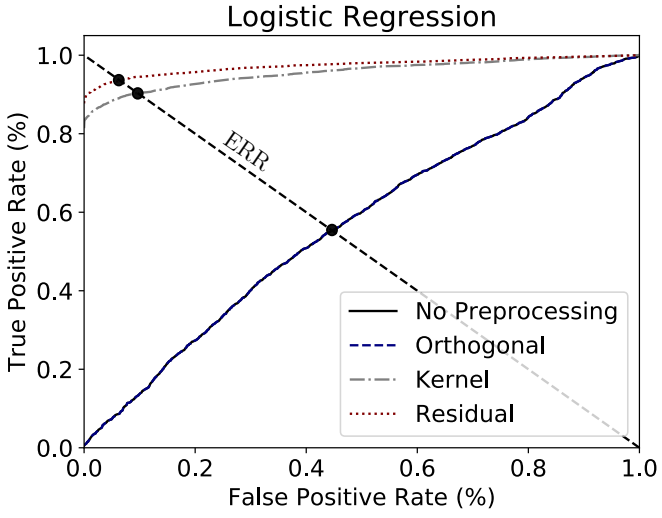


Figure 5.6: The receiver operating curve for the Logistic Regression binary classifier. The optimal decision threshold produces an Equal Error Rate.

In figure 5.6 the residual preprocessing technique shows the most potential as it maximises the area under the ROC curve, whilst the rotation and no preprocessing performs poorly as rotating the dataset fails to assist Logistic Regression in overcoming the non-linear decision boundary. There were a few observations made when analyzing the ROC curves in Appendix C.4. The kernel and residual preprocessing greatly aid the performance of a Logistic Regression architecture. Similarly, the kernel, orthogonal, and residual preprocessing greatly aid the performance of the Naïve Bayes architecture. The use of preprocessing for the remaining classification architectures has negligible impact on the performance. This is expected for the Support Vector Machine that has a built-in Radial Basis Function kernel, while the k-Nearest Neighbors and Decision Tree classifiers are well-suited for differentiating between classes with non-linear decision boundaries separating them.

5.4 Fault Detection and Isolation Evaluation

The outlier detection, binary and multi-class algorithms were optimally trained using 30 000 data points, and using the optimal decision thresholds established above. The respective fault detection and isolation architectures were evaluated on five different test sets as discussed in Chapter 4.3.3. The various classification-based fault detection and isolation architectures were evaluated in terms of accuracy for both fault detection and isolation, sensitivity of the configuration parameters, and the location of all incorrect classifications.

5.4.1 Fault Detection Accuracy

The detection accuracy is defined as the percentage of data points from the test set that the architecture correctly assigned the correct binary label (fault or no-fault). The fault detection results for various preprocessing techniques and classification based FDI architectures are given in figure 5.7, and summarised in table 5.3.

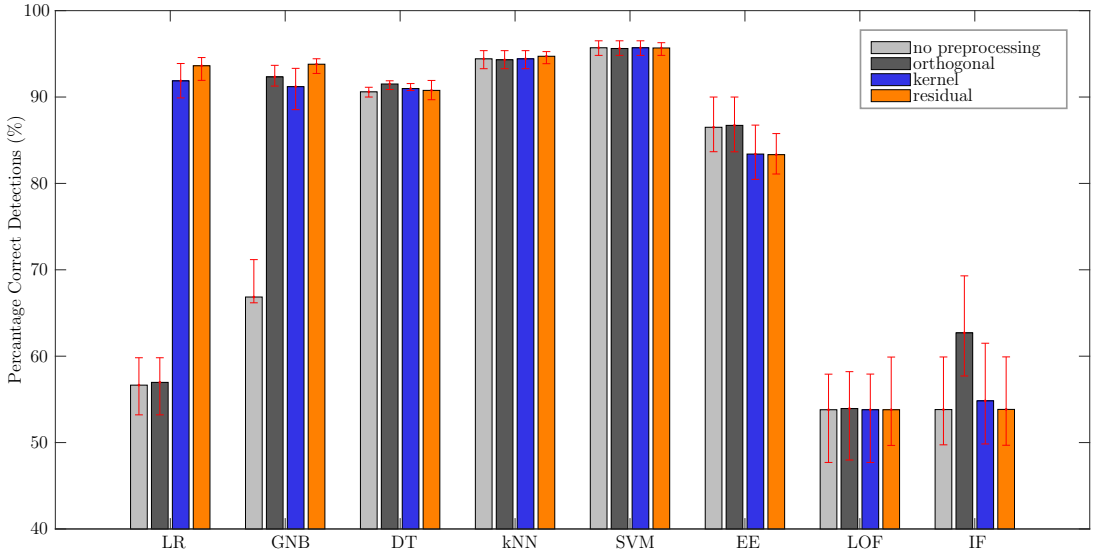


Figure 5.7: The detection accuracy for the various classification algorithms and preprocessing techniques. The minimum and maximum percentages are indicated in red with the mean accuracy shown by individual bars.

Rank	Classifier	Preprocessing	Accuracy
1	**Support Vector Machine (SVM)**	**kernel**	95.75 %
2	k-Nearest Neighbors (kNN)	residual	94.72 %
3	Gaussian Naïve Bayes (GNB)	residual	93.8 %
4	Logistic Regression (LR)	residual	93.62 %
5	Decision Tree (DT)	orthogonal	90.77 %
6	Elliptic Envelope (EE)	orthogonal	86.71 %
7	Isolation Forest (IF)	orthogonal	69.2 %
8	Local Outlier Factor (LOF)	no preprocessing	53.76 %

Table 5.3: Ranking the performance of the fault detection classifiers.

The Support Vector Machine FDI architecture with kernel preprocessing yielded the highest fault detection accuracy of 95.75 %. The worst performing classifier was Local Outlier Factor that yielded 53.76 % accuracy. It is observed that the binary classifiers outperform the outlier detection approaches. The binary classifiers use prior knowledge in the form of labels to establish the optimal decision boundaries that account for the varying sensor accuracies. The outlier detection algorithms are advantageous as they do not require prior labeling, and make no assumptions about the faults. However, their application is better suited to the detection of potential outliers in static datasets where prior information such as sensor accuracy is not available. McNemar's statistical significance test, as discussed in Chapter 4.4.2, was used to verify the results that were all within a 1 % overlapping interval. McNemar's test returned $\chi^2 = 6933.56$ for the statistical significance test between the k-Nearest Neighbors and Gaussian Naïve Bayes result in table 5.3. Similarly, the significance test between k-Nearest Neighbors and Logistic Regression returned $\chi^2 = 11.31$. This proves that the results shown in table 5.3 are statistically significant for a probability threshold of at least 95 %.

5.4.2 Fault Isolation Accuracy

The isolation accuracy is defined as the percentage of data points from a test set that the architecture correctly predicted the correct operating mode or sensor status. The fault isolation results for various preprocessing techniques and classification algorithms are given in figure 5.8, and summarised in table 5.4 when using the default configuration parameters.

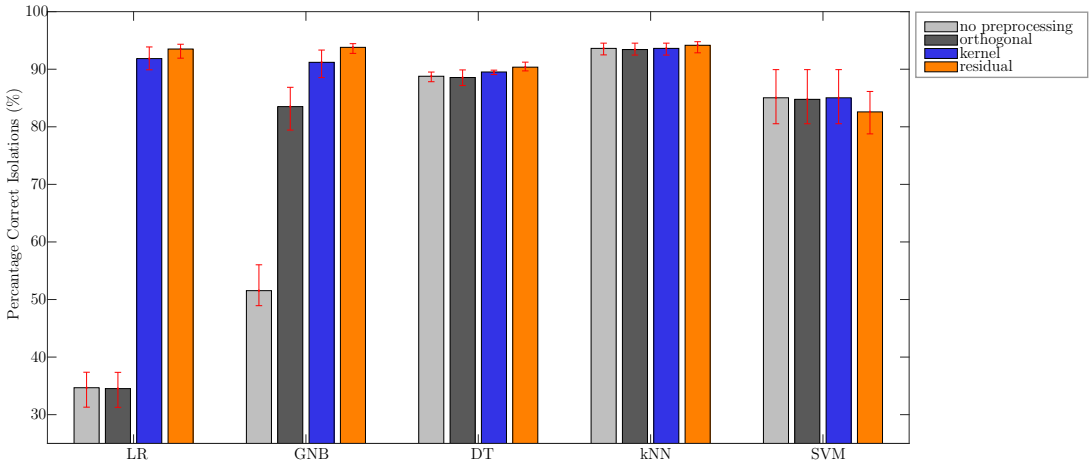


Figure 5.8: The isolation accuracy for the various classification algorithms and preprocessing techniques. The minimum and maximum percentages are indicated in red with the mean accuracy shown by individual bars.

Rank	**Classifier**	**Preprocessing**	**Accuracy**
1	**k-Nearest Neighbors (kNN)**	**residual**	94.16 %
2	Gaussian Naïve Bayes (GNB)	residual	93.8 %
3	Logistic Regression (LR)	residual	93.5 %
4	Decision Tree (DT)	residual	90.37 %
5	Support Vector Machine (SVM)	none	85.04 %

Table 5.4: Ranking the performance of the fault isolation classifiers.

The k-Nearest Neighbors architecture with residual preprocessing yielded the highest fault isolation accuracy of 94.16 %. The worst performing classifier was Support Vector Machine that yielded 85.04 % accuracy. The results within a 1 % overlapping interval were validated using McNemar's test. The Gaussian Naïve Bayes overlaps with both the k-Nearest Neighbors and Logistic Regression classifiers. The results differed enough to be statistically significant with $\chi^2 = 173.25$ (GNB vs kNN) and $\chi^2 = 387.98$ (GNB vs LR). It is clear from both the detection and isolation results that preprocessing aids Logistic Regression that makes use of linear decision boundaries, and Naïve Bayes that is based on feature independence, to overcome their shortfalls. Preprocessing offers little to no assistance to the classifiers that adjust well to non-linear decision surfaces such as Decision Tree, k-Nearest Neighbor and Support Vector Machine. The k-Nearest Neighbor results exhibit very little variation in isolation accuracy between the different test sets. The Support Vector Machine on the other hand exhibits large variations suggesting that it over-fits the data, and therefore fails to generalise well to unseen data.

5.4.3 Sensitivity Study on Configuration Parameters

An investigation was performed with the aim of analysing the effect of the training configuration parameters on the isolation accuracy for the various classification algorithms. The classifiers were trained using the default configuration parameters, but can be fine tuned to optimise the accuracy of the FDI architecture. An experiment was performed by randomly initialising the configuration parameters from a realistic range on either side of the default values. The investigation gives an indication of the possible increase in classification accuracy that is achievable should configuration parameters be optimised. It also indicates the sensitivity of the various classification algorithms to their configuration parameter values. The configuration parameters and the range over which they were varied is shown in Appendix C.5 with the results shown in figure 5.9.

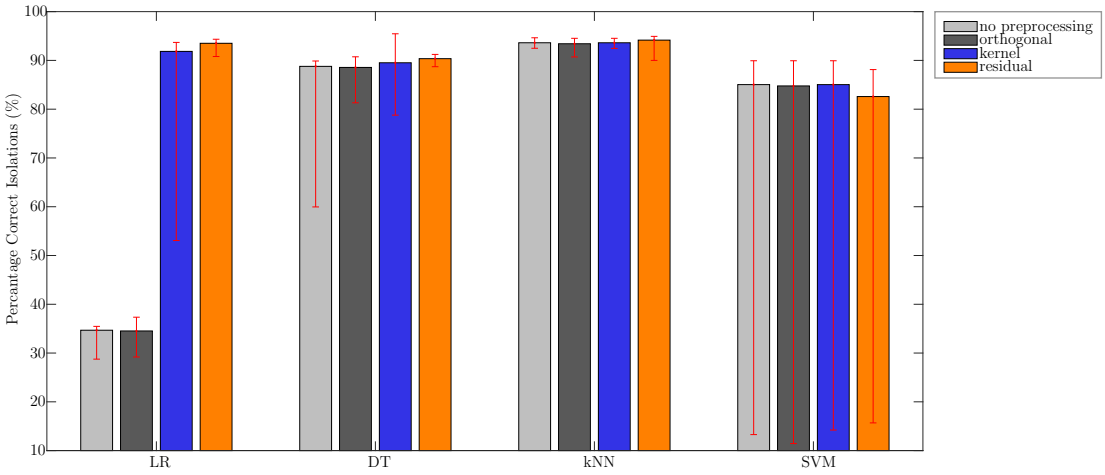


Figure 5.9: The results of a sensitivity study conducted for the various classification algorithms and preprocessing techniques. The minimum and maximum accuracies achieved are indicated by the red whiskers.

The Gaussian Naïve Bayes classifier has no configurable parameters, and is therefore excluded from this investigation. The results show that the k-Nearest Neighbors algorithm is the least sensitive to a change in training parameters, while the Support Vector Machine architecture is the most sensitive. The RBF kernel configuration parameter γ is sensitive as reflected in the large variation in SVM accuracies shown in the figure 5.9. The Decision Tree and Support Vector Machine show the largest increases in accuracy if the training parameters are to be optimised.

5.4.4 Evaluation of Incorrect Classifications

The locations of the false alarms, missed opportunities, and incorrect isolations for the top performing architectures is summarised in table 5.4. Ideally, the locations of false alarms should be restricted to near the decision boundary and there should be no false alarms deep in the no-fault region. False alarms will degrade the system reliability. Similarly, missed opportunities and incorrect isolations beyond the boundary of an extreme outlier are problematic as they will affect the accuracy of the height estimate produced by the

sensor fusion stage. The table below summarises the number of false alarms, missed opportunities, and incorrect isolations of faults when analysing the respective approaches using the test datasets consisting of 283 651 individual data points.

Rank	Approach	False Alarms	Missed Opportunities	Incorrect Isolation
1	kNN	2 568 (0.91 %)	11 890 (4.19 %)	2 221 (0.78 %)
2	GNB	118 (0.04 %)	17 622 (6.21 %)	43 (0.02 %)
3	LR	233 (0.08 %)	17 933 (6.32 %)	445 (0.16 %)
4	DT	15 709 (5.54 %)	9 768 (3.44 %)	3 533 (1.25 %)
5	SVM	138 (0.05 %)	11 850 (4.18 %)	41 799 (14.74 %)

Table 5.5: Analysis of the incorrect assignments performed by the various single time instant approaches.

The obvious observation from the results in table 5.5 is the unacceptably high number of false alarms for the k-Nearest Neighbors and Decision Tree approaches. This will be investigated further in the analyses that follow. The second observation is that the number of missed opportunities by far outweighs the false alarms and incorrect isolations for the top performing single time instant architectures. This is to be expected due to the manner in which a fault was labeled. All data points beyond the time of fault occurrence are labeled as faulty regardless of whether they meet the threshold of being a fault or not. For example fault profiles such as a jamming fault will not immediately yield measurements in the fault region, and an increased noise fault will sporadically produce measurements within the no-fault region. When considering data points as being constructed from singular measurements sampled at the same instant in time, some faulty data points are actually located within the no-fault region and will therefore be incorrectly classified. This is an acceptable error for the single time instant approach as no bias is shown towards previous predictions by considering measurements from each time instant as unique.

5.4.4.1 k-Nearest Neighbors

The analysis done on the location of incorrect classifications for k-Nearest Neighbors is discussed below. Figures 5.10 to 5.13 give a visual overview of the locations of all incorrect classifications. The different scatter plots are established for individual sensors where the measured relative altitude of the aircraft is plotted as a function of the true relative altitude for the respective sensors. The false alarms are indicated in red, missed opportunities in green, and incorrect isolations in orange. The blue line plots the theoretical decision boundary that coincides with the limit of a sensor's range of accuracy. The dashed blue line as labeled in figure 5.11 shows the decision boundary for what will be considered an extreme outlier.

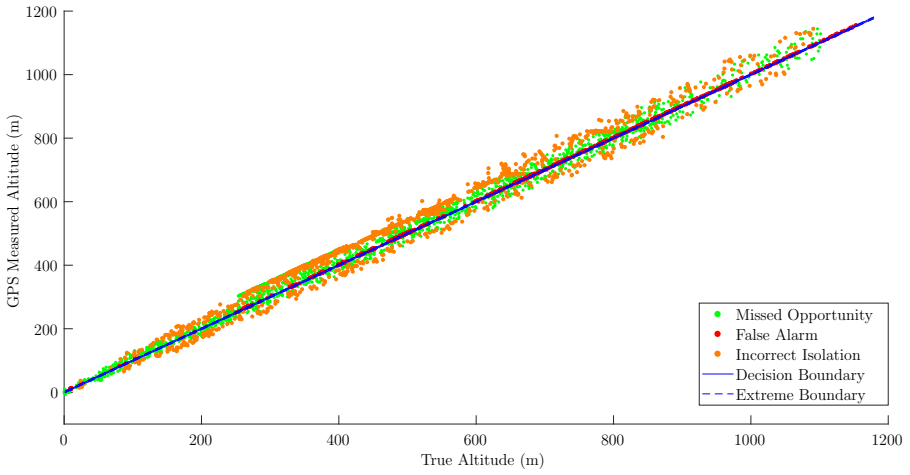


Figure 5.10: The location of the incorrect classifications for the GPS sensor using k-NN.

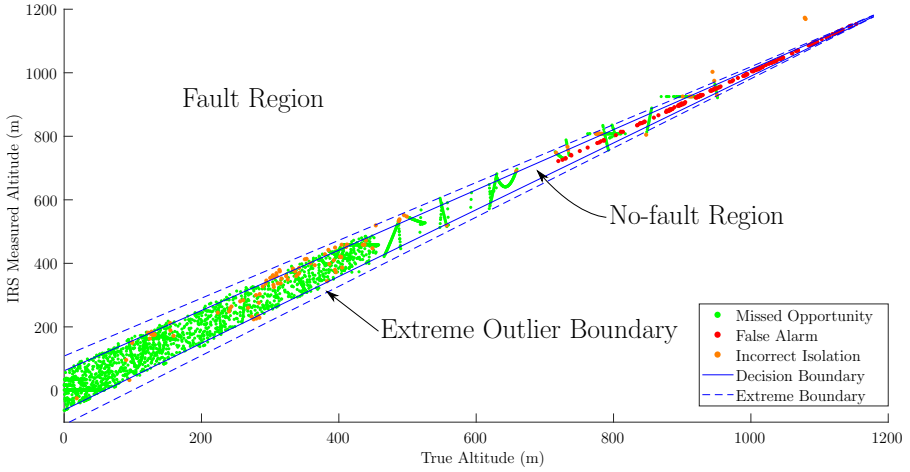


Figure 5.11: The location of the incorrect classifications for the IRS sensor using k-NN.

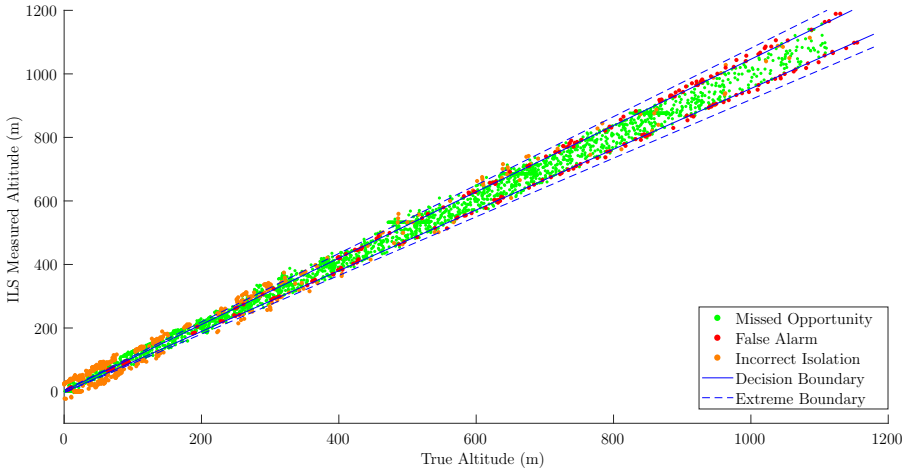


Figure 5.12: The location of the incorrect classifications for the ILS sensor using k-NN.

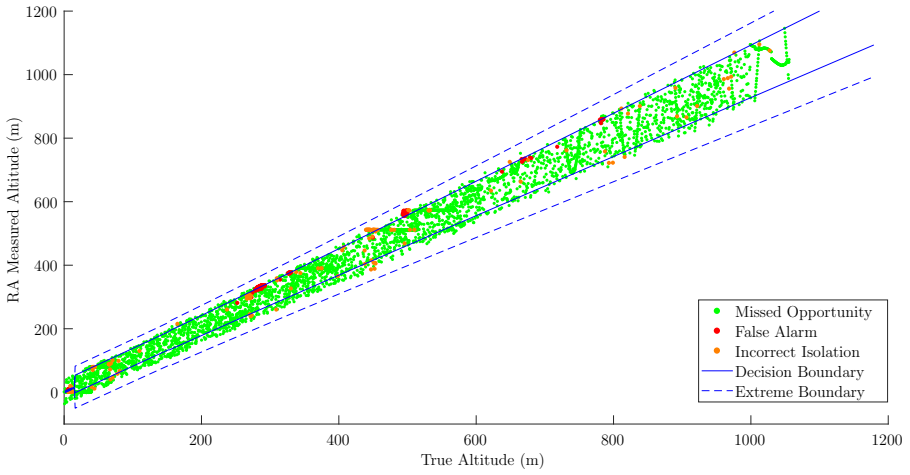


Figure 5.13: The location of the incorrect classifications for the RA using k-NN.

Although the figures above are not completely clear, there are some obvious observations that can be made with a more detailed analysis discussed thereafter. The first is that the location of missed opportunities is unfavorable as many are situated beyond the decision boundary of what is considered to be an extreme outlier. The majority of the missed opportunities are within the no-fault region, which is an acceptable error. The location of the missed opportunities and incorrect isolations are particularly concerning for the GPS and IRS sensors. The location of the false alarms and incorrect isolations are shown relative to the decision boundary for the various sensors in figure 5.14 to 5.17. The measurement error is plotted as a function of the true altitude with the decision boundary given by the blue line.

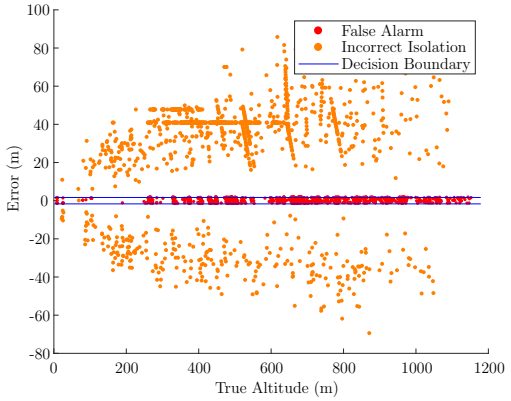


Figure 5.14: The location of the false alarms and incorrect isolations for the GPS sensor.

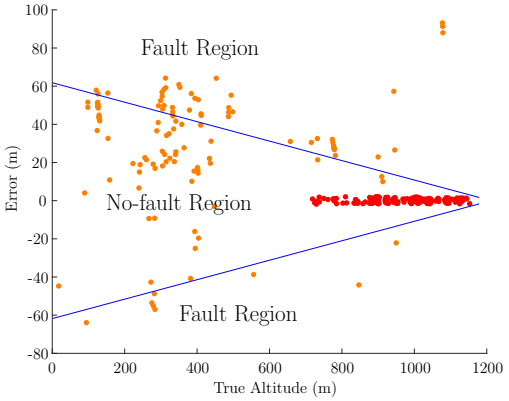


Figure 5.15: The location of the false alarms and incorrect isolations for the IRS sensor.

There are two observations from the results in figure 5.14 to 5.17. Firstly, there are a large number of false alarms deep within the no-fault region. This is prevalent across all sensors where the decision boundaries are narrow. The number of training examples was increased to better cover the parameter space, but had an insignificant effect on the results. It was concluded that the k-NN approach is susceptible to false alarms in areas where there exists a high density of fault and no-fault data points that are located in close

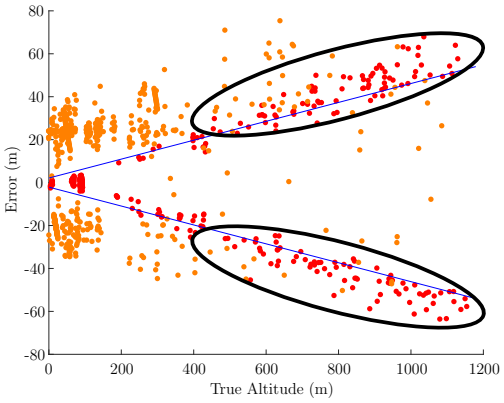


Figure 5.16: The location of the false alarms and incorrect isolations for the ILS sensor.

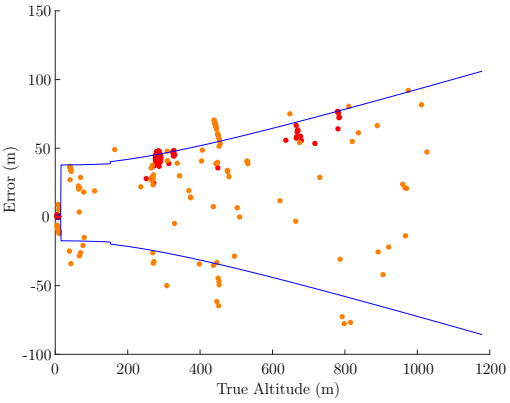


Figure 5.17: The location of the false alarms and incorrect isolations for the radio altimeter.

proximity such as near the decision boundary. The second observation is the presence of false alarms in the fault region as illustrated by the black circles in figure 5.16. The ILS measurement error is modeled by a zero mean Gaussian white noise signal, and therefore it is inevitable that there will be a few no-fault data points beyond the decision boundary of 2.695σ that is only inclusive of 99.65 % simulated data points. These false alarms affirm that the k-NN approach fits a tight decision boundary around the data. Without looking at a window of measurements it is not possible for the classifier to know that these data points are actually no-fault data points. The number and location of the incorrect isolations is also unfavorable as these should be restricted to within a close proximity of the decision boundary and should be kept to a bare minimum. The k-Nearest Neighbors architecture therefore does not meet the requirements of being a reliable fault detection and isolation system.

5.4.4.2 Gaussian Naïve Bayes

The location of the incorrect classifications for the Gaussian Naïve Bayes approach is shown in figure 5.18 to 5.21.

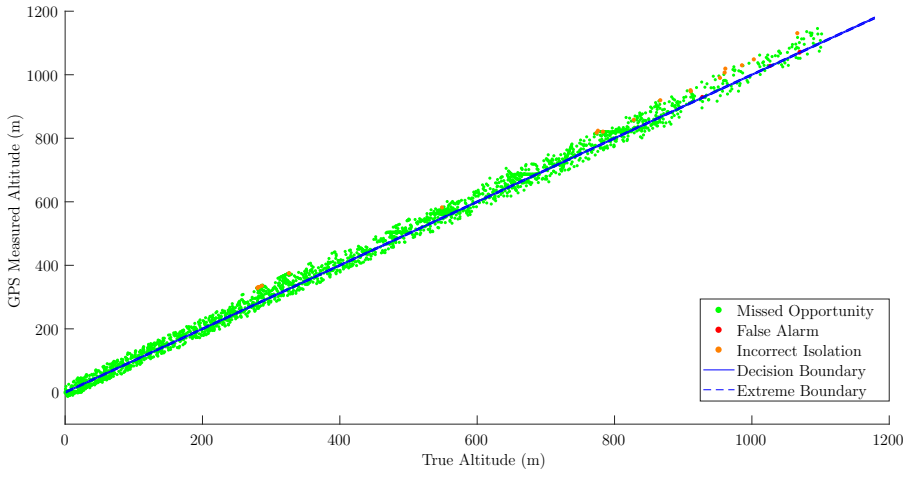


Figure 5.18: The location of the incorrect classifications for the GPS sensor using GNB.

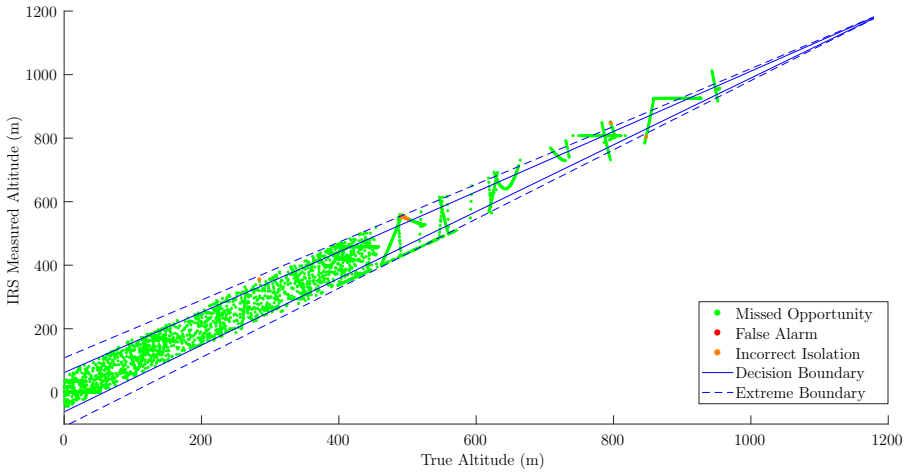


Figure 5.19: The location of the incorrect classifications for the IRS sensor using GNB.

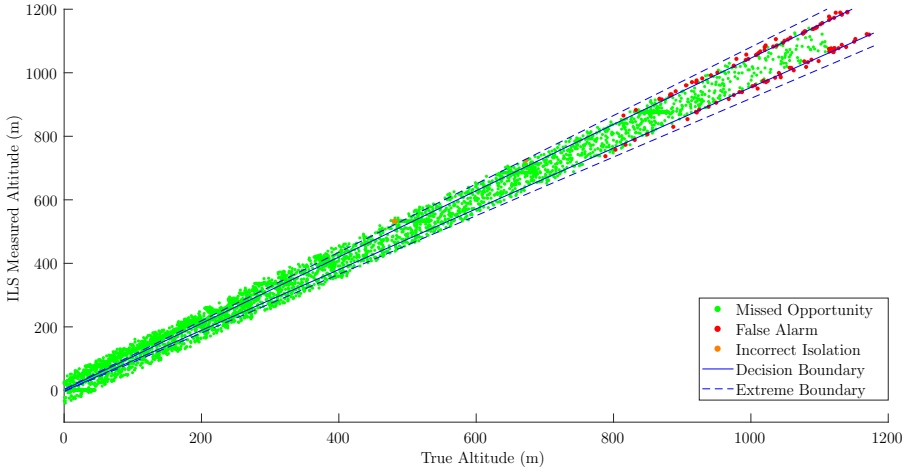


Figure 5.20: The location of the incorrect classifications for the ILS sensor using GNB.

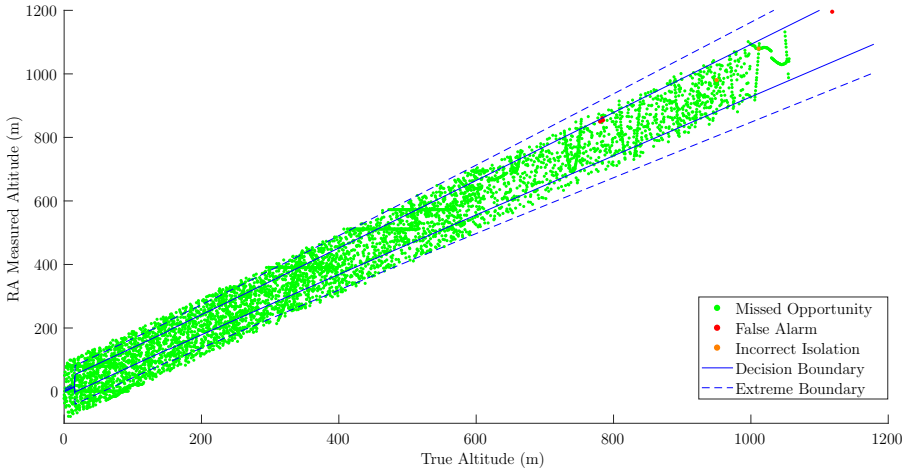


Figure 5.21: The location of the incorrect classifications for the RA using GNB.

The Gaussian Naïve Bayes architecture also yielded numerous missed opportunities deep within the fault region. It is evident that the Gaussian Naïve Bayes FDI architecture struggles to adapt to changes in sensor accuracy. It fits a decision boundary that is aligned with the worst case sensor accuracy as illustrated by the location of the missed opportunities in figure 5.20 and figure 5.21, while being a poor fit of the GPS sensor measurement data in figure 5.18. An analysis of the location of the false alarms revealed that they were all attributed to the ILS and radio altimeter sensors. All of the false alarms were located in a close proximity to the decision boundary with a few of the false alarms being inside the fault region due to the ILS sensor model as shown in figure 5.22 and figure 5.23. The reader is reminded that the ILS sensor model uses random Gaussian white noise to model realistic measurement errors as discussed in the k-NN analysis.

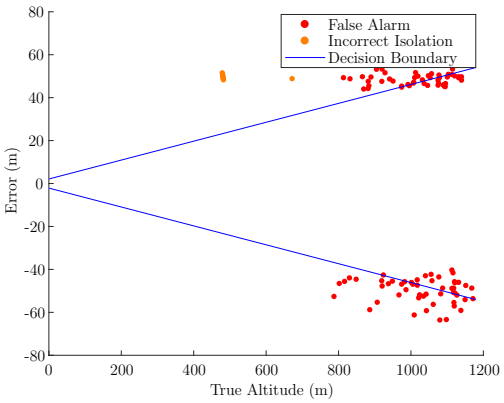


Figure 5.22: The location of the false alarms and incorrect isolations for the ILS sensor.

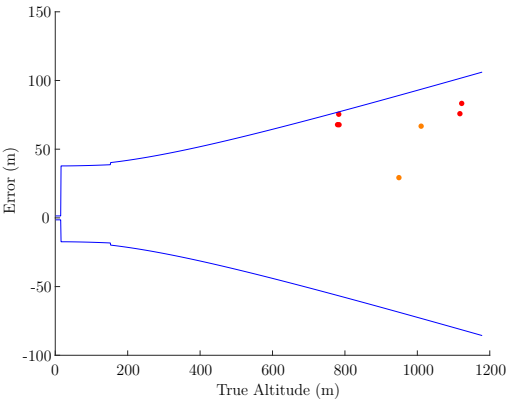


Figure 5.23: The location of the false alarms and incorrect isolations for the RA.

5.4.4.3 Logistic Regression

The location of the incorrect classifications for the Logistic Regression approach is shown in figure 5.24 to 5.27.

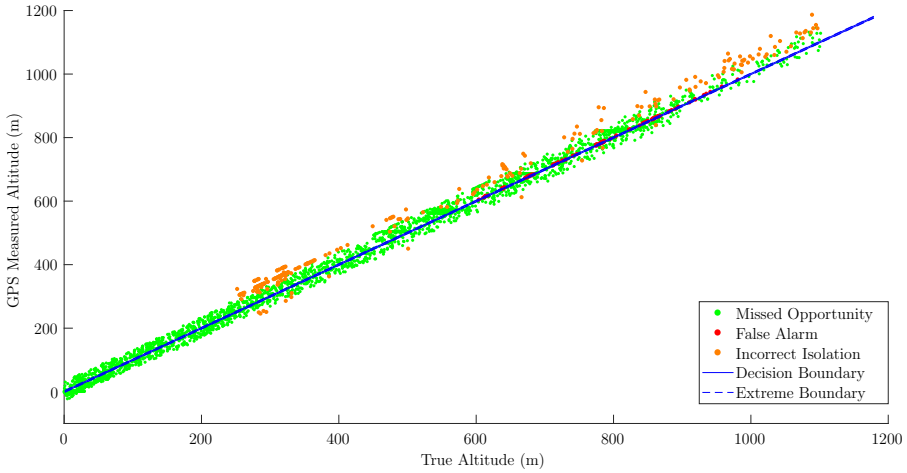


Figure 5.24: The location of the incorrect classifications for the GPS sensor using LR.

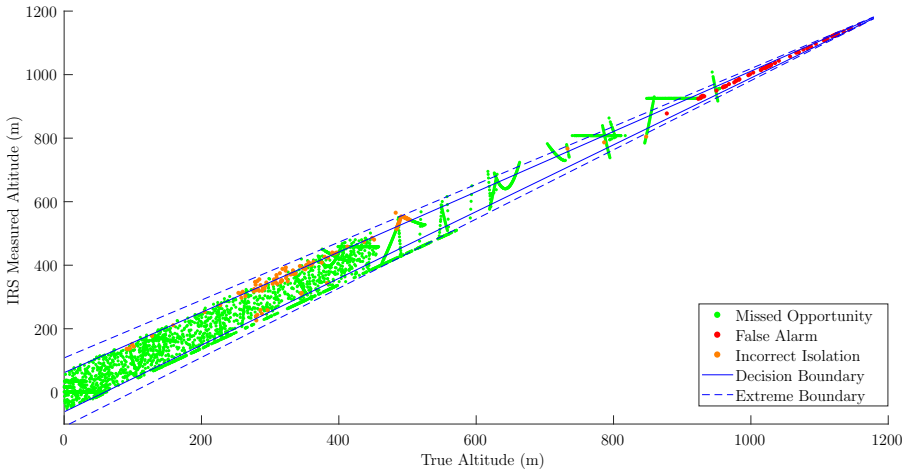


Figure 5.25: The location of the incorrect classifications for the IRS sensor using LR.

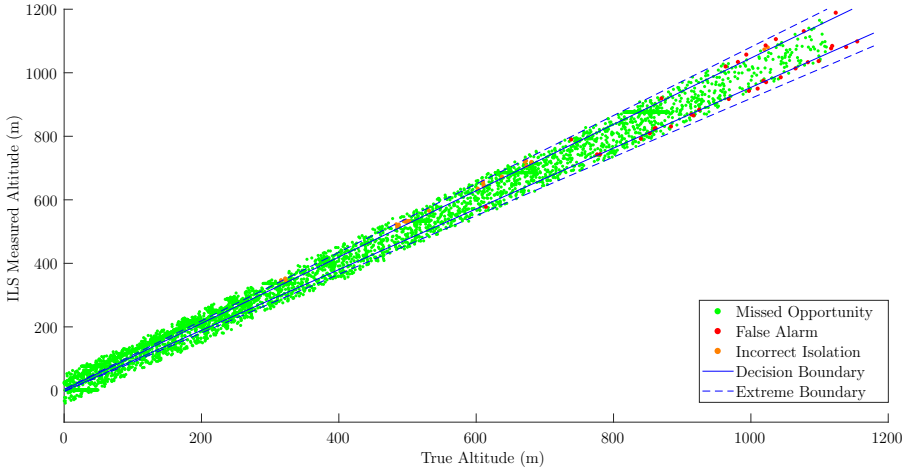


Figure 5.26: The location of the incorrect classifications for the ILS sensor using LR.

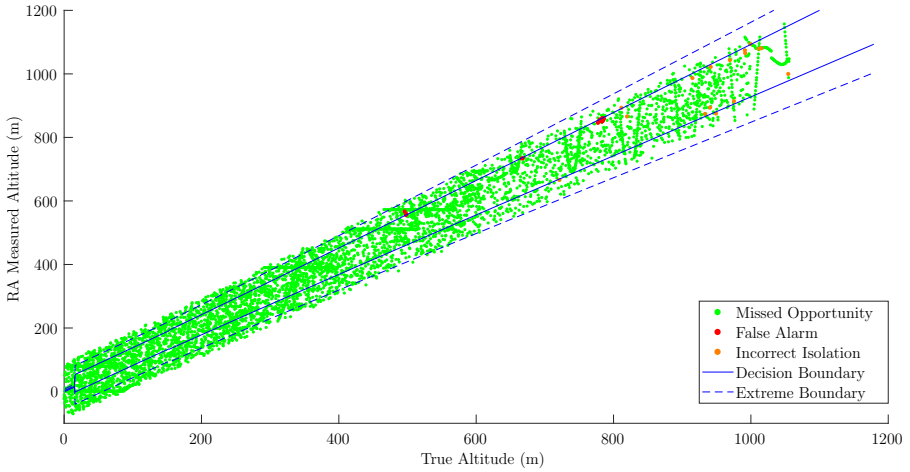


Figure 5.27: The location of the incorrect classifications for the RA using LR.

The Logistic Regression results are similar to the Gaussian Naïve Bayes results as the architecture struggles to adapt to changes in sensor accuracy as indicated by the distribution of the missed opportunities. The locations of the missed opportunities are unfavourable as they are beyond the boundary from an extreme outlier. False alarms and incorrect isolations occurred across all the sensors with an in-depth analysis of their locations illustrated in figure 5.28 to figure 5.31. A concern with the Logistic Regression based FDI architecture is the false alarms that occur deep within the no-fault region for the GPS and IRS sensor as illustrated in figure 5.28 and figure 5.29 respectively.

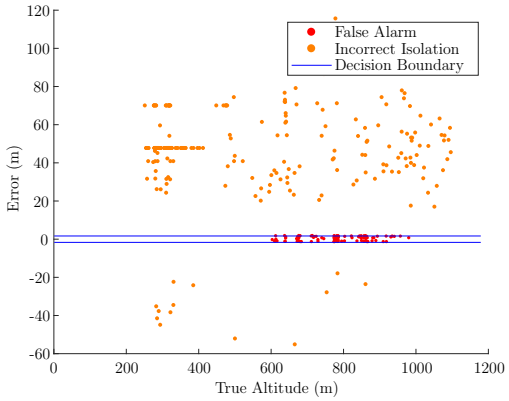


Figure 5.28: The location of the false alarms and incorrect isolations for the GPS sensor.

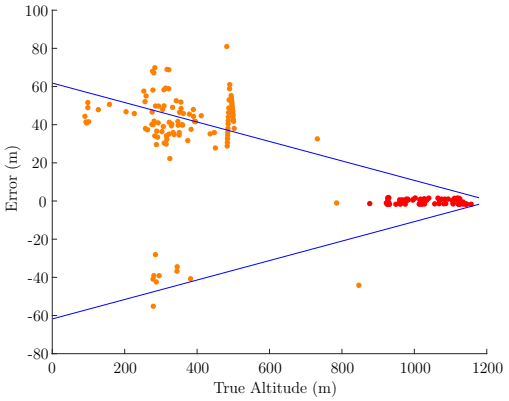


Figure 5.29: The location of the false alarms and incorrect isolations for the IRS sensor.

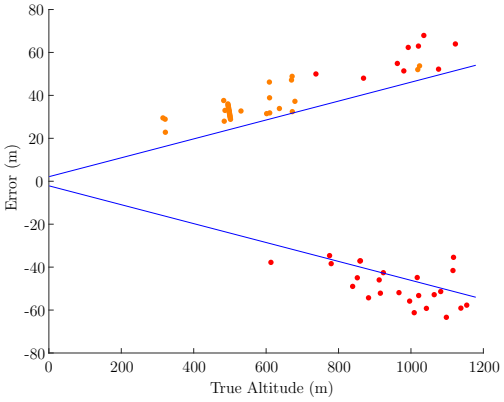


Figure 5.30: The location of the false alarms and incorrect isolations for the ILS sensor.

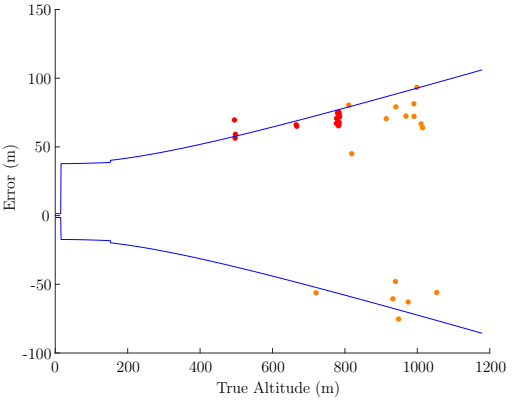


Figure 5.31: The location of the false alarms and incorrect isolations for the RA.

It can be concluded from the location of incorrect classification analyses that none of the single time instant architectures meet the requirement of being a reliable fault detection and isolation architecture. The Gaussian Naïve Bayes architecture is the best performer as it minimises the number of false alarms and incorrect isolations, while restricting their locations to near the decision boundary. However it does not adapt well to the changing

theoretical sensor accuracies, and therefore may result in missed opportunities that will affect the height estimation accuracy of the sensor fusion stage.

5.4.5 Results Validation

The FDI architecture accuracy was validated on actual aircraft data with injected faults as discussed in Chapter 4.3.4. The results validation compares the fault isolation accuracy achieved on the validation dataset illustrated in figure 5.32 with the results from simulated data as illustrated in figure 5.8. Overall the results for actual flight data were similar to the results for simulated flight data with the percentage of correct isolations illustrated in figure 5.32 lying within the minimum and maximum range of accuracy as illustrated in figure 5.8. This suggests that the model used to simulate measurement data is representative of actual sensor measurements and that similar results would be achieved if a more diverse dataset of actual sensor measurement data were available.

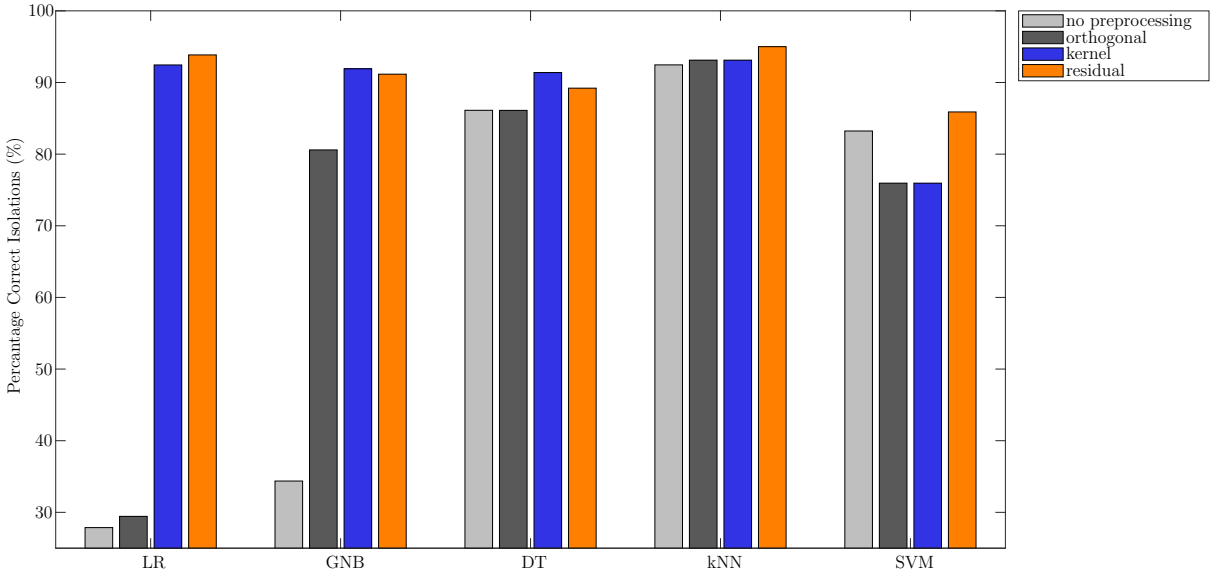


Figure 5.32: The results of FDI accuracy on validation test set consisting of actual flight data.

5.5 FDI Demonstration

The performance of the top three single time instant FDI architectures is demonstrated on example aircraft landing runs with different sensor faults injected as discussed in Chapter 4.6. The results are shown in figures 5.33 to 5.36, where the actual sensor status is indicated in red and the reported status in blue as a function of time.

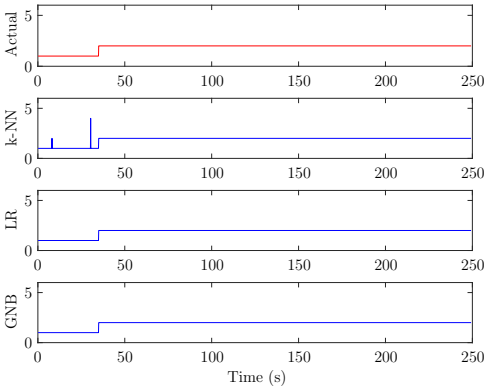


Figure 5.33: Sensor status versus time for a GPS bias fault.

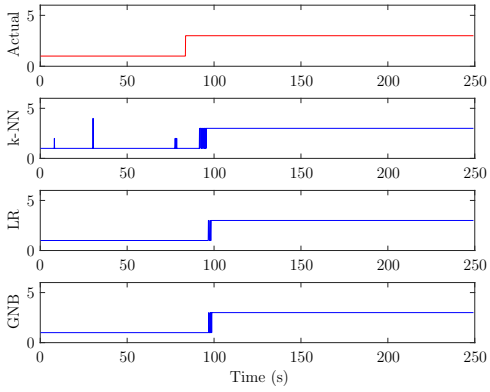


Figure 5.34: Sensor status versus time for a IRS jamming fault.

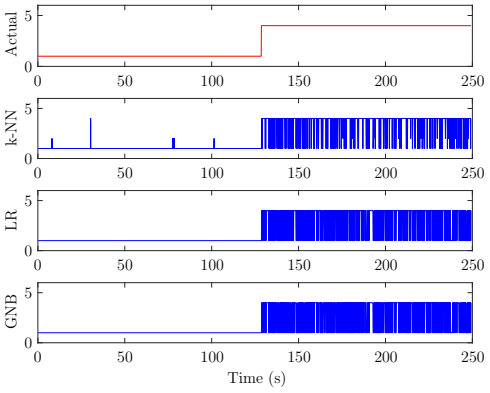


Figure 5.35: Sensor status versus time for a ILS noise fault.

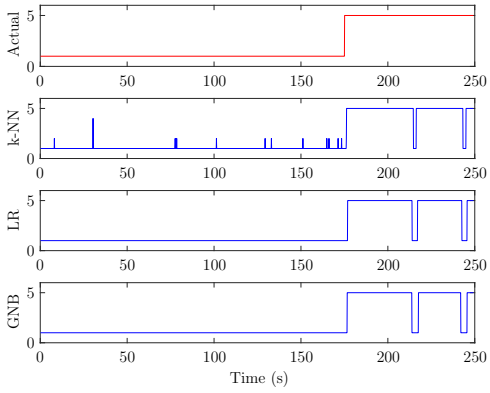


Figure 5.36: Sensor status versus time for a radio altimeter oscillating fault.

The FDI demonstration shows that all architectures correctly identified the sensor status soon after the onset of the respective faults. The shortcomings with the approach of analysing a single instant in time is apparent when looking at the increased noise and oscillation fault in figure 5.35 and figure 5.36 respectively. The FDI architecture could be enhanced by incorporating fault detection memory that would continue to isolate a fault even after first being detected. This suggestion is investigated in the next chapter in an attempt to address the shortcomings with the single time instant approach. Two observations from figure 5.34 warrant further discussion. Firstly, as expected the jamming fault shows a slight delay between the onset of the fault and the fault being detected. A jamming fault is not immediately obvious, but as the aircraft continues to descend so

the error becomes larger with time. Secondly, as the jamming fault measurement error nears the decision boundary, the sensor status becomes unstable, but then stabilises once the measurement is well beyond the decision boundary. The shortcomings of the Nearest Neighbors approach as discussed earlier in this chapter are also confirmed by the many false alarms.

5.6 Conclusion

The single time instant FDI architectures returned isolation accuracies from 85 % (Support Vector Machine) to 94.16 % (Nearest Neighbors). The use of preprocessing greatly aids the performance of the Logistic Regression and Gaussian Naïve Bayes, but is of little advantage to the remaining architectures. A random search of the configuration parameters revealed that the default values will suffice for architecture comparison as these yield accuracies close to the maximum accuracy achieved from the random search. An evaluation of the location of the false alarms, missed opportunities, and incorrect isolations revealed some shortcomings with all the single time instant approaches. The Gaussian Naïve Bayes architecture that makes use of residual preprocessing is the best candidate, although it struggles to adapt to the changes in sensor accuracy. The single time instant approach to fault detection is robust and adaptable in the sense that it re-evaluates measurements at each time instant without any prior knowledge or prejudice towards a previous sensor status. The results were validated using actual data and the fault detection and isolation performance was demonstrated on different fault profiles by plotting the actual versus reported sensor status as a function of time.

Chapter 6

Consecutive Time Instant Fault Diagnosis

This chapter presents the design and evaluation of a fault detection and isolation architecture for a commercial aircraft that uses sensor measurements from a window of consecutive time instants. The system investigates the use of traditional model-based and novel data-driven techniques for fault detection and isolation purposes.

The chapter starts off by presenting the individual architectures for the different consecutive time instant fault diagnostic approaches that were considered. The computational complexity of each approach is then investigated and compared. The fault detection and isolation performance is evaluated separately on numerous datasets. The results are then validated on actual aircraft data. Lastly, the top performing FDI architectures are demonstrated on simulated landing runs with faults injected on certain sensor measurements to show how the predicted and actual sensor status changes as a function of time.

6.1 Conceptual Design and Architecture

The architecture for the consecutive time instant fault detection and isolation techniques will vary for each approach. The detailed design of the individual approaches will be considered separately. A summary of the approaches considered is shown in table 6.1.

Approach Name	**Approach Type**
Robust Kalman filter	Robust statistics and optimal estimation
Bank of Kalman filters	Probabilistic combination of optimal estimators
Model Consensus (variant of RANSAC)	Evaluation of a fixed number of models
Dynamic PCA	Monitoring of statistical measures
Gaussian Naïve Bayes	Single time instant approach with fault detection history

Table 6.1: Summary of the consecutive time instant approaches.

6.1.1 Robust Kalman Filter

The overview of the conceptual design and architecture for the Robust Kalman filter is illustrated in figure 6.1 and will be discussed below.

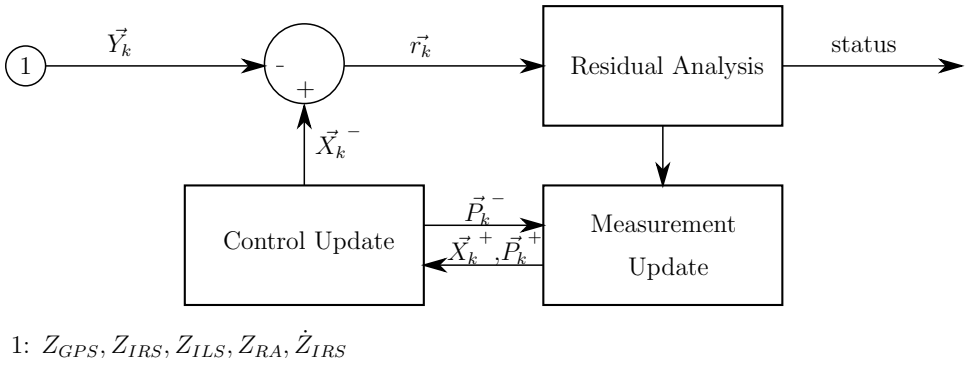


Figure 6.1: The architecture for a Robust Kalman filter fault detector and isolator.

The design of the KF model which constitutes the control and measurement update steps in the architecture above is briefly explained. The Kalman filter model estimates the relative altitude of the aircraft and uses it as the reference against which the sensor measurements are compared. A descending aircraft is modeled by a linear kinematic system. It has two states, namely relative altitude and climb rate. The state space model will form the foundation of the Kalman filter design. The dynamics applicable to predicting the change in aircraft height are approximated as linear over a sampling period. The linear discrete state space model has two states, namely altitude z_k and climb rate $\dot{z}_k$.

$$\vec{X_k} = \begin{bmatrix} z_k \\ \dot{z}_k \end{bmatrix} \tag{6.1.1}$$

The model parameters that govern the linear model $\vec{X_k} = F\vec{X}_{k-1} + G\vec{U}_{k-1}$ are given by:

$$F = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \qquad G = 0 \tag{6.1.2}$$

The sensor measurements are directly observable, whilst the predicted sensor measurements are given by $H\vec{X_k}$, where:

$$\vec{Y_k} = \begin{bmatrix} Z_{GPS}[k] \\ Z_{IRS}[k] \\ Z_{ILS}[k] \\ Z_{RA}[k] \\ \dot{Z}_{IRS}[k] \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.1.3}$$

The Kalman filter requires prior knowledge of the process and sensor noise variances. The process noise is a tuning parameter, whilst the sensor noise matrix is updated for each sampling instant to reflect the expected sensor accuracies that change as a function of

relative altitude. This makes the Robust Kalman filter adaptive as it adjusts the noise covariance matrix.

$$Q_k = E\left\{W_k W_k^T\right\} = \begin{bmatrix} \sigma_w^2 & 0 \\ 0 & \sigma_w^2 \end{bmatrix} \tag{6.1.4}$$

$$R_k = E\left\{V_k V_k^T\right\} = \begin{bmatrix} k_1 \cdot {\Delta_{Z_{gps}}}^2 & 0 & 0 & 0 & 0 \\ 0 & k_2 \cdot {\Delta_{Z_{irs}}}^2 & 0 & 0 & 0 \\ 0 & 0 & k_3 \cdot {\Delta_{Z_{ils}}}^2 & 0 & 0 \\ 0 & 0 & 0 & k_4 \cdot {\Delta_{Z_{ra}}}^2 & 0 \\ 0 & 0 & 0 & 0 & \Delta^2_{\dot{Z}_{irs}} \end{bmatrix} \tag{6.1.5}$$

where k_i is a tuning parameter, and Δ_i the theoretical sensor accuracy for sensor i.

Fault isolation is performed during the residual analysis that compares the measurement residual against the theoretical limits of sensor accuracy at each iteration. When $Y_k^{(i)} - H\vec{X}_k \geq k_i \cdot \Delta_i$, then $Y_k^{(i)} - H\vec{X}_k$ is set to zero and the estimated state update is therefore isolated from the fault. The factor k_i is a tuning parameter that is used to configure the tightness of the decision boundary fitted to the data when identifying a fault. This constant relaxes the decision boundaries to account for the inaccuracies associated with the state estimate and modelling process. A conservative decision threshold is used that only updates the sensor status to reflect a failure when more than 50 % of measurements from within the last 1 s have been flagged as faulty.

6.1.2 Bank of Kalman Filters

The conceptual design and architecture for the Bank of Kalman filters is illustrated in figure 6.2.

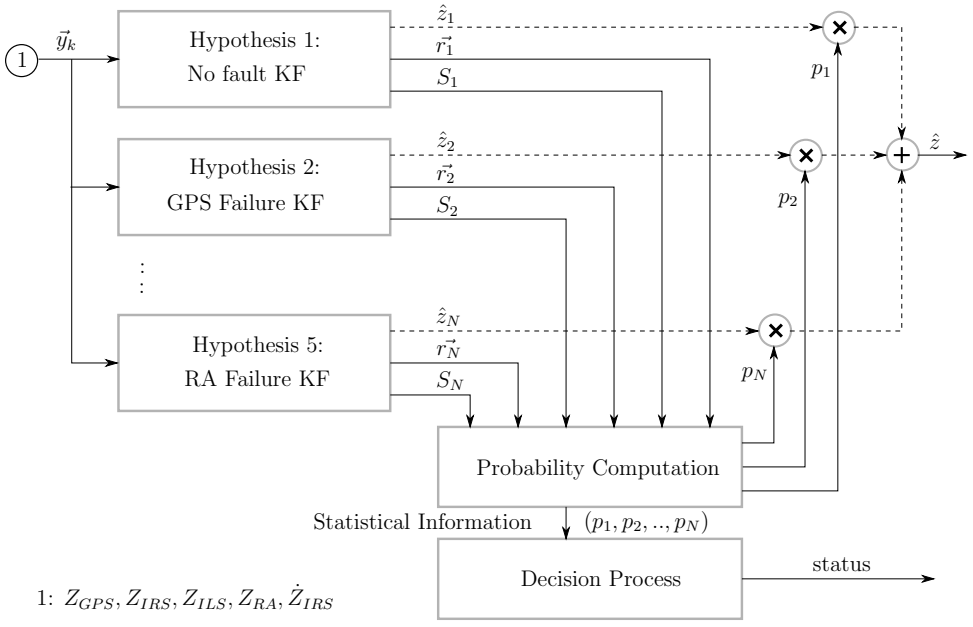


Figure 6.2: The architecture for a Bank of Kalman filters approach.

The architecture establishes five separate Kalman filters. The first KF models the no-fault hypothesis and therefore makes use of all sensors. The remaining hypotheses each adjust the Kalman filter design to omit the sensor that is hypothesised as failing. Equations 6.1.1 to 6.1.5 are adjusted for each individual Kalman filter accordingly. Similar to the Robust KF, the individual Kalman filters are adaptive in the sense that they adjust the noise covariance matrix according to the expected sensor accuracies. The estimated relative altitude and measurement residual covariances are used to calculate the probability associated with each hypothesis as discussed in Chapter 3.3.3.1. The reader is reminded that the probability threshold (P_{thresh}) is used to identify the active hypothesis. The sensor status is only updated to reflect a fault when $p_i > P_{\text{thresh}}$ for longer than 2 seconds. A probability threshold of $P_{\text{thresh}} = 90\,\%$ is used to identify an active hypothesis. The estimated relative altitude, $\hat{z}$, is calculated by weighting each estimate according to the probability associated with it.

6.1.3 Model Consensus

Model Consensus is a proposed fault detection and isolation method that is based on RAndom SAmple Consensus (RANSAC). The proposed method is an online solution that incorporates multiple time steps by applying a sliding window of width L to the incoming data stream of m sensor measurements. The method is based on the assumption that the aircraft dynamics can be approximated as linear over such a short period of time. The proposed model deviates from RANSAC in one fundamental area. The random sampling strategy is replaced by a fixed number of hypotheses between the first and last sampling instants in the sliding window as depicted in figure 6.3. Predictable aircraft dynamics and the limited batch size are the motivation for evaluating a fixed number of hypotheses. Establishing linear models between the first and last sampling instants in a window will produce the hypothesis with the best support. A total of $m^2 + 1$ hypotheses are established, which includes the exhaustive linear possibilities between each sensor measurement at t_1 and t_L as well as the model parameters from the previous window, Θ_{prior}.

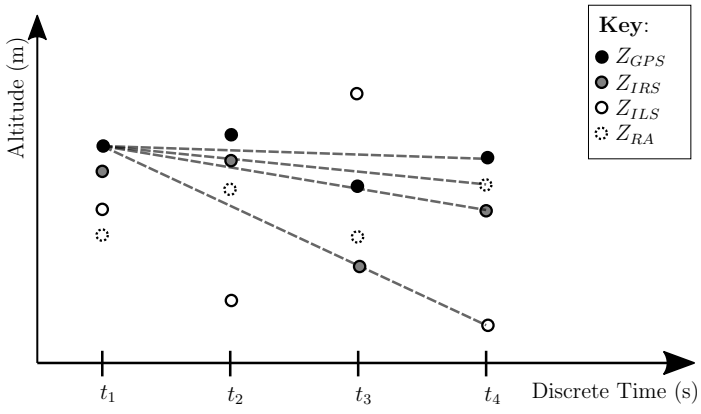


Figure 6.3: Possible hypotheses when linear models are fitted between a sensor measurement at sample period t_1 and measurements at t_4 for a sliding window of width $L = 4$. This is repeated for each sensor at time period t_1.

The model parameters for respective hypotheses are stored in a buffer Θ. Each model is evaluated to determine the hypothesis that maximises the support over the entire dataset by examining the number of data points that are considered to be inlier. The inlier dataset ($\mathcal{I}$) will therefore contain the data points for the model with the most support. Model Consensus deviates from RANSAC's verification stage in that a separate Δ threshold is used for each sensor to accommodate the different theoretical accuracies.

Algorithm 3 Pseudocode for Model Consensus algorithm.

Input:

$\mathcal{Z}$	dataset of observations
Θ_{prior}	model parameters from previous window
Δ	sensor accuracy threshold
m	number of sensors
ϵ	probability threshold for fault detection

Output:

Θ_{post}	model representing inliers
status	sensor status

function MODEL CONSENSUS($\mathcal{Z}$,Θ_{prior},Δ,ϵ_i,m)
  $k \leftarrow m^2 + 1$
  $\Theta \leftarrow$ Possible Models($\mathcal{Z}$,Θ_{prior},m)
  for j = 1:k **do**
    $\mathcal{I}_j \leftarrow$ Model Evaluation($\mathcal{Z}, \Theta_j, \Delta$)
    if $\mathcal{I}_j > \mathcal{I}$ **then**
      $\mathcal{I} \leftarrow \mathcal{I}_j$
    end if
  end for
  $\Theta_{post} \leftarrow$ Fit Optimal Model($\mathcal{I}$)
  status $\leftarrow$ Status Update(Θ_{post},ϵ,Δ)
return(Θ_{post},status)

Once all hypotheses have been evaluated, the inliers are used to identify the model of best fit (Θ_{post}). Weighted least squares regression was implemented for model refinement. It is an extension of ordinary least squares that deems all inlier samples to be heteroscedastic[1] and therefore individual sensor accuracies are accounted for during model refinement. The optimal model parameters are those that minimise the cost function given as the sum of square residuals between the observed measurements and the model predictions.

The optimal model Θ_{post} is used to re-evaluate individual sensor measurements and identify outliers. The sensor status is updated by evaluating the percentage of data points associated with a sensor that are deemed to be inliers. When the probability associated with an individual sensor drops below a predefined threshold ϵ then the sensor status is updated to reflect a failure in that specific sensor.

[1] noise variance differs across observations

6.1.4 Dynamic PCA

The overview of the conceptual design and architecture for the Dynamic PCA is illustrated in figure 6.4 and discussed thereafter. Dynamic PCA is used as a fault detection technique.

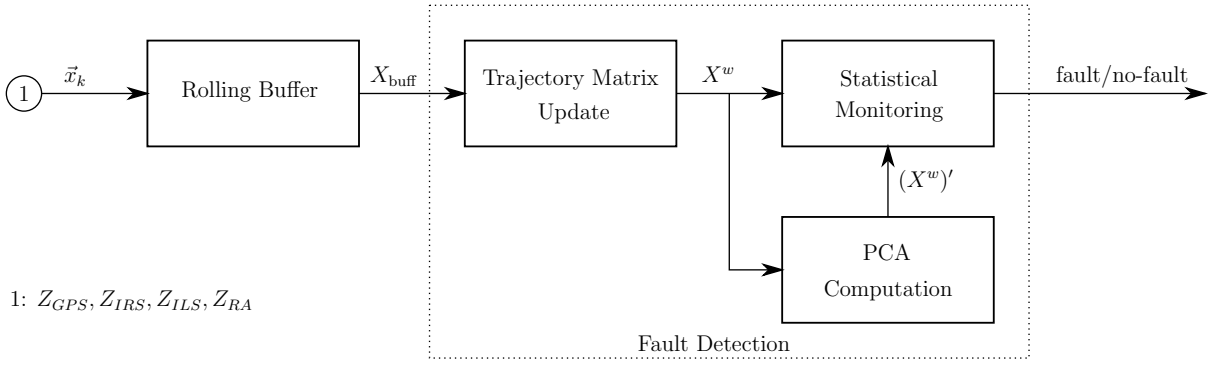


Figure 6.4: The architecture for a Dynamic PCA fault detector.

The rolling buffer stores the n most recent sensor measurements as $X_{\text{buff}} = [\vec{x}_k, ..., \vec{x}_{k-n}]$ and presents the buffer to the fault detection architecture. The trajectory matrix (X^w) is updated using equation 3.8.14, that divides a sliding window of the n most recent measurements into time lagged vectors of trajectory information as illustrated in figure 6.5. When a fault occurs the individual vector that incorporates that measurement will differ from the others.

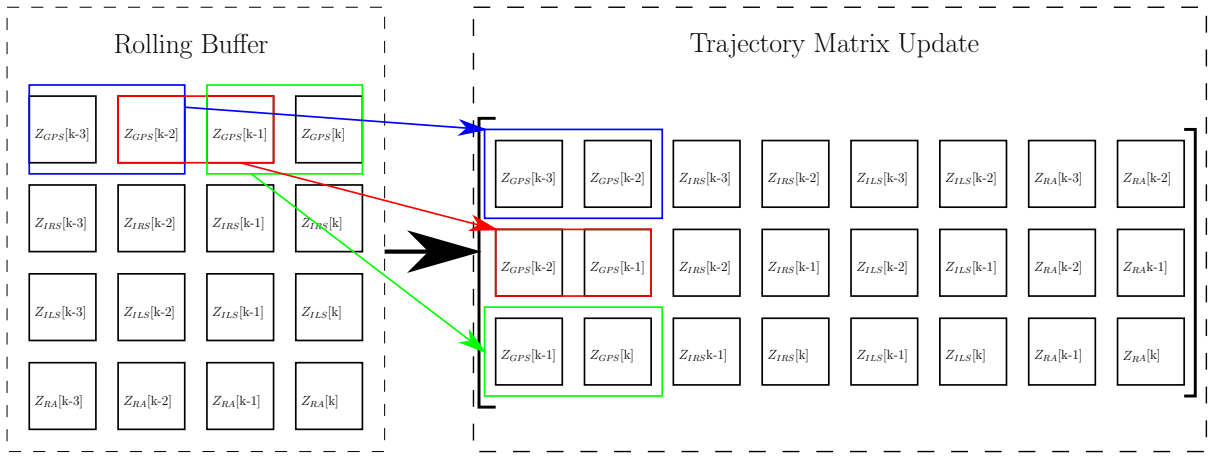


Figure 6.5: Formation of the trajectory matrix for a time lag shift of order $w = 2$ for a sliding window of 4 consecutive data points.

A model of the trajectory information is established using the lower-dimensional representation of the trajectory matrix. This is achieved by first performing an eigenvalue decomposition to identify the transformation matrix (V). The design decision was taken to retain the first principal component or eigenvector corresponding to the largest eigenvalue and let it represent the "trajectory information" or model of the data, where $V_k = [v_1^T] = V_1$. The lower-dimensional representation of the dataset is established in the

transformed data space through $Y_k^w = V_k^T X^w$ before reconstructing the trajectory matrix, $(X^w)'$ in the original data space.

$$(X^w)' = V_k Y_k^w \tag{6.1.6}$$

where V_k is the transformation matrix comprised from the k largest eigenvectors

A comparison between the trajectory matrix (X^w) and its reconstructed lower-dimensional representation $(X^w)'$ that functions as a model provides information that is used for fault detection purposes. The Square Projection Error (SPE) is a measure that is monitored online and used to identify when a fault occurs.

6.1.5 Gaussian Naïve Bayes

One of the top performing single time instant approaches, Gaussian Naïve Bayes, is included as a consecutive time instant approach by giving it the ability to remember its sensor status. Gaussian Naïve Bayes was chosen as it minimised the number of false alarms and incorrect isolations. The conservative decision threshold was used where more than 50 % of the measurements from within the last two seconds had to register as a fault before the sensor status was updated. The decision was also made that once a fault is detected, then it "latches" and the fault status remains active until after the aircraft lands.

6.2 Computational Complexity Study

The computational complexity of implementing the techniques discussed above is derived and then compared. The number of basic iterations required to process a single sampling instant is also considered. The computational complexity for a Kalman filter is given by $\mathcal{O}(9d^{2.376}+10d^2+5d) \approx \mathcal{O}(d^{2.376})$ [55], where d is the dimensionality of the measurement vector. A similar literature search was done for the computational complexity of RANSAC (Model Consensus). This is at least the square order of the number of data points in the observation dataset[56]. This equates to $\mathcal{O}([mL]^2)$, where m is the number of sensors and L the width of the window. The computational complexity for Principal Component Analysis (PCA) consists of a covariance matrix computation $\mathcal{O}(n_v d^2)$ and a eigenvalue decomposition given by $\mathcal{O}(n_v^3)$ [57], where n_v is the number of vectors that make up the covariance matrix. The combined computational complexity is therefore given by $\mathcal{O}(n_v d^2 + n_v^3)$. The results are summarised in the table 6.2.

Approach Name	**Basic Iterations**	**Complexity**
Robust Kalman filter	20	$\mathcal{O}(d^{2.376})$
Bank of Kalman filters	75	k·$\mathcal{O}(d^{2.376})$
Model Consensus	635	$\mathcal{O}([mL]^2)$
Dynamic PCA	20	$\mathcal{O}(n_v d^2 + n_v^3)$

Table 6.2: Summary of the computational complexity for the consecutive time instant approaches.

The matrix operations used in the consecutive time instant approaches adds to the computational complexity. The most efficient approach is the Robust Kalman filter with the fewest number of basic iterations and lowest computational complexity. The Model Consensus approach is the most computationally intensive with numerous inner loops as it repetitively searches for the best model.

6.3 Fault Detection and Isolation Evaluation

The consecutive time instant fault detection and isolation architectures were evaluated on five different test sets as discussed in Chapter 4.3.3. These architectures are evaluated in terms of accuracy for both fault detection and isolation, sensitivity of the configuration parameters, and the location of incorrect classifications.

6.3.1 Fault Detection and Isolation Accuracy

The fault detection and isolation accuracy for the various consecutive time instants approaches was examined across numerous datasets. The results given in figure 6.6 were determined using the default configuration parameters.

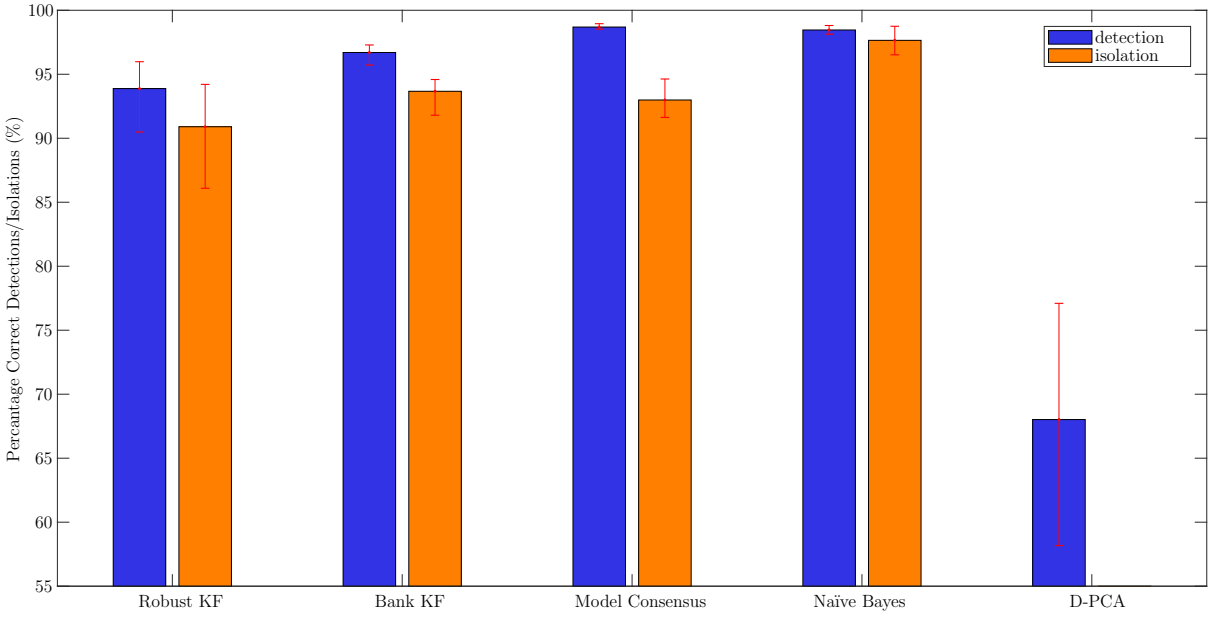


Figure 6.6: The detection and isolation accuracy for the various consecutive time instants based FDI architectures. The minimum and maximum percentages are indicated in red with the mean accuracy shown by individual bars.

The top performing detection architecture was Model Consensus that achieved a detection accuracy of 98.7 %. It was also the most consistent detection architecture exhibiting minimal variation in its results. The worst performing fault detection architecture was Dynamic PCA that achieved a 68.02 % accuracy. PCA is not robust to the effect of outliers that will distort the principal components that describe the underlying structure of the data, resulting in the poor performance. The fault isolation accuracy results are summarised in table 6.3.

Rank	Approach Name	Accuracy
1	Gaussian Naïve Bayes	97.65 %
2	Bank of Kalman filters	93.67 %
3	Model Consensus	92.98 %
4	Robust Kalman filter	90.9 %

Table 6.3: Ranking the performance of the fault isolation accuracy.

The isolation results prove that giving the single time instant approaches the ability to remember their previous sensor status predictions improves the isolation accuracy results from 93.8 % to 97.65 %. The Bank of KF and Model Consensus results were within a 1 % overlapping interval and therefore had to be verified using McNemar's test. This test confirmed that the results differed enough to be statistically significant with $\chi^2 = 442$.

6.3.2 Sensitivity Study on Configuration Parameters

An investigation was performed with the aim of analyzing the effect of the training configuration parameters on the architecture accuracy. Similar to the single time instant approach the architectures were evaluated using default configuration parameters, but can be fine tuned to optimise performance. The values for the default configuration parameters and the range over which the random search was performed is shown in Appendix D.1. The results of the random search are given in figure 6.7.

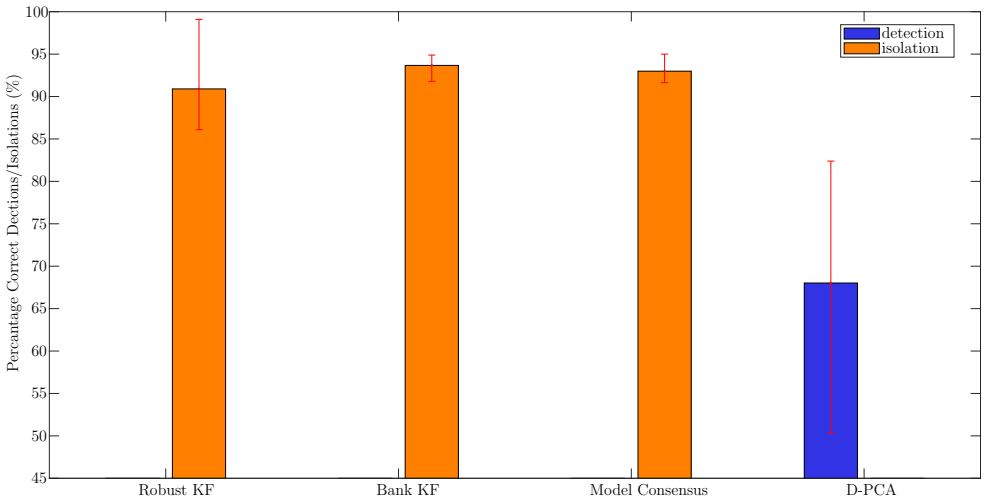


Figure 6.7: The results of a sensitivity study for the various consecutive time instant FDI approaches. The minimum and maximum accuracies achieved are indicated by the red whiskers.

The results show that the Model Consensus and Bank of Kalman filters architectures are insensitive to changes in configuration parameters with a 5 % variation being observed in

the results. The D-PCA architecture shows a large variation in results suggesting that it is very sensitive to the effect of configuration parameters. Ideally, approaches should be insensitive to the effect of configuration parameters to ensure consistency in results. The Robust KF results suggest that the configuration parameters are worth optimizing as it yielded an impressive maximum accuracy 99.15 % when conducting the random search. A closer analysis revealed that when using default configuration parameter values, the Robust KF is prone to false alarms as it fits a very tight decision boundary around the radio altimeter measurements. By adjusting k_4, the decision boundary is relaxed and the false alarm rate is reduced, at the expense of sensitivity and fault reaction time.

6.3.3 Evaluation of Incorrect Classifications

Similar to the single time instant approach the locations of the false alarms, missed opportunities, and incorrect isolations are analysed for the top three consecutive time instant architectures. Table 6.4 summarises the number of false alarms, missed opportunitiess and incorrect isolations of faults when analysing the respective approaches using the test datasets consisting of 283 651 individual data points.

Rank	Approach	False Alarms	Missed Opportunities	Incorrect Isolations
1	Robust KF	4 (0 %)	1 912 (0.67 %)	0
2	Gaussian Naïve Bayes	2 (0 %)	4 882 (1.72 %)	0
3	Bank KF	3 (0 %)	14 031 (4.95 %)	66 (0.02 %)
4	Model Consensus	4 (0 %)	15 285 (5.39 %)	3 953 (1.39 %)

Table 6.4: Analysis of the incorrect assignments performed by the various consecutive time instant approaches.

The optimised Robust KF yielded the best results with 4 false alarms and no incorrect isolations. Giving the single time instant Gaussian Naïve Bayes approach the ability to remember its previous sensor status greatly assisted its performance. Overall the results are similar to the single time instant approach, where most incorrect classifications are attributed to missed opportunities. A concern is the large number of missed opportunities returned by the Bank of Kalman filters and Model Consensus approaches.

6.3.3.1 Robust Kalman Filter

The location of the incorrect classifications for the Robust KF approach is illustrated below in figures 6.8 to 6.11. The Robust Kalman filter FDI architecture fits a tight detection and isolation boundary around the data. The Robust Kalman filter impressively yielded no incorrect isolations and only 4 false alarms.

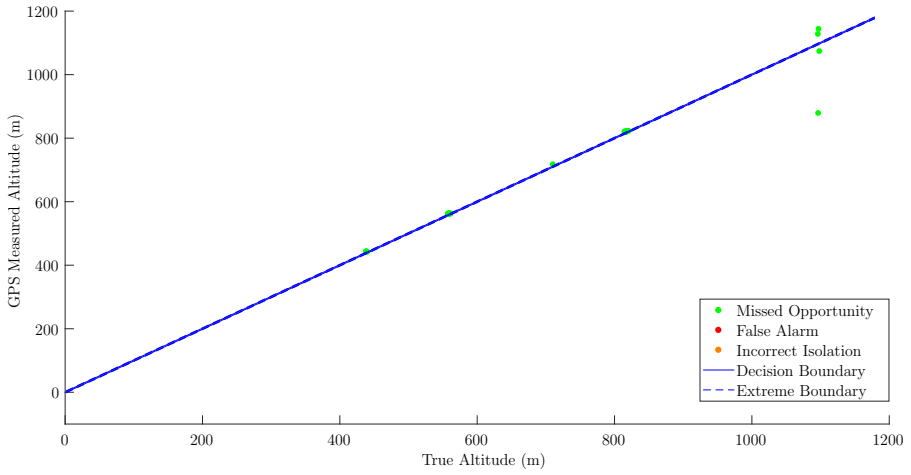


Figure 6.8: The location of the incorrect classifications for the GPS using Robust KF.

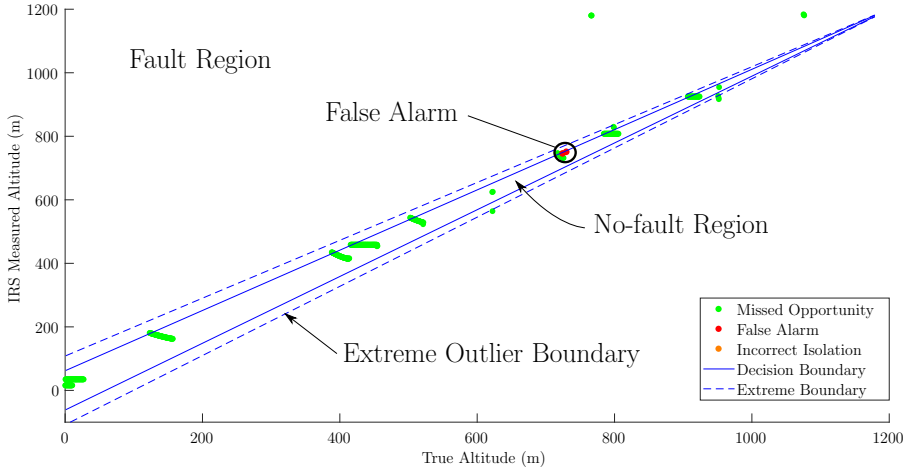


Figure 6.9: The location of the incorrect classifications for the IRS using Robust KF.

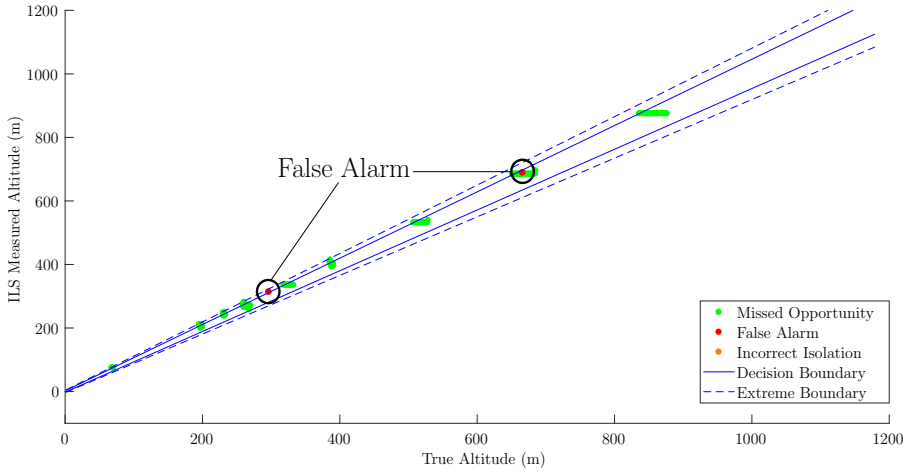


Figure 6.10: The location of the incorrect classifications for the ILS using Robust KF.

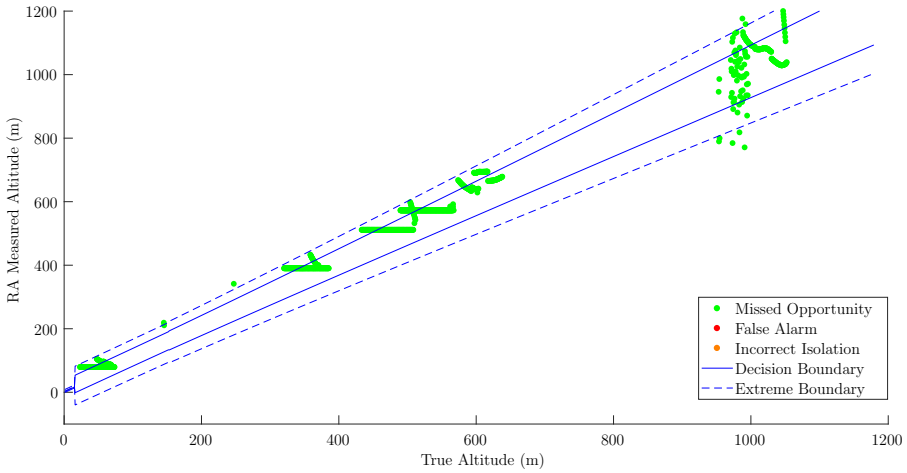


Figure 6.11: The location of the incorrect classifications for the RA using Robust KF.

The locations of the false alarms are favourable as they are all located near the decision boundary. The majority of the missed opportunities are acceptable as they are located within the no-fault region or close to the decision boundary. However there are a few missed opportunities beyond the extreme outlier boundary. Some faults are not immediately isolated when using a window based approach to fault diagnosis, as a certain percentage of measurements from within the window need to be detected and flagged as faulty before updating the sensor status. This ensures that the architecture is absolutely certain a sensor is faulty before updating the sensor status to reflect this prediction. As a result there are a handful of missed opportunities before a sensor fault is correctly identified and isolated as illustrated in figure 6.11 where the architecture was slow to detect an increased noise fault on the radio altimeter. An investigation was performed whereby the size of the sliding window was reduced, but this resulted in more false alarms and incorrect isolations.

6.3.3.2 Gaussian Naïve Bayes

The location of the incorrect classifications for the Gaussian Naïve Bayes approach is illustrated below in figures 6.12 to 6.15.

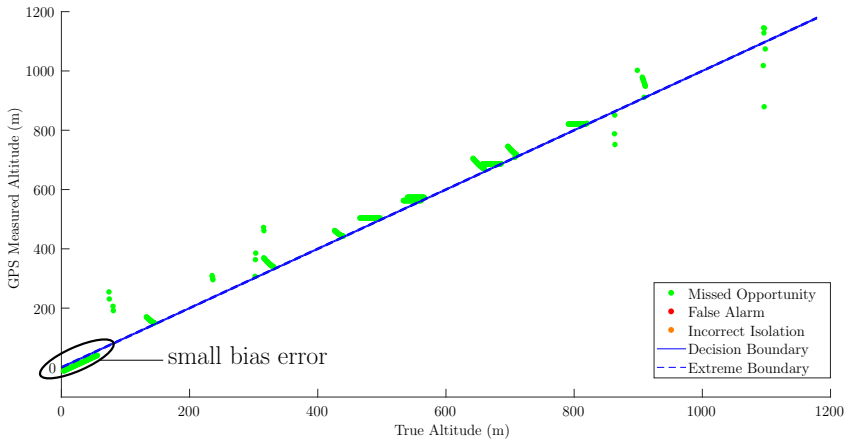


Figure 6.12: The location of the incorrect classifications for the GPS sensor using GNB.

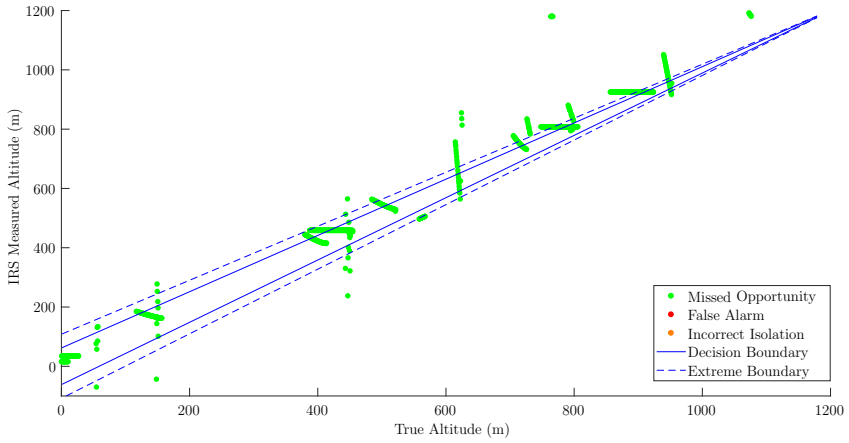


Figure 6.13: The location of the incorrect classifications for the IRS sensor using GNB.

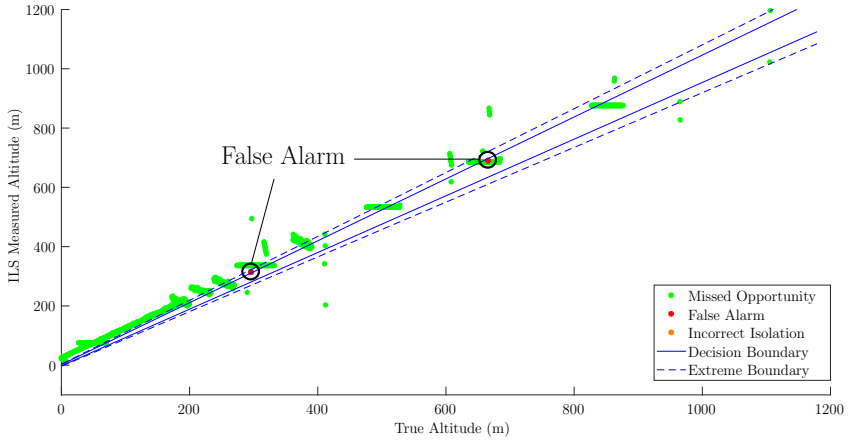


Figure 6.14: The location of the incorrect classifications for the ILS sensor using GNB.

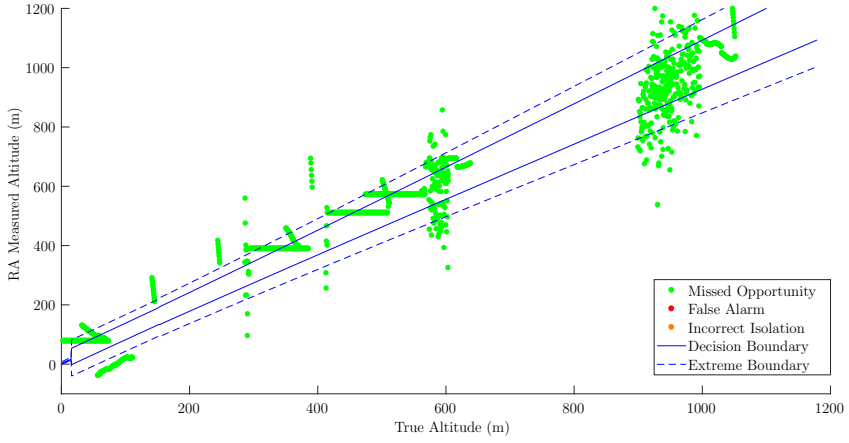


Figure 6.15: The location of the incorrect classifications for the RA using GNB.

Giving the Gaussian Naïve Bayes FDI architecture the ability to remember its previous sensor status predictions resulted in a reduction in the number of false alarms and missed opportunities. The number of false alarms were reduced from 118 to 2, and the number

of missed opportunities was reduced from 17 622 to 4 882. The two false alarms in figure 6.14 are favourable as they are situated near the decision boundary. Two interesting observations are noted with the missed opportunities results. The first is that the Gaussian Naïve Bayes architecture struggles to detect smaller fault profiles such as a bias on the GPS sensor in figure 6.12 as it is aligned with the worst case sensor accuracy. The Gaussian Naïve Bayes decision boundaries do not adapt well to changes in sensor accuracy. The second observation is that most of the missed opportunities for the Gaussian Naïve Bayes architecture are favourable as they are located in the no-fault region or within a close proximity to the decision boundary. However, there are some unfavourable missed opportunity locations. As discussed in the Robust Kalman filter analysis, the drawback of using a two second window of single time instant predictions to update the sensor status is that many fault profiles are not immediately identified as the sensor status will only be updated once the fault detection and isolation architecture is absolutely certain a fault profile is active. This explains why some missed opportunities are located deeper within the fault region. An investigation was performed whereby the time period or window was reduced, but this resulted in more false alarms and incorrect isolations.

6.3.3.3 Bank of Kalman Filters

The location of the incorrect classifications for the Bank of Kalman filters approach is illustrated in figures 6.16 to 6.19. A few shortcomings with the Bank of Kalman filters approach were observed. Firstly, there are many missed opportunities that are located deep within the fault region. When analysing individual runs, it was noted that the Bank of Kalman filters approach performed poorly when identifying an increased noise fault profile due to the conservative design decision that the probability of a hypothesis needs to remain above a probability threshold, P_{thresh}, for a certain period of time. An increased noise profile results in a probability that fluctuates and does not remain above the threshold, $P_{thresh} = 90\,\%$, for long periods. An investigation was performed whereby P_{thresh} was lowered. This unfortunately resulted in a drastic increase in the number of false alarms and incorrect isolations. Secondly, the location of the false alarm in figure 6.17 is unfavorable as it is deep within the no-fault region. These two shortcomings disqualify the Bank of Kalman filters as a reliable fault detection and isolation approach.

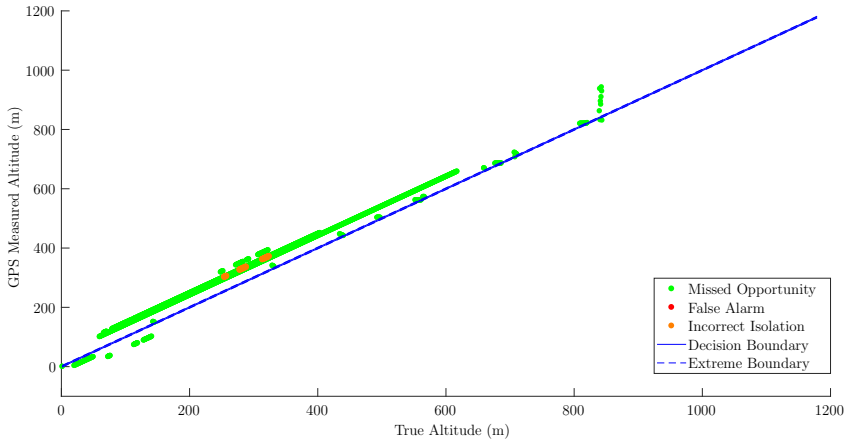


Figure 6.16: The location of the incorrect classifications for the GPS using Bank KF.

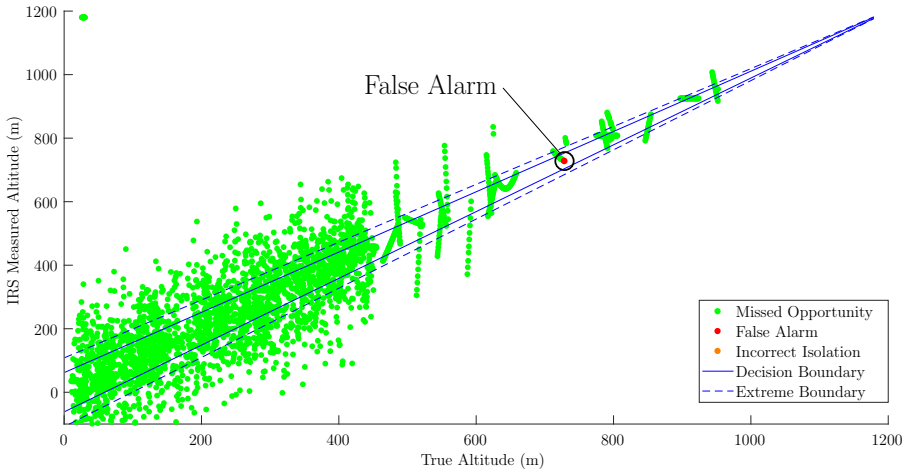


Figure 6.17: The location of the incorrect classifications for the IRS using Bank KF.

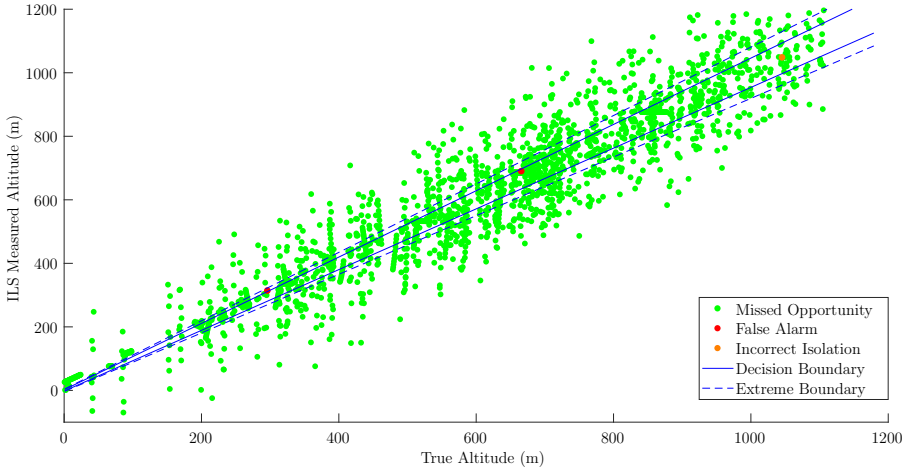


Figure 6.18: The location of the incorrect classifications for the ILS using Bank KF.

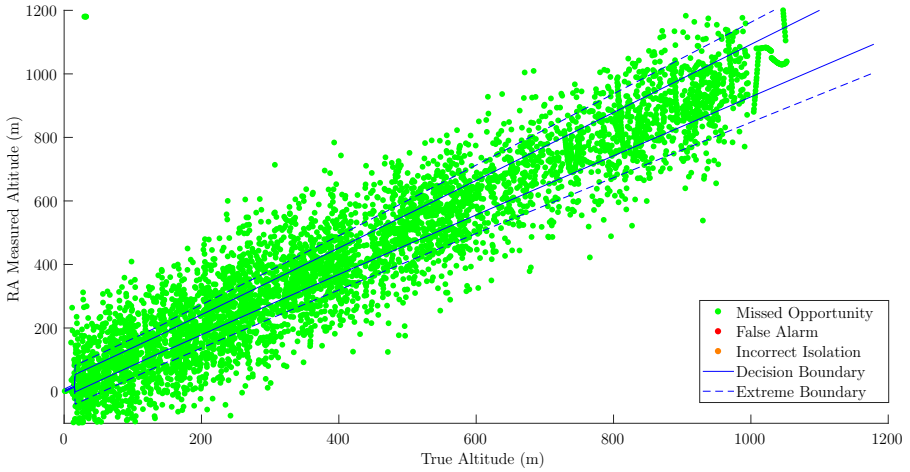


Figure 6.19: The location of the incorrect classifications for the RA using Bank KF.

6.3.4 Results Validation

The FDI architecture accuracy was validated on actual aircraft data with injected faults as discussed in section 4.3.4.

Rank	Classifier	Accuracy
1	Robust KF	98.81 %
2	Gaussian Naïve Bayes	97.89 %
3	Bank KF	95.9 %
4	Model Consensus	91.58 %

Table 6.5: Ranking of the results validation for consecutive time instant approaches.

The results achieved on the validation dataset were compared with the results from the simulated data as illustrated in figure 6.6. All results except for Model Consensus were within the minimum and maximum range of accuracy. The Model Consensus validation result was 0.05 % below the range achieved during testing on the simulated datasets. This difference is insignificant as the Model Consensus result is still less accurate than the Robust KF, Gaussian Naïve Bayes, and Bank of Kalman filters architectures. These results suggest that the consecutive time instant fault detection and isolation architectures would yield similar results if applied on a commercial aircraft.

6.4 FDI Demonstration

The performance of the top three consecutive time instant FDI architectures are demonstrated on example aircraft landing runs with different sensor faults injected. The results are shown in figures 6.20 - 6.23. The sensor status for the various approaches are plotted as a function of time with the actual sensor status shown in red.

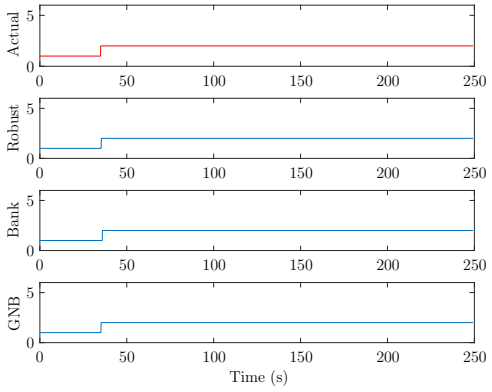


Figure 6.20: Sensor status versus time for a GPS bias fault.

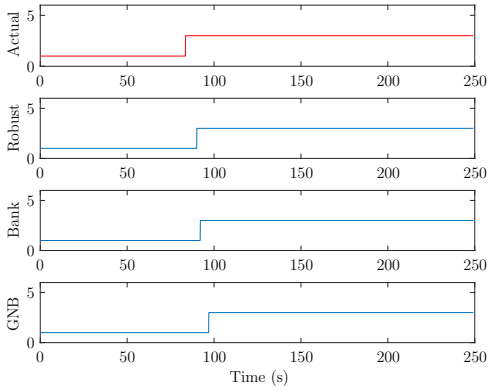


Figure 6.21: Sensor status versus time for a IRS jamming fault.

The FDI demonstration shows that all architectures correctly identified the sensor status soon after the onset of the respective faults. There are a few interesting observations that

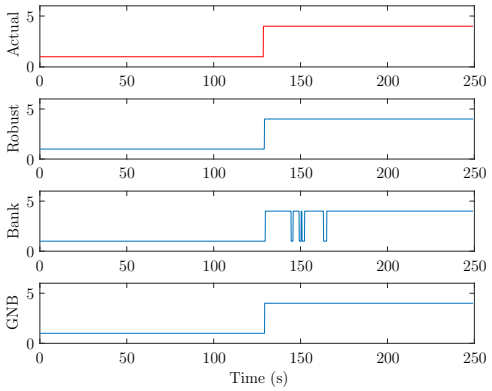


Figure 6.22: Sensor status versus time for a ILS noise fault.

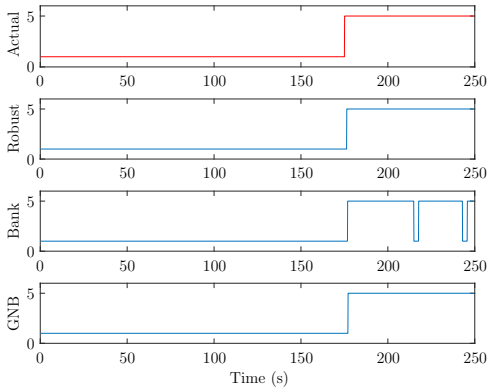


Figure 6.23: Sensor status versus time for a RA oscillating fault.

can be made. Firstly, the consecutive time instant approaches perform better than their single time instant counterparts as the sensor statuses are much steadier. The second observation is that the Bank of KF approach yielded similar results to that of the single time instant approaches in that it also struggles to consistently identify an increased noise and an oscillation error as illustrated in figures 6.22 to 6.23. The use of a probability threshold has a distinct advantage in that it allows for flexibility in the architecture to quickly adapt to changes, but can also be a drawback as the sensor status oscillates.

6.5 Conclusion

The consecutive time instant FDI architectures outperformed the single time instant approaches with architecture accuracies ranging from 92.98 % (Model Consensus) to 99.15 % (Robust Kalman filter). A random search of the configuration parameters revealed a shortcoming with the Robust Kalman filter approach in that it was prone to false alarms as it fitted too tight a decision boundary around the radio altimeter. Using the default values for the remaining architectures will suffice as these yield accuracies close to the maximum accuracy achieved from the random search. An evaluation of the location of the false alarms, missed opportunities, and incorrect isolations revealed that giving the single time instant architectures the ability to remember the previous sensor status significantly improved their performance in this regard. There were few false alarms across all of the consecutive time instant approaches, while the Robust Kalman filter and Gaussian Naïve Bayes architectures also had no incorrect isolations. The results were validated using actual flight data, and the fault detection and isolation performance was demonstrated on different fault profiles by plotting the actual versus reported sensor status as a function of time.

Chapter 7

Sensor Fusion

This chapter considers and evaluates different sensor fusion methods that use the fault detection and isolation information to robustly estimate the height of a commercial aircraft.

This chapter initially gives an overview of the conceptual design and architecture, before introducing the various sensor fusion methods that will be considered. Thereafter these methods are evaluated on a simulated dataset, where the true aircraft height is known. Lastly a visual demonstration of the estimated and true aircraft height is shown as a function of time.

7.1 Conceptual Design and Architecture

The architecture for a robust height estimator is illustrated below.

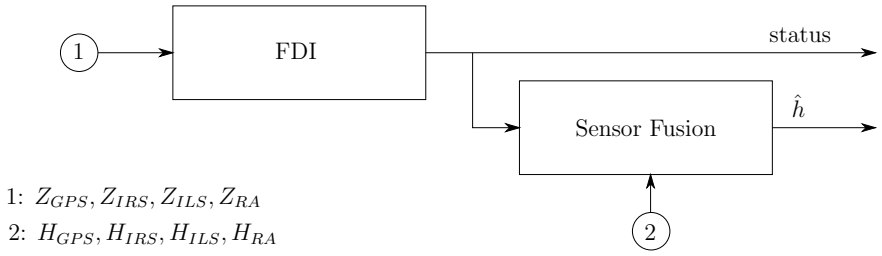


Figure 7.1: The generic architecture for robust height estimation.

The generic architecture for a robust height estimator consists of two sequential stages. The fault detection and isolation (FDI) stage firstly identifies if there is a failure in any of the sensor technologies. When there is a failure, the status is updated to reflect the technology type that experienced a failure. This status is then used by the sensor fusion stage to isolate the faulty sensor technology and thereby prevent it from influencing the height estimate.

7.2 Sensor Fusion Approaches

Three separate methods of sensor fusion were considered and compared. The first is an extension of the current approach used on commercial aircrafts that takes the median of a group of radio altimeter sensors. The redundant radio altimeter sensors are replaced by different sensor technologies. Sensor fusion is performed by taking the median of the group of sensors that are deemed to be healthy. The second method isolates the faulty sensor before fusing the remaining sensors using a weighting function. The third method is that of a Robust Kalman filter in the height domain, where the measurement residual that coincides with the faulty sensor is set to zero.

7.2.1 Median

The median is the oldest and simplest sensor fusion method. The estimated height is given by:

$$\hat{h} = \begin{cases} \text{median}\{H_{GPS}, H_{IRS}, H_{ILS}, H_{RA}\} & \text{status} = \text{no-fault} \\ \text{median}\{H_{IRS}, H_{ILS}, H_{RA}\} & \text{status} = \text{GPS fault} \\ \text{median}\{H_{GPS}, H_{ILS}, H_{RA}\} & \text{status} = \text{IRS fault} \\ \text{median}\{H_{GPS}, H_{IRS}, H_{RA}\} & \text{status} = \text{ILS fault} \\ \text{median}\{H_{GPS}, H_{IRS}, H_{ILS}\} & \text{status} = \text{RA fault} \end{cases} \tag{7.2.1}$$

The advantages of this approach is the ease of implementation and its robustness. It is however expected to result in a suboptimal estimate.

7.2.2 Weighted Averaging

This intuitive sensor fusion algorithm proposed by Elmenreich [58] estimates the aircraft height at each time instant using a weighting function.

$$\hat{h} = \sum_{i=1}^{n} h_i \cdot w_i \tag{7.2.2}$$

where n is the number of healthy sensors, h_i is the height measurement for the i^{th} sensor, and w_i the weight associated with the i^{th} sensor.

A faulty sensor is omitted before fusing the healthy measurements. The weight assigned to the i^{th} sensor is inversely proportional to its variance σ_i^2 when compared with the remaining sensors. A less noisy and more accurate sensor will therefore be assigned a larger weight.

$$w_i = \frac{1}{\sigma_i^2 \sum_{j=1}^{n} \frac{1}{\sigma_j^2}} \tag{7.2.3}$$

The need therefore exists to quantify the sensor variance in terms of accuracy. Using the definition established in Chapter 2.5.1 sensor accuracies are given as absolute maximum bounds for no-fault measurements and can be approximated by $\Delta = 2.695\sigma$. The sensor variance is therefore given by $\sigma_i^2 = (\frac{\Delta_i}{2.695})^2$. The only sensor whose accuracy is a function of height is the radio altimeter. The remainder are manipulated to be a function of

height using the accuracy of the terrain database. The assumption is made that both the sensor and terrain database accuracy can be treated as independent Gaussian distributions and combined following the approach discussed in Chapter 2.5.6.5. A lookup table of sensor accuracies in the height domain was developed, where the median of the height measurements was used as a lookup value at individual time instants. This approach is expected to result in a more optimal height estimate than the median approach, but has a drawback in that it requires prior knowledge in the form of a lookup table or sensor accuracies as a function of height.

7.2.3 Optimal Estimation

The Robust Kalman filter was adapted for optimal height estimation. It uses the sensor status from the FDI architecture to ensure that the correct measurement residual is excluded. The state space model from Chapter 3.3.2 is for inertial space, but height is with reference to the terrain below the aircraft. The climb rate is therefore omitted from the model, as the inertial reference system is no longer valid. The state estimate is therefore given by $\vec{X}_k^- = \vec{X}_{k-1}^+$. The sensor measurements are directly observable and presented to the Kalman filter in the height domain, where the predicted sensor measurements are given by $H\vec{X}_k$:

$$\vec{Y}_k = \begin{bmatrix} H_{GPS}[k] \\ H_{IRS}[k] \\ H_{ILS}[k] \\ H_{RA}[k] \end{bmatrix} \qquad H = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{7.2.4}$$

The last change is that the sensor noise covariance matrix as shown in equation 6.1.5 is adapted to reflect sensor accuracy as a function of height.

$$Q_k = E\left\{W_k W_k^T\right\} = \sigma_w^2 \tag{7.2.5}$$

$$R_k = E\left\{V_k V_k^T\right\} = \begin{bmatrix} {\Delta_{H_{gps}}}^2 & 0 & 0 & 0 \\ 0 & {\Delta_{H_{irs}}}^2 & 0 & 0 \\ 0 & 0 & {\Delta_{H_{ils}}}^2 & 0 \\ 0 & 0 & 0 & {\Delta_{H_{ra}}}^2 \end{bmatrix} \tag{7.2.6}$$

7.3 Height Estimation Results

The accuracy of the three sensor fusion approaches was evaluated on a simulated dataset, where the aircraft height is known. A baseline was established that used perfect fault detection and isolation performance when performing sensor fusion. This baseline is compared against the sensor fusion results that made use of imperfect fault detection and isolation. The top performing Robust Kalman filter was chosen as the imperfect fault detection and isolation system. An investigation was done to evaluate the effect of failures in the different sensor technologies on the accuracy of the height estimate. The steady state and transient errors after fault occurrence were considered separately.

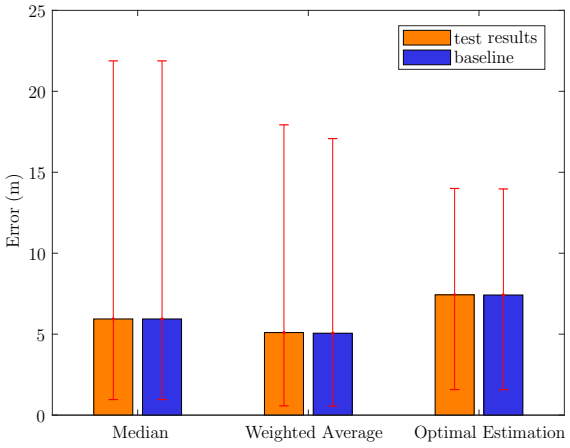


Figure 7.2: The average height estimation error. The 10^{th} and 90^{th} percentiles are shown by the red whiskers.

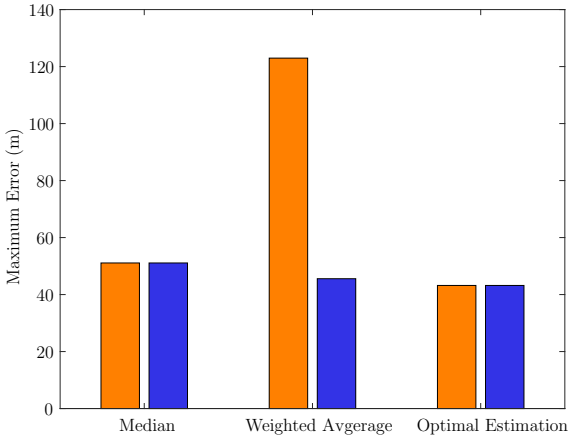


Figure 7.3: The maximum transient height estimate error.

Distinct properties of the three sensor fusion approaches are evident from an overview of the results shown in figure 7.2 and figure 7.3 above. The weighted average and median approach yield an average error of 5.1 m and 5.94 m respectively, as well as exhibiting a large variation in results between the 10^{th} and 90^{th} percentiles. Both the weighted average and median sensor fusion techniques use single instants in time, and have no model, or regard for prior height estimates. The optimal estimation technique incorporates prior knowledge in an attempt to improve the sensor fusion accuracy. It was the most consistent approach with the smallest variation in results, but unfortunately yielded the lowest average accuracy of 7.4 m. The shortcoming of the optimal estimation sensor fusion method is the lack of an accurate model with which future states can be propagated. It is recommended that this approach be investigated further.

The only significant deviation from the baseline results is the transient error returned by the weighted average approach. Fault profiles with instant changes such as a bias and Non-Return-to-Zero fault are not immediately identified by the fault detection and isolation architecture. This results in large transient errors for the weighted average approach. The median approach is inherently robust to the effect of an outlier or sensor failure, therefore its maximum transient error is not affected by the shortcomings of the fault detection and isolation architecture. Similarly, the optimal estimation approach is also advantageous as it reacts much slower to a fault profile, and therefore exhibits smaller transient errors. It can be concluded that the accuracy of the fault detection and isolation architecture has the greatest impact on the weighted average approach.

7.4 Visual Demonstration

The sensor fusion approaches are demonstrated below in figures 7.4 to 7.7. The estimated height and error are shown as a function of time with the dashed line indicating the time of fault occurrence. The upper plot illustrates the estimated height, while the lower plot gives the resultant error for the various sensor fusion approaches. Similar to the FDI demonstration, a variety of fault profiles are selected that are the most difficult to detect across the different sensor technologies.

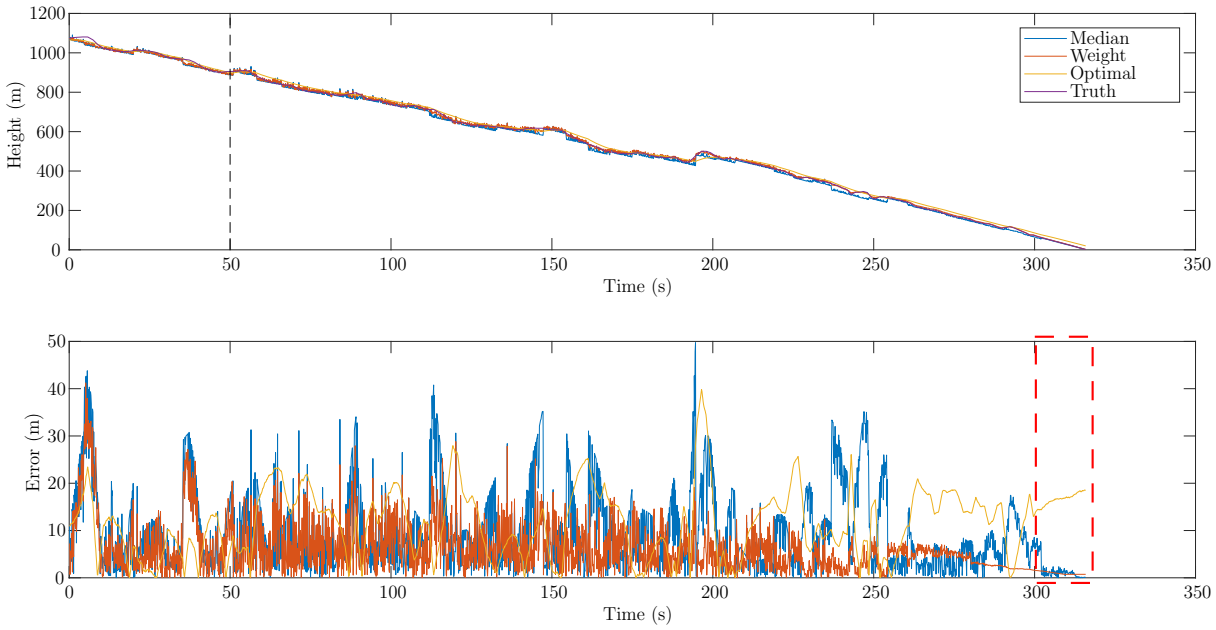


Figure 7.4: The estimated and actual height for a GPS bias fault versus time.

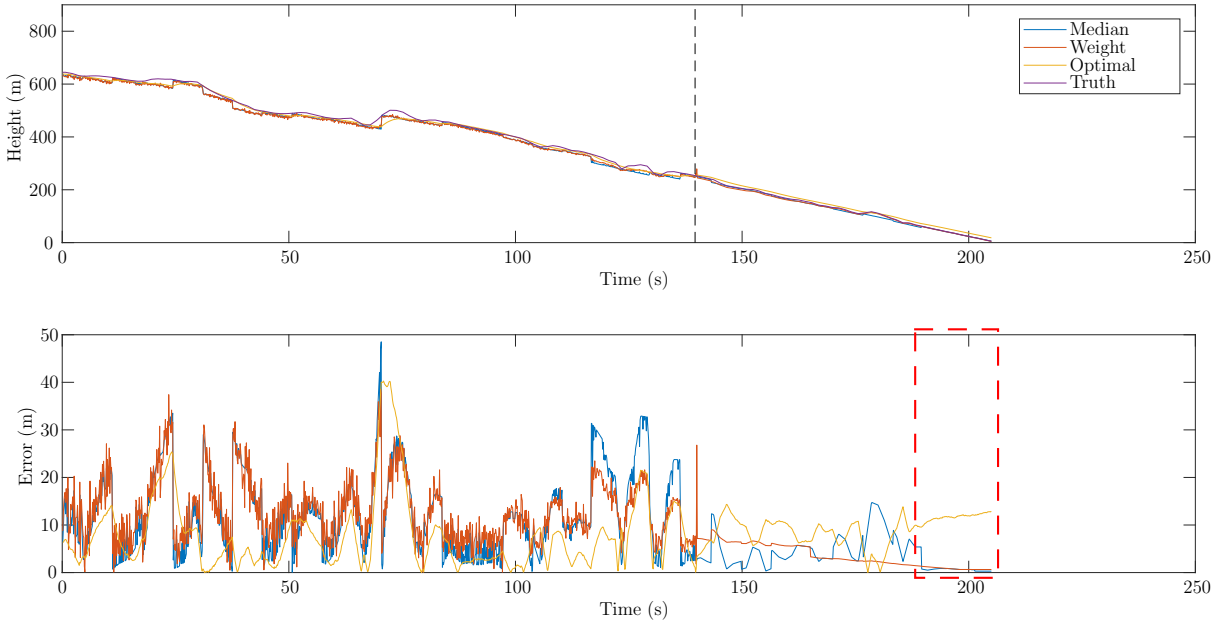


Figure 7.5: The estimated and actual height for a IRS oscillation fault versus time.

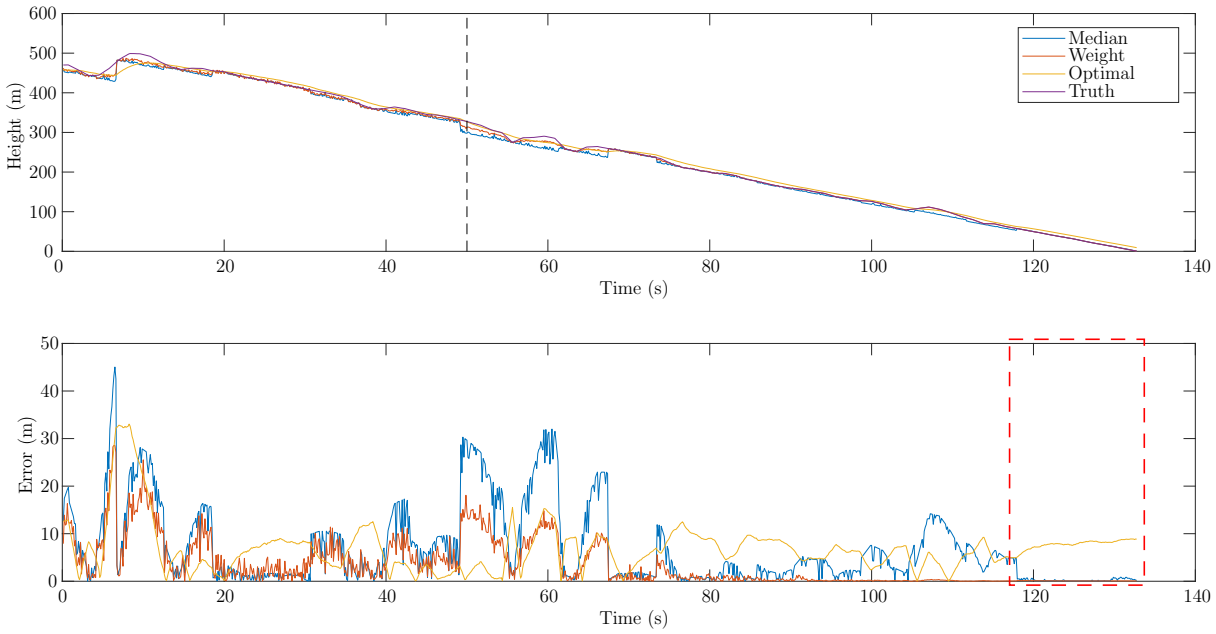


Figure 7.6: The estimated and actual height for a ILS noise fault versus time.

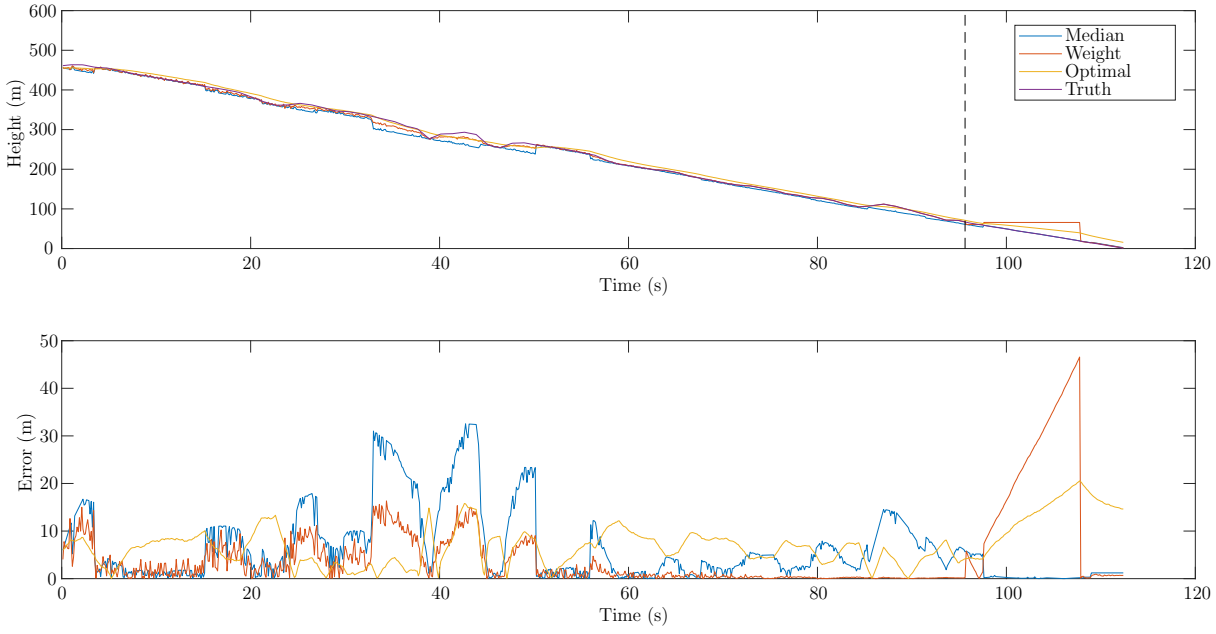


Figure 7.7: The estimated and actual height for a RA jamming fault versus time.

The upper plots in figures 7.4-7.7 show no obvious deviations and estimate the aircraft height in a robust manner. However, the lower plots of the actual height estimation error are concerning as none of the sensor fusion approaches can consistently and accurately estimate the aircraft height. None of the error plots drastically deviate after fault occurrence except for the RA jamming error in figure 7.7, where the fault detection and isolation approach does not immediately identify the fault. Once the fault is identified, the weighted averaging sensor fusion immediately adjusts, while the optimal estimation approach has a smaller transient error as it deviates and corrects itself much slower. An interesting observation is that the error in the weighted average and median approaches

is negligible just before landing (highlighted by the dashed red rectangles), while the optimal estimator performs poorly. This corresponds to the location where the aircraft is above the runway and the effect of the terrain database can be ignored as height and relative altitude is with reference to the runway. This proves that the performance of the median and weighted average sensor fusion methods is dependent on the accuracy of the measurements and terrain database.

7.5 Conclusion

The three sensor fusion architectures yield average accuracies between 5.1 m and 7.4 m, with the 90^{th} percentile error in a range of 14 m and 21.8 m. It is not conclusive which approach yields the best performance and therefore the merits and drawbacks are summarised instead. The weighted average approach has the smallest average error, but is dependent on the accuracy of the FDI architecture. The median is robust to any fault profile, but will always result in a sub-optimal height estimate as it has no regard for the various sensor accuracies. The optimal estimation has the largest average error due to the lack of an adequate model that can be used to predict future height estimates. It is suggested that further work should be done on incorporating the rate of change in height in the Kalman filter model as this should improve its performance. The uncertainty and accuracy of the terrain database should also be investigated further as the use of a more accurate terrain database will decrease the measurement uncertainty and improve the sensor fusion accuracy. The robust height estimator has met the research goal of fusing dissimilar sensor measurements in an manner that is robust to the failure in at most one sensor technology.

Chapter 8

Conclusion and Recommendations

This chapter presents a high-level summary of the work done in this project, highlights the most significant results and observations, and provides a summary of the contributions of this project to the research field. Recommendations are also made for future research on the topic of Robust Height Estimation.

8.1 Summary and Conclusion

This project developed and verified a fault-tolerant sensor fusion system that provides robust height estimates for commercial airliners. Commercial aircraft systems estimate the aircraft height by taking the median of a group of radio altimeter sensors. However, this approach is not robust to the failure of the entire radio altimetry system. The proposed system uses technology redundancy by combining the measurements from the available aircraft sensors to improve the robustness of the height estimate. The robust height estimation problem is divided into two major sub-problems: *fault detection and isolation* and *sensor fusion*. Firstly, a faulty sensor is identified by the fault diagnostic system before the remaining healthy sensor measurements are fused in an optimal manner.

The real dataset supplied by Airbus was not deemed rich enough to serve as comprehensive training and testing data. A simulation model was therefore used to generate synthetic training and testing data. Mathematical models were established for the aircraft motion, the sensors, and the terrain. The structure and nominal parameters for the sensor models were based on information sourced from literature, and then the sensor parameters were tuned to fit a real dataset provided by Airbus. Fault models for six types of sensor faults were also created. The simulation model was used to generate a large dataset of representative sensor measurements containing both "no-fault" and "fault" conditions.

A variety of analytical redundancy approaches were investigated to address the fault detection and isolation sub-problem. These included data-driven and model-based techniques. Two general approaches were considered: fault diagnosis using sensor measurements from a single time instant, and fault diagnosis using sensor measurements from a window of consecutive time instants.

The single time instant fault diagnostic system made use of binary and outlier detection algorithms for fault detection. The multi-class classifiers were activated when a fault was detected and used to identify the faulty sensor. The data-driven outlier detection

algorithms that were considered include: Elliptic Envelope, Local Outlier Factor and Isolation Forest. The following binary and multi-class classifiers were considered: Logistic Regression, Gaussian Naïve Bayes, k-Nearest Neighbors, Decision Tree and Support Vector Machine. The use of outlier, binary and multi-class classifiers required establishing different sets of training data. The outlier detection algorithms were trained on a single class and used as novelty detectors. The binary classifiers were trained on equal percentages of fault and no-fault data, while the multi-class classifiers were trained on equal percentages of fault data. Data preprocessing was introduced to improve class separability and classification accuracy. The following techniques were considered: orthogonal transform, kernel transform and residual transform. Preprocessing was applied to all training and test datasets to ensure that the best combination of classification technique, and preprocessing technique was chosen. Learning curves were analysed using five-fold cross validation to ensure that algorithms were properly trained, and ROC curves were analysed to ensure that the optimal unbiased decision threshold was used. The fault detection accuracy ranged from 95.72 % (Support Vector Machine with kernel preprocessing) to 53.76 % (Local Outlier Factor with no preprocessing). The fault isolation accuracy ranged from 94.16 % (k-Nearest Neighbors with residual preprocessing) to 85.04 % (Support Vector Machine with no preprocessing). The locations of all incorrect classifications for the top three performing single time instant approaches were analysed. This revealed that the single time instant approaches struggled to adapt to changes in the sensor accuracies. The top performing k-Nearest Neighbors with residual preprocessing had false alarms that were situated deep within the no-fault region. Therefore, the Gaussian Naïve Bayes with residual preprocessing architecture was selected as the most reliable single time instant fault diagnostic system. The results were validated on actual aircraft landing data, and the fault detection and isolation performance was demonstrated on different fault profiles.

A similar approach was taken to evaluating the consecutive time instant approaches. The following techniques were investigated: Robust Kalman filter, Bank of Kalman filters, Model Consensus, and Dynamic Principal Component Analysis. The single time instant Gaussian Naïve Bayes with residual preprocessing was given fault detection memory to allow for an objective comparison. This significantly improved the isolation accuracy of the Gaussian Naïve Bayes architecture from 93.8 % (single time instant) to 97.65 % (consecutive time instant). The fault detection accuracy ranged from 99.18 % (Robust Kalman filter) to 68.02 % (Dynamic PCA). The fault isolation accuracy ranged from 99.15 % (Robust Kalman filter) to 92.98 % (Model Consensus). The locations of the incorrect classifications were analysed for the top three consecutive time instant approaches. The top performing Robust Kalman filter had no incorrect isolations, four false alarms situated near the decision boundary, and the majority of its missed opportunities were situated in the no-fault region. Similar to the single time instant approaches, the results were validated on actual aircraft landing data, and the fault detection and isolation performance was demonstrated on different fault profiles.

Both the single and consecutive time instant approaches have their own unique attributes. The single time instant approach is adaptable in the sense that each sample is re-evaluated and classified accordingly with no bias shown towards previous sensor statuses. The drawback to this approach is the large number of missed opportunities. It is impossible to label faulty data points within the no-fault region correctly without prior knowledge that a sensor fault is active. Likewise, the consecutive time instant approach incorporates prior knowledge in the form of previous predictions, trajectory data, or measurement history

to improve performance, but is more rigid and cannot instantly isolate a failure.

The knowledge of an active fault is used to ensure that the estimated height is robust to failure. The following sensor fusion methods were considered: median, weighted averaging and optimal estimation. The 90^{th} percentile for height estimate error ranged between 14 m and 21.8 m for the optimal estimation and the median approach respectively. Simulated data was used to demonstrate the performance of the complete Robust Height Estimation architecture. It can be concluded that this project has successfully met the objectives that were set in Chapter 1. A wide variety of methods and approaches were considered that can be used to improve safety and reliability in the aviation industry by demonstrating their performance on the problem of aircraft height verification.

8.2 Contributions

The main contributions of this project include:

- A fault-tolerant sensor fusion system to provide robust height estimates for commercial airliners was designed and verified. The proposed system uses technology redundancy by combining the measurements from all the available aircraft sensors that provide either height or altitude measurements.
- The proposed system incorporates the use of the Instrument Landing System as a relative altitude sensor with knowledge of the X_{GPS} position.
- The robust height estimation problem was divided into two major sub-problems: *fault detection and isolation* and *sensor fusion*.
- A variety of data-driven and model-based techniques were investigated to address the fault detection and isolation sub-problem. Two general approaches were considered: fault diagnosis using sensor measurements from a single time instant, and fault diagnosis using sensor measurements from a window of consecutive time instants.
- A novel preprocessing technique, the residual transform, was proposed for the height verification problem. The residual transform transforms the original data set that consists of the instantaneous relative altitude measurements into a new dataset that consists of the estimated relative altitude and the instantaneous sensor measurement residuals.
- A simulation model was developed to generate synthetic training and testing data for the data-driven techniques. Mathematical models were established for the aircraft motion, the sensors, and the terrain. The structure and nominal parameters for the sensor models were based on information sourced from literature, and then the sensor parameters were tuned to fit a real dataset provided by Airbus.
- Three methods were investigated to address the sensor fusion sub-problem: using the median of the sensor measurements, using the weighted average of the sensor measurements, and using a Kalman filter to perform optimal sensor fusion.
- The fault detection and isolation, and the sensor fusion were tested using both simulated data and real datasets of actual flight data with synthetic sensor failures injected.

8.3 Future Work

The suggestions for future work and improvements that can be made to the design of a robust height estimator for a commercial aircraft are as follows:

- The sensor models and sensor fault models that were developed in this project should be validated against a larger dataset of actual flight data.
- A comprehensive optimisation of the configuration parameters for all fault detection and isolation techniques (both model-based and data-driven) should be performed.
- The sensor fusion techniques should be investigated in more detail. The height estimation accuracy achieved during this project is limited by the resolution of the onboard terrain database. The implementation of a more accurate terrain database should increase the reliability and accuracy of the height estimate. The inclusion of the rate of change in height for the radio altimeter should also be investigated for the Kalman filter sensor fusion technique.
- The most promising robust height estimation architectures should be implemented on the actual flight computer to verify that the algorithms can be implemented and executed using the limited processing and memory resources of the flight computer.

Bibliography

[1] G. Szafranski, R. Czyba, W. Janusz, and W. Blotnicki, "Altitude estimation for the UAV's applications based on sensors fusion algorithm," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, may 2013, pp. 508–515.

[2] L. Thomas, A. Monin, P. Mouyon, and N. Houberdon, "Sensor fusion for relative altimetry using a hybrid Gaussian mixture filter," in *2013 10th Workshop on Positioning, Navigation and Communication (WPNC)*. IEEE, mar 2013, pp. 1–6.

[3] R. Beard and T. McLain, *Small Unmanned Aircraft: Theory and Practice*, 1st ed. Princeton University Press, 2012.

[4] A. Quinchia, G. Falco, E. Falletti, F. Dovis, and C. Ferrer, "A Comparison between Different Error Modeling of MEMS Applied to GPS/INS Integrated Systems," *Sensors*, vol. 13, no. 8, pp. 9549–9588, jul 2013.

[5] P. S. Maybeck, *Stochastic Models: Estimation and Control: - Google Books*, 1st ed. New York: Academic Press, 1979.

[6] F. Schettini, G. Di Rito, R. Galatolo, and E. Denti, "Sensor fusion approach for aircraft state estimation using inertial and air-data systems," in *2016 IEEE Metrology for Aerospace (MetroAeroSpace)*. IEEE, jun 2016, pp. 624–629.

[7] P. J. Malan, "Upset Detection for Passenger Airliners using Classification of Anemometric and Inertial Sensor Data," Ph.D. dissertation, Stellenbosch University, 2016.

[8] I. Peddle, "Autonomous Flight of a Model Aircraft," Ph.D. dissertation, Stellenbosch University, 2004.

[9] I. Peddle and J. Engelbrecht, "Advance Automation 883- Introductory course to aircraft dynamics," p. 15, 2018.

[10] I. H. Adil, A. u. R. Irshad, and A. u. R. Irshad, "A Modified Approach for Detection of Outliers," *Pakistan Journal of Statistics and Operation Research*, vol. 11, no. 1, p. 91, apr 2015.

[11] W. Wooden, "Navstar Global Positioning System," *Proceedings of the first international symposium of Precise Positioning with Global Positioning System*, vol. 1, pp. 23–32, 1985.

[12] D. Maier and A. Kleiner, "Improved GPS Sensor Model for Mobile Robots in Urban Terrain," *2010 IEEE International Conference on Robotics and Automation*, pp. 4385–4390, 2010.

[13] J. Rankin, "An error model for sensor simulation GPS and differential GPS," in *Proceedings of 1994 IEEE Position, Location and Navigation Symposium - PLANS'94*. IEEE, pp. 260–266.

[14] I. Skog and P. Händel, "Calibration of a MEMS inertial measurement unit," *in procceedings of XVII Imeko World Congress (Rio De Janeiro)*, 2006.

[15] P. Petkov and T. Slavov, "Stochastic Modeling of MEMS Inertial Sensors," Tech. Rep. 2, 2010.

[16] C. C. M. Naranjo and K. T. Hgskolan, "Analysis and Modeling of MEMS based Inertial Sensors Stockholm 2008 Signal Processing School of Electrical Engineering," Kungliga Tekniska Hgskolan, Stockholm, Tech. Rep., 2008.

[17] M. Capkova, "ILS – Instrument Landing System Ground-Based Instrument Approach System," 2009, pp. 40–42.

[18] D. McCollum, "Instument Landing System DDM Calibration Accuracies," Ph.D. dissertation, Airforce Institute of Technology Air University, 1983.

[19] R. Geister, T. Dautermann, and M. Felux, "Total Sytem Error Performance During Precision Approaches," Institute of Flight Guidance, Braunschweig, Tech. Rep., 2014.

[20] Á. J. Jarama, J. López-Araquistain, G. de Miguel, and J. A. Besada, "Complete Systematic Error Model of SSR for Sensor Registration in ATC Surveillance Networks." *Sensors (Basel, Switzerland)*, vol. 17, no. 10, sep 2017.

[21] M. Skolnik, *Introduction to Radar Systems*, 2nd ed. Singapore: McGraw-Hill, 1981.

[22] A. Nebylov and F. Yanovsky, "Radar Altimeter," in *Aerospace Sensors*, 2013, ch. 3, pp. 55–88.

[23] National Telecommunications and Information Administration, "4200 - 4400MHz NTIA," 2014. [Online]. Available: https://www.ntia.doc.gov/files/ntia/publications/compendium/4200.00-4400.00{_}01MAR14.pdf

[24] F. I. Petrescu, *A New Doppler Effect Germany 2012*, 1st ed. Norderstadt: Books on Demand, 2012.

[25] J. Powell, *Aircraft Radio Systems*. Himalayan Books, 1981.

[26] F. T. Malan, "Reduction of the antenna coupling in a bi-static, FM-CW radar system," Ph.D. dissertation, Stellenbosch University, 2011.

[27] C. Hajiyev and R. Saltoglu, "Adaptive filtration algorithm with the filter gain correction applied to integrated INS/radar altimeter," *Journal of Aerospace Engineering*, vol. 221, no. 5, pp. 847–855, 2007.

[28] R. J. Patton, P. M. Frank, and R. N. Clark, "Introduction," in *Issues of Fault Diagnosis for Dynamic Systems*. London: Springer London, 2000, pp. 1–13.

[29] F. Liu, "Data-Based Fault Detection and Isolation (FDI) Methods for a Nonlinear Ship Propulsion System," Ph.D. dissertation, Simon Fraser University, 2004.

[30] N. Meskin, "Fault Detection and Isolation in a Networked Multi-Vehicle Unmanned System," Ph.D. dissertation, Concordia University, 2008.

[31] M. Ballion and L. De Baudus, "Radio Altimeter erroneous values," *The Airbus Safety Magazine*, no. 11, 2011.

[32] D. Larose, "Logistic Regression," in *Data Mining - Methods and Models*. John Wiley & Sons, 2006, ch. 4, p. 158.

[33] I. Witten, E. Frank, and M. Hall, *Data Mining: Practical macchine Learning Tools and Techniques*, 3rd ed. Sprigner, 2011.

[34] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley and Sons, 2001.

[35] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2007.

[36] C. M. Bishop, *Pattern Reconginition and Machine learning*. Springer, 2006.

[37] C. van Daalen, J. Thomas, and C. Jaquet, *Advanced Automation 813*, 2017.

[38] A. Chiu, "Probabilistic Outlier Removal for Stereo Visual Odometry," Ph.D. dissertation, Stellenbosch University, 2017.

[39] G. J. Ducard, *Fault-tolerant Flight Control and Guidance Systems*, ser. Advances in Industrial Control. London: Springer, 2009.

[40] R. J. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society.*, no. Series B (Methodological), pp. 267–288, 1996.

[41] P. Flach, *Machine Learning- The Art of science and algorithms that make sense of data*. Cambridge Press, 2012.

[42] L. Rokach and O. Maimon, *Data Mining with Decision Trees: Theory and Applications*, 2nd ed. World Scientific, 2014.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 2825-2830, 2011.

[44] P. Rousseeuw and K. van Driessen, "A fast algorithm for Minimum Covariance Determinant Estimator," *Technometrics*, vol. 41, no. 3, 1999.

[45] H.-P. Kriegel, E. Schubert, and A. Zimek, "LoOP: Local Outlier Probabilities," in *ACM conference on Information and knowledge management*.

[46] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *ACM Sigmod*, 2000.

[47] H. Liu, X. Li, J. Li, and S. Zhang, "Efficient Outlier Detection for High-Dimensional Data," *Systems, man and cybernetics*, vol. PP, no. 99, 2017.

[48] K. M. Ting, F. T. Liu, and Z.-H. Zhou, "Isolation Forest," *Data Mining*, 2008.

[49] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Apphcatlons to Image Analysis and Automated Cartography," *Communications of ACM*, 1981.

[50] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed. The John Hopkins University Press, 1996.

[51] W. Ku, R. H. Storer, and C. Georgakis, "Disturbance detection and isolation by dynamic principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 30, pp. 179–196, 1995.

[52] L. Kuncheva, *Combining Pattern Classifiers: methods and algorithms*, 1st ed. Hoboken: John Wiley & Sons, 2004.

[53] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classiiers," in *COLT '92 Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[54] M. Claesen, F. De Smet, J. A. K. Suykens, and B. De Moor, "Fast Prediction with SVM Models Containing RBF Kernels," 2014.

[55] C. Montella, "The Kalman Filter and Related Algorithms: A Literature Review," Tech. Rep., 2011.

[56] K. Tanaka and E. Kondo, "Incremental RANSAC for Online Relocation in Large Dynamic Environments," Kyushu University, Tech. Rep., 2015.

[57] S. Roweis, "EM Algorithms for PCA and SPCA," in *Neural Information Processing Systems Conference*, 1998.

[58] W. Elmenreich, "Fusion of Continuous-valued Sensor Measurements using Confidence-weighted Averaging," *Journal of Vibration and Control*, vol. 13, no. 9-10, pp. 1303–1312, sep 2007.

[59] P. Torr and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, apr 2000.

[60] B. Tordoff and D. W. Murray, "Guided sampling and consensus for motion estimation," in *European Conference on Computer Vision*, 2002, pp. 92–96.

[61] O. Chum and J. Matas, "Matching with PROSAC – Progressive Sample Consensus," Czech Technical University, Prague, Tech. Rep., 2005.

[62] N. A. Barbosa Roa, "A data-based approach for dynamic classification of functional scenarios oriented to industrial process plants," Ph.D. dissertation, Université de Toulouse III Paul Sabatier, 2016.

Appendices

Appendix A

Simulation Model Specifications

This appendix documents all additional information on the terrain database and sensor model parameters. The range of sensor accuracy and source of model parameters is presented in a tabular format. The methodology used to derive parameter values that were taken from actual measurement data is described in greater detail thereafter.

A.1 Model Parameters

Terrain Database:

The terrain database statistics were derived from the difference between the less accurate onboard terrain database and the higher resolution database supplied by Airbus.

Name	Description	Range	Source
η_{terr}	Terrain database accuracy	$\mathcal{N}$(10.22,9.16) m	Terrain databases

Table A.1: Terrain Model Parameter Values

GPS Model:

The GPS sensor that makes use of satellite navigation has both a x and z component that was supplied by Airbus.

Name	Description	Range	Source
b_{Zgps}	Z_{GPS} accuracy	$\mathcal{U}$(-1.6764,1.6764) m	Airbus supplied
b_{Xgps}	X_{GPS} accuracy	$\mathcal{U}$(-15,15) m	Airbus supplied

Table A.2: GPS Model Parameter Values

IRS Model:

The IRS sensor derives the aircraft position from the angular rates and accelerations. It has both a x and z component that was derived from a combination of analyzing actual flight data and parameters supplied by Airbus.

Name	Description	Range	Source
$b_{\dot{Z}irs}$	Z_{IRS} drift (time)	$\mathcal{U}$(-0.0127,0.0127) m/sample	Derived from data
$\eta_{\dot{Z}irs}$	Z_{IRS} white noise	$\mathcal{N}$(0,0) m/sample	-
$b_{\dot{X}irs}$	X_{IRS} drift (time)	$\mathcal{U}$(-0.107,0.107) m/sample	Airbus supplied
$\eta_{\dot{Z}irs}$	X_{IRS} white noise	$\mathcal{N}$(0,0) m/sample	-
δ_{drift}	Z_{IRS} drift (relative altitude)	$\mathcal{U}$(-0.051,0.051) m	Derived from data

Table A.3: IRS Model Parameter Values

ILS Model:

The ILS relative altitude sensor model combines the glide slope angle measurement, aircraft x position and glide slope station position. A verification of the glide slope angle noise supplied by Airbus found this parameter to be slightly too conservative and was therefore adapted to better fit the data.

Name	Description	Value/Range	Source
x_{st}	X_R location glide slope station	286 m	Airbus supplied
z_{st}	Z_R location Glide Slope Station	-7 m	Derived from data
η_{Zils}	Glide slope angle white noise	$\mathcal{N}$(0,0.049) °	Derived from data

Table A.4: ILS Model Parameter Values

Radio Altimeter Model:

The radio altimeter makes use of microwaves to measure the aircrafts height. Its accuracy function was supplied by Airbus.

Name	Description	Range	Source
b_{Hra}	H_{RA} accuracy	$\mathcal{U}$(-0.914,0.914) m	Airbus supplied

Table A.5: RA Model Parameter Values

Appendix B

Additional Literature

This appendix documents additional and variations of the outlier detection methods that were considered. A brief overview of each of the techniques is discussed below.

B.1 Variations of RANSAC

There are many variations of the standard RANSAC framework that improve the robustness and model accuracy of the algorithm. Three different variants are considered that improve the manner in which hypotheses are generated and verified such as MLESAC, Guided Sampling and PROSAC. Maximum Likelihood Estimation SAmple Consensus (MLESAC) improves on the hypothesis verification strategy by introducing a probabilistic measure of inlierness. The remaining variants are techniques that improve the manner in which hypotheses are generated.

Maximum Likelihood Estimation SAmple Consensus

MLESAC is an extension of RANSAC that aims to maximize the likelihood of the data as opposed to the number of inliers. This is achieved by introducing a probabilistic cost function, where a penalty is introduced to inlier as well as outlier data points. MLESAC models the residual errors, given estimated model parameters, as a mixture model between Gaussian inlier and uniformly distributed outlier random variables, such that

$$p(\Delta x_i|H(\theta)) = \epsilon\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{\Delta x_i{}^2}{2\sigma^2}} + \frac{(1-\epsilon)}{v} \tag{B.1.1}$$

Where: Δx_i is the residual error of the i^{th} data point, σ is the standard deviation of the inlier distribution and v is the width of the outlier uniform distribution

MLESAC is identical to RANSAC in that it also follows an iterative hypothesise and verify approach. A model is hypothesised by choosing a minimalistic subset of m random points and fitting a model. This model is evaluated over the entire dataset using equation B.1.1. Torr and Zimmerman assume that the probability of a point being an inlier is the same for all points in the dataset $\mathcal{Z}$[59]. A model is evaluated by calculating the negative log likelihood:

$$-L = -\sum_{i=1}^{N} log(p(\Delta x_i|H(\theta))) \tag{B.1.2}$$

The model parameters are chosen for the minimalistic subset that minimizes the negative log likelihood function.

Guided Sampling

The guided sampling algorithm builds on the MLESAC algorithm by aiming to replace random sampling with a more structured approach. The standard RANSAC hypotheses framework assumes that samples are all drawn uniformly from the observation dataset. This is based on the naive assumption that each data point is equally likely to be an inlier. In many applications prior information is available that allows data points to be scored as a consensus measure of their likelihood of being an inlier. This score is used to generate better inlier datasets and improve computational efficiency. Todoff and Murray argue that a data point is more likely to be an inlier if it consistently forms part of the inlier dataset $\mathcal{I}$ [60]. A guided sampling approach uses the available prior information to 'guide' the sampling process and thereby reduce the number of required iterations. Equation B.1.1 can be modified to account for prior information by replacing ϵ with ϵ_i for each data point.

$$p(\Delta x_i|H(\theta)) = \epsilon_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\Delta x_i{}^2}{2\sigma^2}} + \frac{(1-\epsilon_i)}{v} \tag{B.1.3}$$

Computation time is reduced by replacing the uniform sampling strategy with a Monte Carlo sampling method that is more likely to sample inliers. The dataset is arranged in order of increasing probability ϵ_i. Points with a higher probability are therefore sampled more often leading to a reduction in the required number of iterations to ensure a certain confidence interval[60].

The intended application of guided sampling is outlier detection in images to identify features that are inconsistent with the motion of a camera. Image preprocessing is used to calculate a match score which is the measure of the validity of feature match between two images. Match scores are used to identify a consistent set of matches from the entire dataset of possible combinations $\mathcal{Z}$. Although the intended application of guided sampling may differ, the underlying principle is still valid for other applications. Measurements from sensors that are deemed more accurate are more likely to be inliers.

PROSAC

PROgressive SAmple Consensus is another flavor of RANSAC that improves on RANSAC's sampling strategy. The objective of PROSAC is to find the subset of inliers $\mathcal{I}$ in the minimum possible time. PROCSAC as proposed by Chum and Matas operates on similar principles to RANSAC, except that samples are drawn from a reduced dataset consisting of probable inliers. By linearly ordering the complete dataset $\mathcal{Z}$ based on a chosen similarity function[61], a smaller subset is established from which hypotheses are drawn. While this approach draws many similarities with guided sampling, there are two distinct differences. Firstly PROSAC doesn't require prior knowledge of the dataset in the form of probabilities ϵ_i as it uses a similarity function to determine probable inliers. Secondly it dynamically adapts the sampling strategy in accordance to the information revealed by the sampling process itself. For trivial problems PROSAC can perform up to a hundred times faster than the standard RANSAC and at worst yields equivalent performance[61].

B.2 DyClee

Dynamic Clustering for tracking evolving environments (*DyClee*) is an algorithm that groups similar data points from a dynamic data stream together into subsets called natural clusters that can evolve over time. Initially Manhattan distance is used to perform the micro-cluster assignment, thereafter density-based monitoring algorithms will group similar micro-clusters into larger macro or natural clusters. The algorithm is dynamic in the sense that clusters can drift, split, merge or disappear as new data comes in.

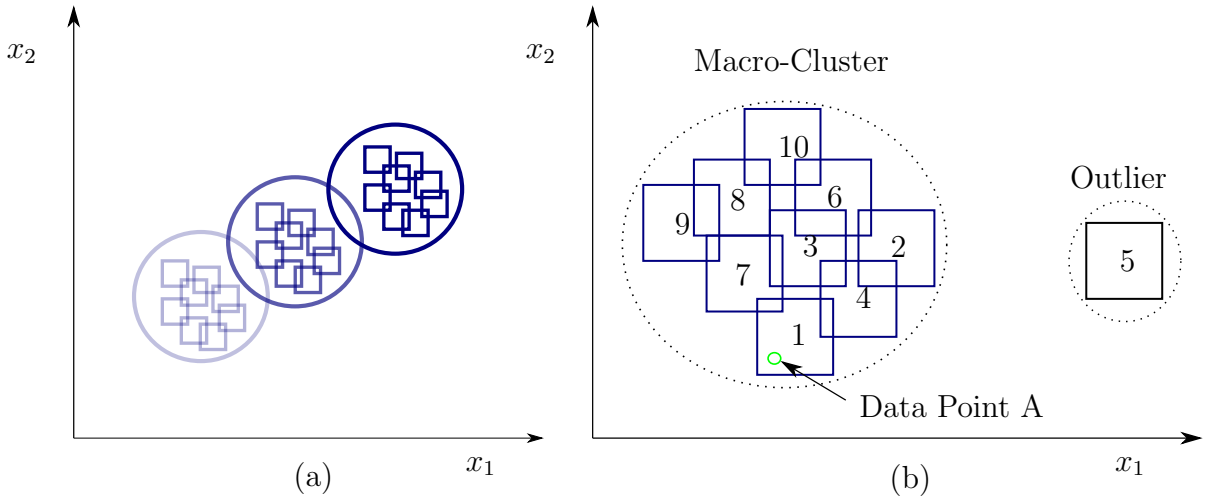


Figure B.1: Cluster drift is depicted in a time varying system shown in figure (a). A forgetting function is used to place a greater emphasis on newer data points. Figure (b) depicts the micro-clusters indicated by the blue squares and the resultant macro-clusters shown with dotted circles. Exemplary Data Point A will be assigned to micro-cluster 1.

DyClee is an online approach comprising of two distinct stages, namely micro and macro cluster assignment. In the first stage the input data stream is divided into micro clusters using a distance based clustering algorithm that operates at the sampling rate of the data stream. A tuple is established for individual micro clusters and different statistical and temporal measures are used to summarize the data points assigned to that micro-cluster[62]. A micro-cluster is a d dimensional hyper-box that contains n_z samples since its creation at time instant t_{sz}. The characteristic tuple CT_z that summarizes information is described by: $[n_z, LS_z, SS_z, t_{lz}, t_{sz}, D_z, lab_z]$. The individual parameters of the tuple can be summarized as follows:

- LS_z - d dimensional vector containing the linear sum of individual features
- SS_z - d dimensional vector containing the square sum of individual features
- tl_z - last time a data sample was assigned to the micro-cluster
- D_z - micro-cluster density defined by $\frac{n_z}{V_z}$, where V_z is the volume of the hyperbox
- lab_z - label assigned to micro-cluster

Dyclee is intended for normalized datasets where the hyper-boxes are designed to occupy a small fraction of the total hyperspace. The parameters are in turn used to determine the centroid of the micro-cluster from which points with in a predefined distance Δ are

assigned to that micro-cluster. A forgetting or decay function is employed during the micro-cluster update step to ensure that new data points are given more emphasis by assigning them a larger weight. This allows for improved tracking and detection of dynamic classes.

Algorithm 4 Pseudocode for the distance based micro cluster assignment.

Input:

z_k	current data point
μC_{k-1}	list of micro-clusters presented in tuples for previous sampling instant
M	number of micro-clusters
Δ	threshold for belonging to micro-cluster

Output:

μC_k	updated list of micro-clusters presented in tuples for current sampling instant

function MICRO CLUSTER ASSIGNMENT(z_k,μC_{k-1}, M,Δ)
  while $i < j$ **do**
    L_i = Manhattan-Dist(z_k,$\mu C_{k-1}(i)$)
    if $L_i < \Delta$ **then**
      $\mu C_k \leftarrow$ Micro-Cluster Update($\mu C_{k-1}(i)$)
    end if
    $i = i + 1$
  end while
  if $i == M$ **then**
    $\mu C_k \leftarrow$ Create New Micro-Cluster ($\mu C_{k-1}, z_k$)
    $M = M+1$
  end if
return($\mu C_k, M$)

The secondary stage employs a density based algorithm that analyses the distribution of the micro clusters and then groups them into macro-clusters. The central idea is that interconnected groups of dense micro-clusters will form a macro-cluster while sparse micro-clusters will remain outliers. Three different categories are considered for the analysis of micro-cluster density, namely dense ($\mathbb{D}$-μC), semi-dense ($\mathbb{S}$-μC) and outlier ($\mathbb{O}$-μC) micro-clusters. The *Dyclee* density analysis algorithm uses either a global and local threshold when categorizing the density of a micro-cluster. The global threshold is derived from the mean and median density for all micro-clusters. $\mathbb{D}$-μC's are those whose density is greater than or equal to both the mean and median threshold. $\mathbb{S}$-μC's are those whose density is greater than or equal to only one of the thresholds. Lastly the density of $\mathbb{O}$-μC is less than both the median and mean threshold. Global density thresholds perform poorly in environments where there are clusters with varied densities as low density clusters will be classified as noise. A local threshold addresses this shortfall by deriving mean and median thresholds from micro-clusters belonging to the same group.

The final macro cluster assignment analyzes the connections and density of micro-clusters. The task of analyzing the neighborhood of individual micro-clusters is computationally expensive and comparable with the nearest neighbors query. *Dyclee* employs a KD-Tree hierarchical data structure that is used to decompose the data space into sub-regions that will simplify the task of locating all directly connected micro-clusters. A more detailed

explanation on the KD-Tree is available in Chapter 3.5.4.5. A macro cluster is created if there exists an inner group of ($\mathbb{D}$-μC)'s whilst every border micro-cluster is either a ($\mathbb{D}$-μC) or ($\mathbb{S}$-μC)[62]. This is based on the assumption that dense micro-clusters are close enough to belong to the same final or natural cluster[62]. Any micro clusters that are less frequently used or have become inactive are stored in long-term memory and are referred to when the micro-cluster assignment fails to assign a data point to an active micro-cluster. This secondary stage occurs at fixed intervals of t_{global} seconds.

Algorithm 5 Pseudocode for density based macro cluster assignment.

Input:

μC_k	list of micro-clusters presented in tuples
M	number of micro-clusters

Output:

C_k	list of macro-clusters representing classes

```
function MACRO-CLUSTER ASSIGNMENT(μC_k, M)
    D_Gk ← Global Density Analysis(μC_k, M)
    D_Lk ← Local Density Analysis(μC_k, M)
    C_k ← Macro Cluster Assignment(D_Gk, D_Lk, M)
return(C_k)
```

Appendix C

Additional Single Time Instant Results

C.1 Multi-class Architecture

Two separate single time instant architectures were considered and evaluated for fault detection and isolation purposes. The design task of identifying and isolating a faulty sensor is inherently multi-class and technically there is no need for a binary classification stage. The architecture below is considerably simpler and more natural for the problem at hand. Training data was generated using equal percentages of each sensor status.

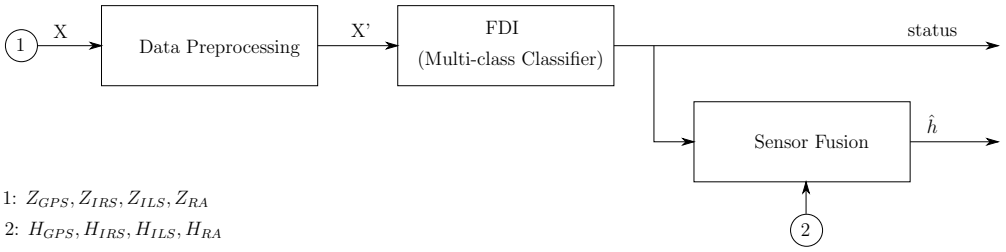


Figure C.1: The alternative architecture for a single time instant fault detector.

C.2 Results

The multi-class architecture was optimally trained using 30000 data points and the detection and isolation accuracy was evaluated using the default configuration parameters and the five different test sets as discussed in Chapter 4.3. The detection and isolation results are shown in figure C.2 and figure C.3.

The results are comparable with the original architecture presented in Chapter 5. The multi-class architecture was found to be marginally less accurate. The top performing architecture was also the k-Nearest Neighbors with 93.4 % accuracy. The results of the two architectures were compared and found to be statistically significant for a probability threshold of at least 95 %. Similar trends were observed between the two architectures with preprocessing aiding the Logistic Regression and Gaussian Naïve Bayes classifiers.

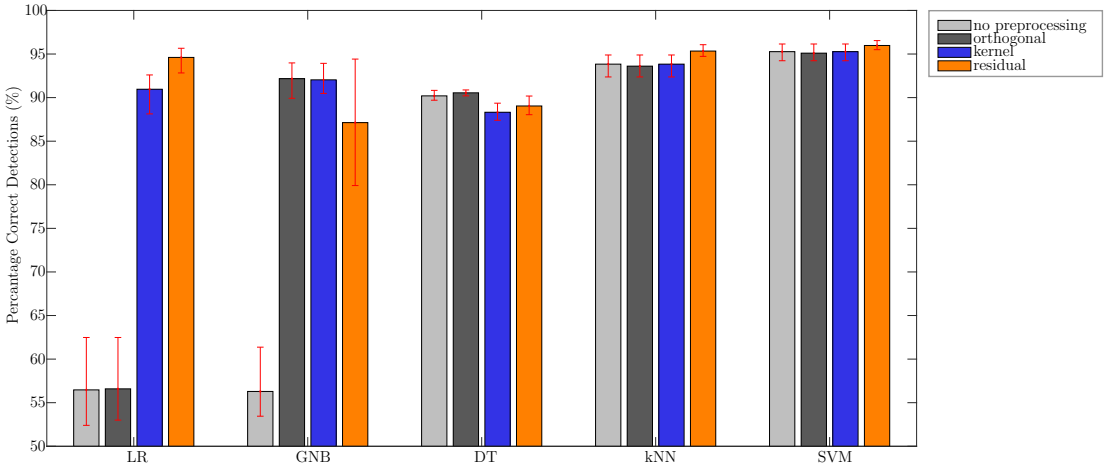


Figure C.2: The detection accuracy for the various classification algorithms.

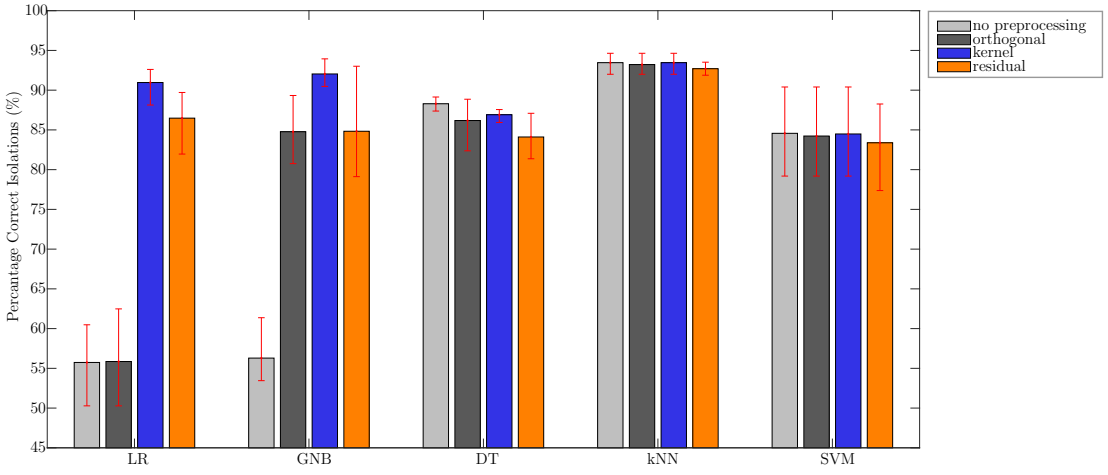


Figure C.3: The isolation accuracy for the various classification algorithms.

C.3 Learning Curves

The learning curves were established to determine the optimal size of the training dataset required when using the default classifier parameter values. The learning curves plot the classifier accuracy against the size of the training dataset. The K-fold methodology divides the dataset into separate training and test sets, with 80 % allocated to training and 20 % to testing when five-fold cross validation. The mean accuracy for the five different test sets is plotted against the size of the training dataset.

C.3.1 Outlier Detectors

The outlier detectors are treated as novelty detection algorithms. They are only trained data that consists of the no-fault class .

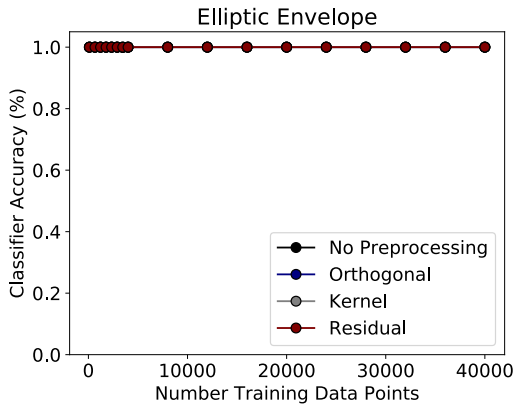


Figure C.4: The learning curve for the Elliptic Envelope.

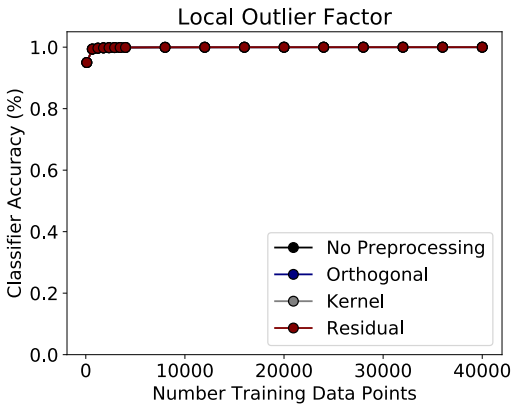


Figure C.5: The learning curve for Local Outlier Factor.

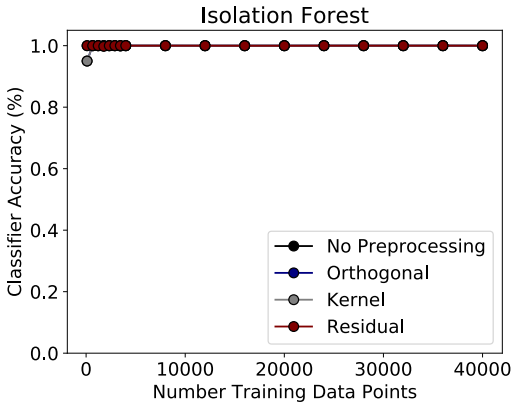


Figure C.6: The learning curve for the Isolation Forest.

C.3.2 Binary Classifier

The binary classifiers are trained on datasets that consist of an equal percentage of fault and no-fault data.

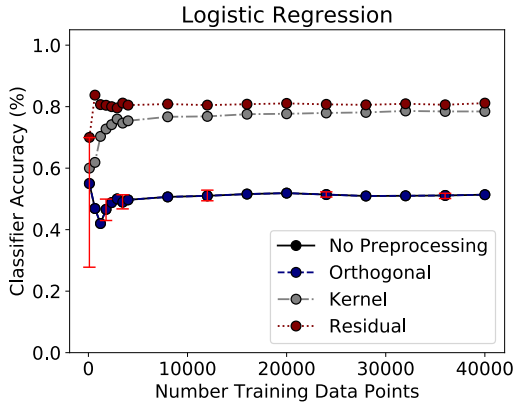


Figure C.7: The learning curve for the Logistic Regression binary classifier.

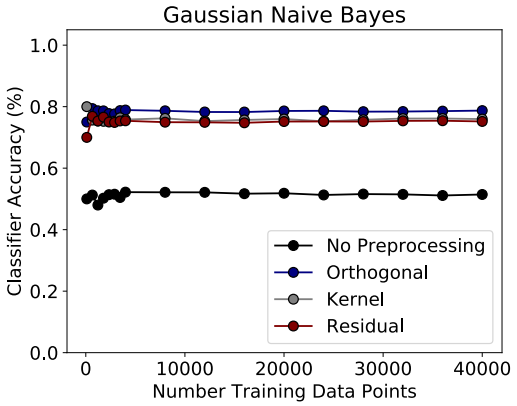


Figure C.8: The learning curve for Naïve Bayes binary classifier.

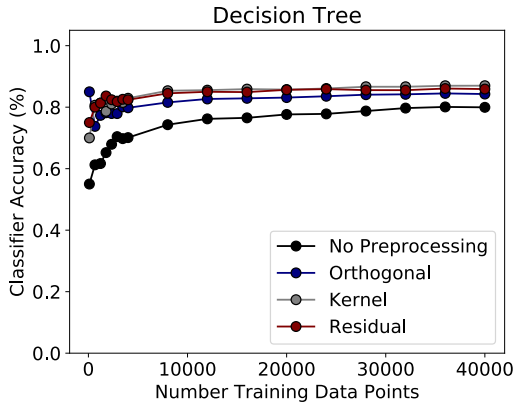


Figure C.9: The learning curve for the Decision Tree binary classifier.

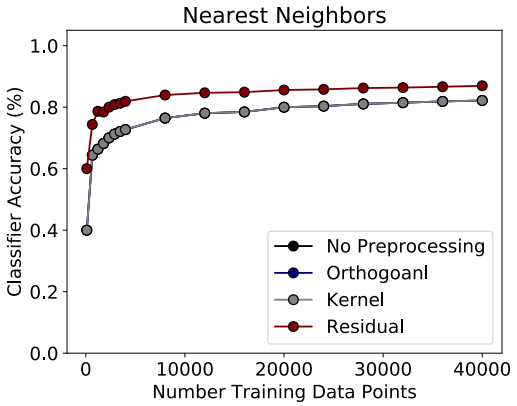


Figure C.10: The learning curve for Nearest Neighbors binary classifier.

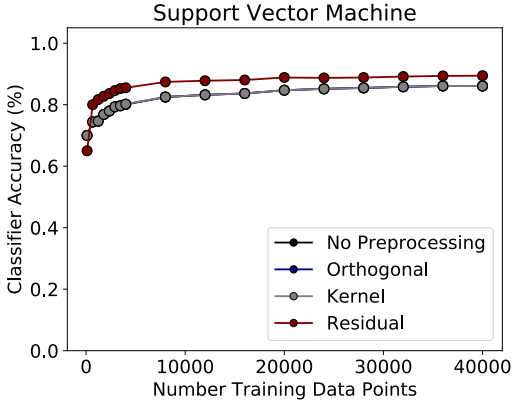


Figure C.11: The learning curve for the Support Vector Machine binary classifier.

C.3.3 Multi-class Classifier

The multi-class classifiers are trained on datasets that consist of an equal percentage of each sensor failure.

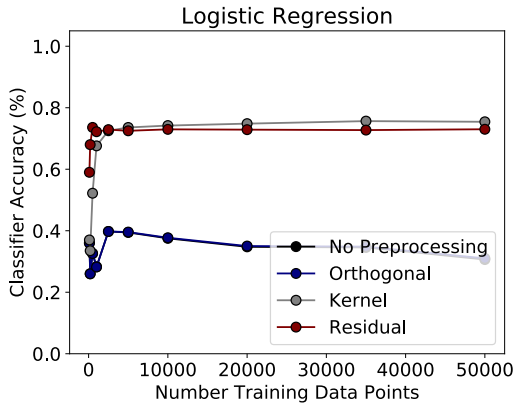


Figure C.12: The learning curve for the Logistic Regression multi-class classifier.

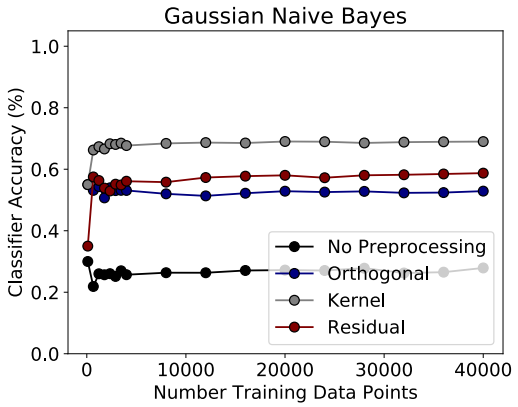


Figure C.13: The learning curve for Naive Bayes multi-class classifier.

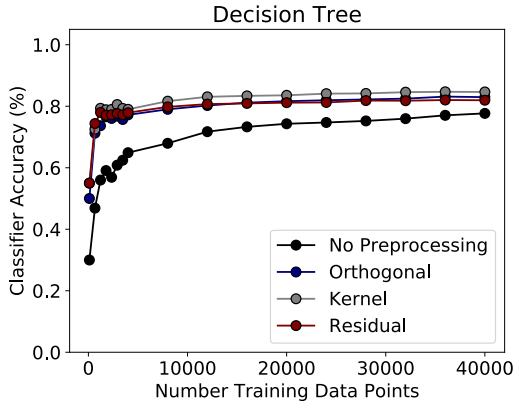


Figure C.14: The learning curve for the Decision Tree multi-class classifier.

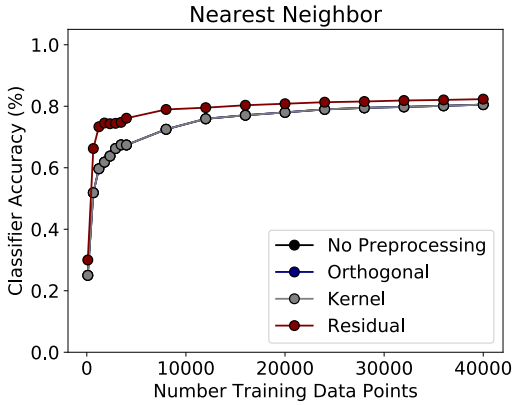


Figure C.15: The learning curve for Nearest Neighbors multi-class classifier.

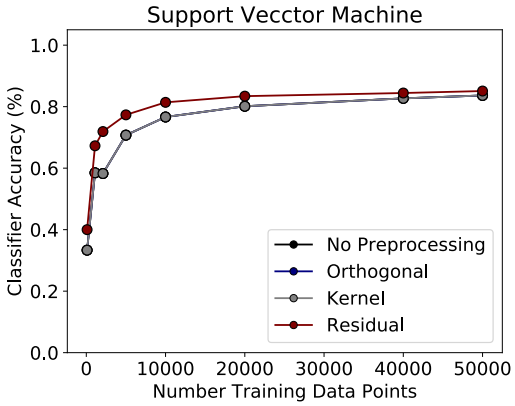


Figure C.16: The learning curve for the Support Vector Machine multi-class classifier.

C.4 ROC curves

The ROC curve ensures that the optimal decision threshold will be used when evaluating a classification algorithm. The optimal decision threshold is one that produces an equal error rate (ERR) of false alarms and missed opportunities. This coincides with the intersection between the dashed line and ROC curves in the figures below.

C.4.1 Binary Classifiers

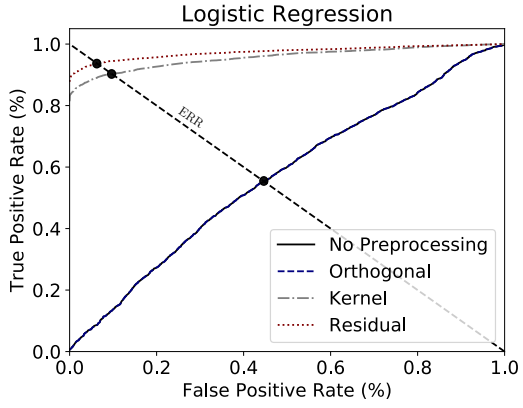


Figure C.17: The ROC curve for the Logistic Regression binary classifier.

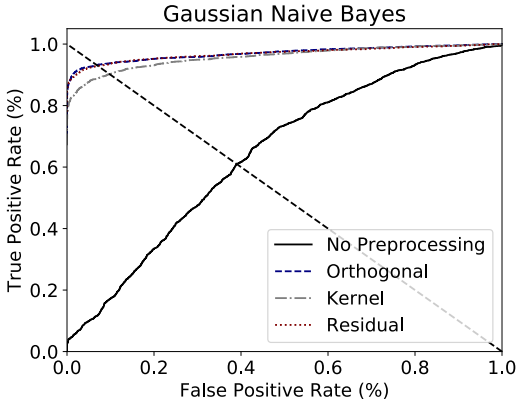


Figure C.18: The ROC curve for the Gaussian Naive Bayes binary classifier.

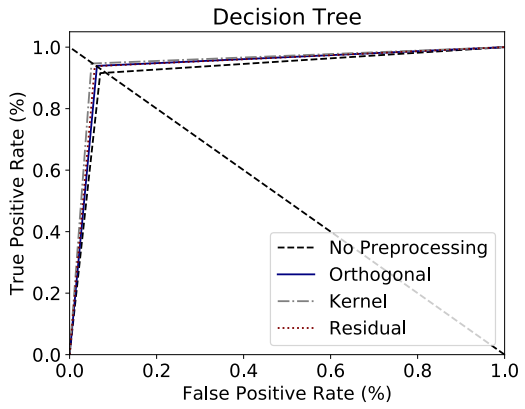


Figure C.19: The ROC curve for the Decision Tree binary classifier.

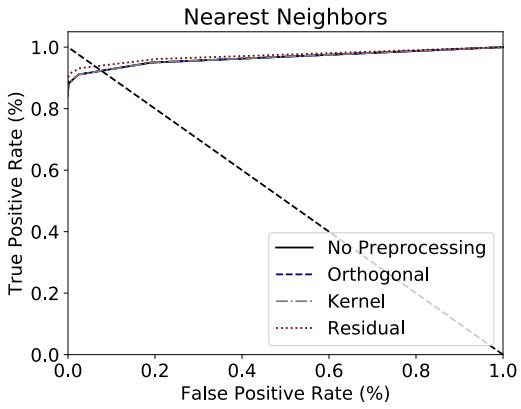


Figure C.20: The ROC curve for the Nearest Neighbors binary classifier.

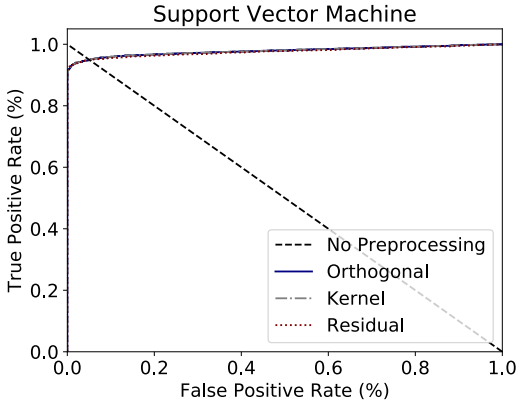


Figure C.21: The ROC curve for the Support Vector Machine binary classifier.

C.4.2 Multi-class Classifiers

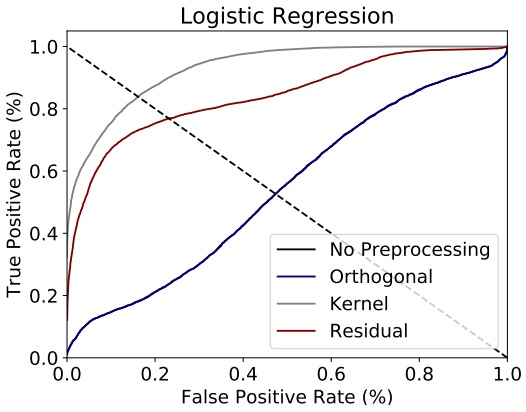


Figure C.22: The ROC curve for the Logistic Regression multi-class classifier.

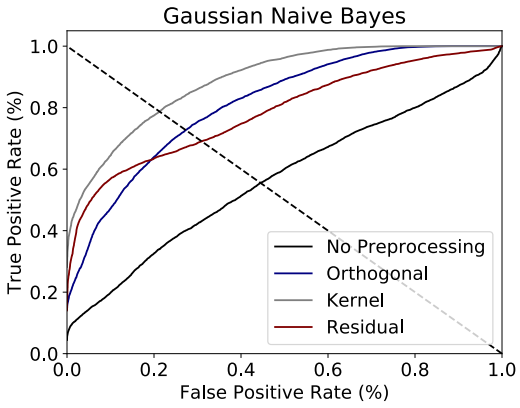


Figure C.23: The ROC curve for the Gaussian Naive Bayes multi-class classifier.

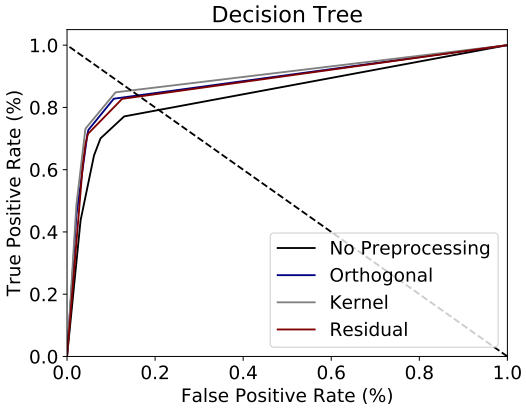


Figure C.24: The ROC curve for the Decision Tree multi-class classifier.

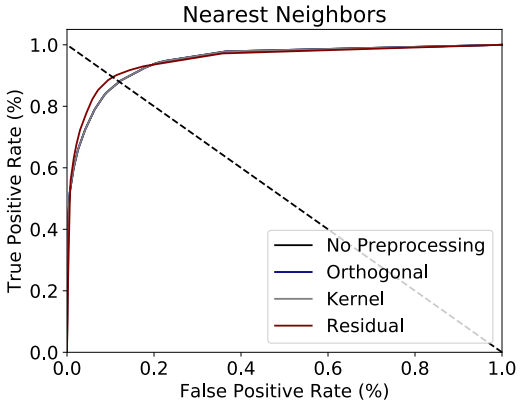


Figure C.25: The ROC curve for the Nearest Neighbors multi-class classifier.

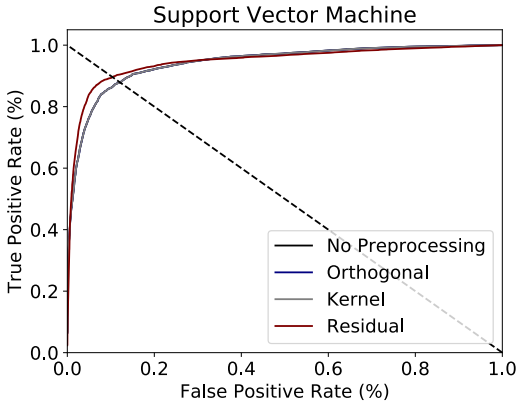


Figure C.26: The ROC curve for the Support Vector Machine multi-class classifier.

C.5 Configuration Parameter Random Search

A study was performed to identify configurable parameters that would have a significant impact on the performance of single time instant architectures. The adjustable parameters were identified and analyzed to ensure that only those that have a significant effect on classifier accuracy were considered. A random search was performed by sampling configuration parameter values from a uniform range surrounding the default value. Random sampling was used for parameters with a fixed number of choices. The search of the parameter space was conducted by repetitively drawing sets of random configuration parameters and analyzing the isolation accuracy on the test set.

C.5.1 Logistic Regression

The default parameter values and the range over which the random search was conducted for Logistic Regression classifier is given in the table below:

Parameter	**Description**	**Default**	**Search Range**
penalty	regularization algorithm	l_2	[l_1, l_2]
C	inverse regularization parameter	1	[0.1, 1.5]
tol	tolerance for stopping criteria	1e-4	[0.001,0.00005]
max_iter	maximum number of iterations used in training	100	[50,150]
solver	optimization algorithm	liblinear	[newton-cg, sag, saga, lbfgs]

Table C.1: The default and random search parameter values for the Logistic Regression classifier.

C.5.2 Naive Bayes Classifier

The Naïve Bayes classifier has no adjustable parameters and therefore is excluded from the sensitivity study.

C.5.3 Decision Tree

The default parameter values and the range over which the random search was conducted for the Decision Tree classifier is given in the table below:

Parameter	**Description**	**Default**	**Search Range**
criterion	impurity measure	gini	[gini, entropy]
max_depth	max tree depth	none	[4,30]
min_samples_split	min samples to split internal node	2	[2,120]
min_samples_leaf	min samples required at a leaf	1	[1,120]
min_impurity_decrease	required impurity change to split	0	[0,0.002]

Table C.2: The default and random search parameter values for a Decision Tree.

C.5.4 Nearest Neighbors

The default parameter values and the range over which the random search was conducted for the K-Nearest Neighbors classifier is given in the table below:

Parameter	**Description**	**Default**	**Search Range**
n_neighbors	number of neighbors considered	5	[2,10]
algorithm	algorithm used to compute neighbors	auto	[auto, ball_tree, kd_tree, brute]
metric	distance metric	euclidean	[euclidean, manhattan, minkowski]

Table C.3: The default and random search parameter values for the K-Nearest Neighbors classifier.

C.5.5 Support Vector Machine

The default parameter values for the Support Vector Machine classifier is given in the table below:

Parameter	**Description**	**Default**	**Search Range**
C	penalty parameter	1	[0.1, 1.5]
γ	kernel coefficient	0.001	[0.0001,1]

Table C.4: The default and random search parameter values for the Support Vector Machine classifier.

Appendix D

Additional Consecutive Time Instant Results

D.1 Configuration Parameters Random Search

A study was performed to identify configurable parameters that would have a significant impact on the performance of the consecutive time instant architectures. A random search was performed by sampling configuration parameter values from a uniform range surrounding the default value. Random sampling was used for parameters with a fixed number of choices. A random search of the parameter space was conducted by repetitively drawing sets of random configuration parameters and analyzing the effective fault isolation accuracy.

D.1.1 Robust Kalman Filter

The default parameter value and range over which the random search was conducted for the Robust Kalman Filter is given in the table below:

Parameter	Description	Default	Search Range
σ_w	process noise variance	0.1	[0.0001, 10]
k_1	GPS decision boundary tuning	1	[1, 1.5]
k_2	IRS decision boundary tuning	1	[1, 1.5]
k_3	ILS decision boundary tuning	1	[1, 1.5]
k_4	RA decision boundary tuning	1	[1, 1.5]

Table D.1: The default and random search parameter values for Robust Kalman Filter.

D.1.2 Bank of Kalman Filter

The default parameter value and range over which the random search was conducted for the Bank of Kalman Filters is given in the table below:

Parameter	Description	Default	Search Range
σ_w	process noise variance	0.1	[0.0001, 10]
$P(\theta = \theta_1)$	prior probability of no failure	0.2	[0.2, 0.95]
$P(\theta = \theta_2)$	prior probability of GPS failure	0.2	[0.0125, 0.2]
$P(\theta = \theta_3)$	prior probability of IRS failure	0.2	[0.0125, 0.2]
$P(\theta = \theta_4)$	prior probability of ILS failure	0.2	[0.0125, 0.2]
$P(\theta = \theta_5)$	prior probability of RA failure	0.2	[0.0125, 0.2]

Table D.2: The default and random search parameter values for Bank of Kalman Filters.

D.1.3 RANSAC

The default parameter value and range over which the random search was conducted for RANSAC is given in the table below:

Parameter	Description	Default	Search Range
L	width of sliding window	4	[3, 7]
v	width of outlier distribution	400	[50, 1000]
ϵ	probability threshold	0.1	[0.25, 0.75]

Table D.3: The default and random search parameter values for Model Consensus.

D.1.4 Dynamic PCA

The default parameter value and range over which the random search was conducted for Dynamic PCA is given in the table below:

Parameter	**Description**	**Default**	**Search Range**
n	width of sliding window	4	[3, 7]
w	order of time lag shift	2	[2, 4]
Δ	error threshold	14200	[5000, 20000]

Table D.4: The default and random search parameter values for Dynamic PCA.