



Titre: Assessment of GPS Spoofing Defences for UAVs
Title:

Auteur: Nicolas Soriano
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Soriano, N. (2023). Assessment of GPS Spoofing Defences for UAVs [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie.
Citation: <https://publications.polymtl.ca/57063/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/57063/>
PolyPublie URL:

Directeurs de recherche: Gabriela Nicolescu
Advisors:

Programme: Génie informatique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Assessment of GPS Spoofing Defences for UAVs

NICOLAS SORIANO

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie informatique

Décembre 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

Assessment of GPS Spoofing Defences for UAVs

présenté par **Nicolas SORIANO**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Omar ABDUL WAHAB, président

Gabriela NICOLESCU, membre et directrice de recherche

Alejandro QUINTERO, membre

DEDICATION

*This thesis is dedicated to my family, especially my parents Barbara Duda and Patrick Soriano,
without whom I would most assuredly not be here today.*

ACKNOWLEDGEMENTS

I would first like to thank my research director, Gabriela Nicolescu. Her help has been extremely useful in pursuing this research.

I would also like to thank Felipe Gohring de Magalhães, research associate. His assistance greatly helped in the process of this masters.

I wish to thank the researchers at the National Research Council, particularly Shahrear Iqbal and Pulei Xiong, who assisted in the first part of this project.

I also wish to thank the Natural Sciences and Engineering Research Council of Canada for providing some funding through grants through their Canada Graduate Scholarship — Master's program.

I wish to also thank my family whose support was instrumental in enabling me to pursue these studies.

I also wished to thank the other students in the HESL Lab for their company during this program.

RÉSUMÉ

Les drones sont des aéronefs avec une utilité importante dans de nombreux domaines, que cela soit militaire, agricole, industriel, commercial, policier ou autres, où ils sont de plus en plus répandus. Ils sont cependant vulnérables à de nombreuses attaques, notamment certaines qui falsifient des signaux GPS. Bien que de nombreuses défenses aient été proposées, il n'existe que très peu de comparaisons de celles-ci permettant de bien comprendre leurs forces et faiblesses relatives. Il n'existe d'ailleurs pas non plus de jeu de données utilisable pour effectuer cette comparaison. Cette thèse a pour but de changer cela, fournissant un jeu de données de vols et une comparaison de plusieurs défenses afin de rendre l'amélioration et la comparaison de défenses, futures comme actuelles, plus aisées.

Ce jeu de données regroupe cent vols d'un drone simulés au travers du système Ardupilot. Ces vols furent simulés dans une grande variété de conditions afin de représenter aux mieux les différentes conditions de vols auxquelles un drone pourrait faire face. Afin de bien pouvoir vérifier la capacité de détection d'attaques d'une défense, la moitié des vols contenaient une attaque qui fit dévier le drone de sa trajectoire prévue. Ces attaques étaient variées, allant de la simple téléportation à des versions bien plus discrètes et complexes.

Différentes défenses furent ensuite implémentées pour essayer de détecter ces attaques au mieux tout en évitant autant que possible des fausses alarmes. Ces défenses furent principalement directement tirées de la littérature, notamment un autoencodeur, une machine à vecteurs de support à une classe ainsi qu'une régression linéaire et autres. À celles-ci s'ajoutent un filtre de Kalman étendu tel qu'implémenté dans le drone utilisé, ainsi que des variations sur celui-ci, et des défenses plus simples, basées sur la vitesse ou la direction du drone, qui servent pour améliorer la comparaison et l'évaluation des défenses précédentes, fournissant des points de repères supplémentaires.

Ce mémoire a déterminé que la meilleure défense contre les attaques de falsifications de signaux GPS envers les drones est un autoencodeur. Ceci fut découvert en simulant l'application de la série de défenses mentionnées précédemment sur le jeu de données de cent vols en testant ces défenses dans une grande variété de conditions.

ABSTRACT

Unmanned aerial vehicles are an important emerging technology with wide-ranging applications in multiple domains, whether these be military, agricultural, industrial, commercial, public safety, or others, where they are becoming more and more widespread. These crafts are however at risk from multiple attacks, notably some that spoof GPS signals. Even though multiple defences have been proposed, there are few comparisons between these that make it possible to understand their relative strength and weaknesses. There is also no dataset suitable to make that comparison. This thesis aims to change this, providing a dataset of a hundred flights as well as a comparison of multiple existing defences to make enhancements and comparisons of defences, both future and current, easier.

This dataset contains one hundred flights of a UAV simulated using the Ardupilot system. These flights were simulated under a wide variety of conditions to better represent the different flight conditions that a UAV might face. To best test a defence's ability to detect attacks, half of the flights contained an attack which made the UAV deviate from its expected trajectory. These attacks were varied, with some acting as simple teleportation while others were both subtler and more complex.

A series of defences were then implemented to try and detect these attacks as best as possible while minimizing the number of false alarms. Defences inspired from literature, notably an autoencoder, a one-class support vector machine as well as a linear regression and others were used. On top of those, an extended Kalman filter as implemented within the used UAV, as well as some variations, and some simpler defences, based on the UAV's speed or direction, were also implemented to add to the comparison and evaluation of the previous defences as they provide more points of comparison.

This thesis found that the best defence against GPS spoofing attacks for UAVs was an autoencoder. This was discovered by simulating the application of the series of defences previously mentioned onto the dataset of one hundred flights, subjecting said defences to a diverse range of conditions.

TABLE OF CONTENTS

DEDICATION	III
ACKNOWLEDGEMENTS	IV
RÉSUMÉ	V
ABSTRACT.....	VI
LIST OF TABLES	XIII
LIST OF FIGURES.....	XVII
LISTE OF SYMBOLS AND ABBREVIATIONS	XX
LIST OF APPENDICES.....	XXI
CHAPTER 1 INTRODUCTION	1
1.1 Motivation.....	1
1.2 Objectives and Contributions.....	2
1.3 Organization.....	2
CHAPTER 2 STATE OF THE ART	3
2.1 Assumptions.....	3
2.2 Attacks	4
2.3 Defences	5
2.3.1 Differences with Other Types of Communication Defences.....	5
2.3.2 GNSS-Specific Detection Methods.....	6
2.3.3 GNSS-Specific Mitigation Methods	8
2.3.4 Defence Techniques Summary	9
CHAPTER 3 METHODOLOGY	12
3.1 Flight Data Collection	12
3.1.1 General Flight Methodology.....	12

3.1.2	Attack Simulation.....	13
3.2	Defence Implementation and Result Collection	14
3.2.1	Implementation Per Flight	14
3.2.2	Evaluation Metric.....	15
3.2.3	Results Collection	19
3.2.4	Comparison Methodology	20
CHAPTER 4 FLIGHT SIMULATION AND DATA COLLECTION.....		21
4.1	Motivations for a New Dataset	21
4.2	Simulators or Real Flights	22
4.3	Simulator Selection	23
4.3.1	Requirements	24
4.3.2	Setups Tested	24
4.4	Dataset Conception.....	25
4.4.1	Missions.....	26
4.4.2	Flight Conditions.....	26
4.4.3	Plane or Copter.....	27
4.5	Types of Attacks	27
4.5.1	Movement	28
4.5.2	Subtlety.....	31
4.6	Dataset Description	32
CHAPTER 5 DEFENCES ASSESSMENT		36
5.1	Implemented Defences	36
5.2	EKF: Extended Kalman Filter.....	37
5.2.1	Description.....	37

5.2.2	Hyperparameters	38
5.2.3	Implementations	38
5.2.4	Results	39
5.2.5	Discussion	42
5.3	Autoencoder-based Defence	43
5.3.1	Description	43
5.3.2	Hyperparameters	44
5.3.3	Implementation	45
5.3.4	Results	45
5.3.5	Discussion	46
5.4	Linear Regression-based Defence	47
5.4.1	Description	47
5.4.2	Hyperparameters	48
5.4.3	Implementation	49
5.4.4	Results	49
5.4.5	Discussion	51
5.5	LSTM (Long Short-Term Memory)-based Defence	52
5.5.1	Description	52
5.5.2	Hyperparameters	52
5.5.3	Implementation	53
5.5.4	Results	53
5.5.5	Discussion	54
5.6	Linear Acceleration Defence	54
5.6.1	Description	54

5.6.2	Hyperparameters	55
5.6.3	Implementation	56
5.6.4	Results	56
5.6.5	Discussion.....	56
5.7	LOF (Local Outlier Factor)-based Defence.....	57
5.7.1	Description.....	57
5.7.2	Hyperparameters	58
5.7.3	Implementation	58
5.7.4	Results	60
5.7.5	Discussion.....	64
5.8	OCSVM (One-Class Support Vector Machine)-based Defence.....	66
5.8.1	Description.....	66
5.8.2	Hyperparameters	67
5.8.3	Implementation	68
5.8.4	Results	69
5.8.5	Discussion.....	74
5.9	Distance Defence.....	76
5.9.1	Description.....	76
5.9.2	Hyperparameters	77
5.9.3	Implementation	77
5.9.4	Results	77
5.9.5	Discussion.....	80
5.10	Speed Defence	81
5.10.1	Description.....	81

5.10.2	Hyperparameters	81
5.10.3	Implementation	81
5.10.4	Results	82
5.10.5	Discussion.....	84
5.11	Heading Defence	84
5.11.1	Description.....	84
5.11.2	Hyperparameters	85
5.11.3	Implementation	85
5.11.4	Results	85
5.11.5	Discussion.....	87
5.12	Combination Defence	87
5.12.1	Description.....	87
5.12.2	Hyperparameters	88
5.12.3	Implementation	88
5.12.4	Results	89
5.12.5	Discussion.....	91
5.13	Mean Defence.....	91
5.13.1	Description.....	91
5.13.2	Hyperparameters	92
5.13.3	Implementation	92
5.13.4	Results	92
5.13.5	Discussion.....	94
CHAPTER 6	DISCUSSION.....	96
6.1	Comparison.....	96

6.2	Important Nuances	98
6.2.1	Sensitivity of Defences.....	98
6.2.2	Time and Power Costs Concerns	98
6.2.3	Flight Plan.....	98
6.2.4	Training Phase.....	99
6.2.5	Flight Dataset Considerations	100
6.3	Implementation Choices	100
6.3.1	Parameterization Step.....	100
6.3.2	Datapoints or Difference	101
6.3.3	Principal Component Analysis	101
6.3.4	Multiplier-Based Thresholds	102
6.4	Limitations	102
6.4.1	Simulator Limitations.....	102
6.4.2	Dataset Limitations	103
6.4.3	Implementation Limitations.....	104
6.4.4	Comparison Limitations	104
6.5	Future Research.....	105
6.5.1	Defence Enhancements	105
6.5.2	Position Prediction	105
6.5.3	Fixing the Limitations	106
CHAPTER 7 CONCLUSION		107
REFERENCES.....		108
APPENDICES.....		112

LIST OF TABLES

Table 2.1 UAV GNSS Defences	10
Table 2.2 UAV GNSS Defences (cont'd)	11
Table 5.1 EKF defence results, second version (position)	40
Table 5.2 EKF defence results, third version (velocity).....	40
Table 5.3 EKF defence results, fourth version (position or velocity)	41
Table 5.4 EKF defence results, fifth version (position and velocity).....	41
Table 5.5 Autoencoder defence results, first version (fused logs)	45
Table 5.6 Autoencoder defence results, second version (GNSS only).....	46
Table 5.7 Linear regression defence results, first version (no parameterization)	49
Table 5.8 Linear regression defence results, second version (with parameterization).....	50
Table 5.9 Linear regression defence results, third version (paper)	50
Table 5.10 LSTM defence results	53
Table 5.11 Linear acceleration defence results	56
Table 5.12 LOF defence results, first version	60
Table 5.13 LOF defence results, second version	60
Table 5.14 LOF defence results, third version	60
Table 5.15 LOF defence results, fourth version.....	61
Table 5.16 LOF defence results, fifth version	61
Table 5.17 LOF defence results, sixth version.....	62
Table 5.18 LOF defence results, seventh version	62
Table 5.19 LOF defence results, eighth version	62
Table 5.20 LOF defence results, ninth version	63
Table 5.21 LOF defence results, tenth version	63

Table 5.22 LOF defence results, eleventh version	64
Table 5.23 LOF defence results, twelfth version	64
Table 5.24 OCSVM defence results, first version	69
Table 5.25 OCSVM defence results, second version	70
Table 5.26 OCSVM defence results, third version	70
Table 5.27 OCSVM defence results, fourth version	70
Table 5.28 OCSVM defence results, fifth version	71
Table 5.29 OCSVM defence results, sixth version	71
Table 5.30 OCSVM defence results, seventh version	72
Table 5.31 OCSVM defence results, eighth version	72
Table 5.32 OCSVM defence results, ninth version.....	72
Table 5.33 OCSVM defence results, tenth version	73
Table 5.34 OCSVM defence results, eleventh version.....	73
Table 5.35 OCSVM defence results, twelfth version.....	74
Table 5.36 OCSVM defence results, thirteenth version	74
Table 5.37 Distance defence results, first version.....	77
Table 5.38 Distance defence results, second version	78
Table 5.39 Distance defence results, third version.....	78
Table 5.40 Distance defence results, fourth version	79
Table 5.41 Distance defence results, fifth version	79
Table 5.42 Speed defence results, first version.....	82
Table 5.43 Speed defence results, second version	82
Table 5.44 Speed defence results, third version.....	83
Table 5.45 Speed defence results, fourth version.....	83

Table 5.46 Speed defence results, fifth version	83
Table 5.47 Speed defence results, fifth version (cont'd).....	84
Table 5.48 Heading defence results, first version	85
Table 5.49 Heading defence results, second version.....	86
Table 5.50 Heading defence results, third version	86
Table 5.51 Heading defence results, fourth version.....	86
Table 5.52 Heading defence results, fifth version.....	87
Table 5.53 Combination defence results, first version	89
Table 5.54 Combination defence results, second version.....	89
Table 5.55 Combination defence results, third version	90
Table 5.56 Combination defence results, fourth version	90
Table 5.57 Combination defence results, fifth version.....	90
Table 5.58 Mean defence results, first version	92
Table 5.59 Mean defence results, second version	93
Table 5.60 Mean defence results, third version	93
Table 5.61 Mean defence results, fourth version	94
Table 5.62 Mean defence results, fifth version	94
Table 6.1 Overall Results.....	96
Table B.1 Complete Results	143
Table B.2 Complete Results (cont'd)	144
Table B.3 Complete Results (cont'd)	145
Table C.1 Flight Coordinates	146
Table C.2 Flight Coordinates (cont'd).....	147
Table C.3 Flight Coordinates (cont'd).....	148

Table C.4 Flight Coordinates (cont'd)	149
Table C.5 Flight Coordinates (cont'd)	150
Table C.6 Flight Plans	151
Table C.7 Flight Plans (cont'd).....	152
Table C.8 Flight Plans (cont'd).....	153
Table C.9 Flight Plans (cont'd).....	154
Table C.10 Flight Plans (cont'd).....	155
Table C.11 Flight Conditions.....	156
Table C.12 Flight Conditions (cont'd)	157
Table C.13 Flight Conditions (cont'd)	158
Table C.14 Flight Conditions (cont'd)	159
Table C.15 Flight Conditions (cont'd)	160

LIST OF FIGURES

Figure 4.1 Deviation Attack (Flight 85)	29
Figure 4.2 Speed Attack (Flight 64)	30
Figure 4.3 Teleportation Attack (Flight 53)	32
Figure 4.4: Flight Types Distribution	33
Figure 4.5 Flight Lengths	34
Figure 4.6: Flight Conditions	35
Figure 5.1 EKF defence graph, first version (baseline)	39
Figure A.1 EKF defence results, second version	112
Figure A.2 EKF defence results, third version	112
Figure A.3 EKF defence results, fourth version	113
Figure A.4 EKF defence results, fifth version	113
Figure A.5 Autoencoder defence results, first version	114
Figure A.6 Autoencoder defence results, second version	114
Figure A.7 Linear regression defence results, first version	115
Figure A.8 Linear regression defence results, second version	115
Figure A.9 Linear regression defence results, third version	116
Figure A.10 LSTM defence results	116
Figure A.11 Linear acceleration defence results	117
Figure A.12 LOF defence results, first version	117
Figure A.13 LOF defence results, second version	118
Figure A.14 LOF defence results, third version	118
Figure A.15 LOF defence results, fourth version	119
Figure A.16 LOF defence results, fifth version	119

Figure A.17 LOF defence results, sixth version	120
Figure A.18 LOF defence results, seventh version	120
Figure A.19 LOF defence results, eighth version	121
Figure A.20 LOF defence results, ninth version	121
Figure A.21 LOF defence results, tenth version	122
Figure A.22 LOF defence results, eleventh version.....	122
Figure A.23 LOF defence results, twelfth version	123
Figure A.24 OCSVM defence results, first version	123
Figure A.25 OCSVM defence results, second version.....	124
Figure A.26 OCSVM defence results, third version	124
Figure A.27 OCSVM defence results, fourth version	125
Figure A.28 OCSVM defence results, fifth version.....	125
Figure A.29 OCSVM defence results, sixth version.....	126
Figure A.30 OCSVM defence results, seventh version.....	126
Figure A.31 OCSVM defence results, eighth version.....	127
Figure A.32 OCSVM defence results, ninth version.....	127
Figure A.33 OCSVM defence results, tenth version.....	128
Figure A.34 OCSVM defence results, eleventh version.....	128
Figure A.35 OCSVM defence results, twelfth version.....	129
Figure A.36 OCSVM defence results, thirteenth version.....	129
Figure A.37 Distance defence results, first version.....	130
Figure A.38 Distance defence results, second version	130
Figure A.39 Distance defence results, third version	131
Figure A.40 Distance defence results, fourth version	131

Figure A.41 Distance defence results, fifth version	132
Figure A.42 Speed defence results, first version.....	132
Figure A.43 Speed defence results, second version	133
Figure A.44 Speed defence results, third version.....	133
Figure A.45 Speed defence results, fourth version	134
Figure A.46 Speed defence results, fifth version	134
Figure A.47 Heading defence results, first version	135
Figure A.48 Heading defence results, second version	135
Figure A.49 Heading defence results, third version.....	136
Figure A.50 Heading defence results, fourth version.....	136
Figure A.51 Heading defence results, fifth version.....	137
Figure A.52 Combination defence results, first version.....	137
Figure A.53 Combination defence results, second version	138
Figure A.54 Combination defence results, third version.....	138
Figure A.55 Combination defence results, fourth version.....	139
Figure A.56 Combination defence results, fifth version.....	139
Figure A.57 Mean defence results, first version	140
Figure A.58 Mean defence results, second version.....	140
Figure A.59 Mean defence results, third version	141
Figure A.60 Mean defence results, fourth version	141
Figure A.61 Mean defence results, fifth version.....	142

LISTE OF SYMBOLS AND ABBREVIATIONS

GNSS Global Navigation Satellite System

LOF Local Outlier Factor

LSTM Long Short-Term Memory

OCSVM One-Class Support Vector Machine

PCA Principal Component Analysis

UAV Unmanned Aerial Vehicle

LIST OF APPENDICES

APPENDIX A Version Graphs.....	112
APPENDIX B Complete Results Table	143
APPENDIX C Flight Info.....	146

CHAPTER 1 INTRODUCTION

This chapter presents the motivations as well as the objectives and the main contributions of this research project. It also details the organization of the thesis.

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) are a rapidly developing technology that is beginning to see widespread use within multiple domains. They have been used to deliver packages in e-commerce, perform various landscape surveying tasks and are even used in military-related matters [1].

Due to these developments and increasing uses, securing UAVs against possible attacks has become a pressing matter requiring serious attention. This has brought to light multiple concerns regarding UAV security [2, 3]. These concerns are quite varied, ranging from the susceptibility to attacks on host operating systems to the vulnerabilities of computer vision machine learning algorithms [4] and communications components [5]. All of these are quite critical to ensure the safety of both UAVs and anything else in their vicinity [6, 7].

These UAV communication components are vulnerable to numerous attacks that can have a critical impact on the UAVs themselves. For example, it is possible to hijack a drone by attacking its communications system, as notably seen in [8]. It is also possible to spoof the Global Navigation Satellite System (GNSS) signals received by a drone to mislead it regarding its true position and then make it land at an originally unintended destination, thereby stealing it [9, 10].

The concerns regarding these attacks are exacerbated by the fact that any mitigation attempts cannot be too demanding because, unlike conventional computers, UAVs do not have access to a powerful computer or large power supply. Thus, addressing such concerns often requires countermeasures to be less computationally intensive than for a standard computer, while attacks do not always face the same limitations [11].

Defences have been developed to protect UAV from these attacks, however, they are rarely compared to other proposed defences and, when they are, these comparisons are to one or two others at most and on only a few flights [12, 13]. There is thus a need for a more comprehensive comparison of the existing defences and data from more flights to use for this comparison.

1.2 Objectives and Contributions

The main objective of this project was to find out which defence is best at detecting spoofing attacks against UAVs. In other words, the one that was able to detect the most attacks while having the least false alarms.

This leads to the first contribution of this project, a dataset of UAV logs for one hundred flights, half of which were spoofed and half of which were not. This requires testing different simulator setups, implementing different attacks and then simulating the flights with these attacks as well as varied flight conditions.

The second contribution is the assessment of the existing defences. Three categories of defences were considered, these being the state-of-the-art defences found directly in the literature, the UAV's built-in extended Kalman filter and some basic defences. This implies the implementation and execution of multiple versions of these defences as well as the definition of metrics for the assessment.

The third contribution is the comparison of these defences. Through the use of the results obtained during the assessment, it was possible to find out which defence would best fit the task of detecting GNSS spoofing attacks against UAV, and thus accomplish the main objective of this thesis.

1.3 Organization

Chapter 2 covers the state-of-the-art documenting GNSS attacks on UAV and the existing defences. Chapter 3 will go over the methodology for the elaboration of the dataset and then the comparison of the defences. Once that is covered, Chapter 4 will detail the choices made for the dataset implementation. After this, Chapter 5 will cover each defence and the results it obtained once applied to the dataset before opening the way for Chapter 6's comparison of the defences in light of these results, as well as a discussion of the results, their limitations and some possible future works. The thesis will then be closed by a conclusion on this work.

CHAPTER 2 STATE OF THE ART

The Global Positioning System (GPS), and all similar Global Navigation Satellite Systems like Galileo, Beidou, GLONASS and others (GNSS will be used to refer to all of these), are systems used for worldwide positioning [5]. They consist of a series of satellites in orbit around the Earth who regularly transmit their current positions at precise and synchronised timestamps. From the values of multiple satellites (at least four), it is possible to calculate the receiver's position with a small margin of error [14]. This position can be quite useful during the operations of the UAV, permitting it to act on location-based missions and correct for any drift caused by equipment imprecision, air resistance, wind or other factors [12].

However, as these signals are neither encrypted nor authenticated [15], they are vulnerable to some of the same attacks as other types of communications, which have been used to affect drones in multiple instances [16, 17]. On top of that, the methods used to defend from these attacks with other types of communications may not be applicable in this case.

This chapter presents the state of the art, covering the possible attacks and then the defences before summarizing the conclusions.

2.1 Assumptions

In order to present the different attacks and vulnerabilities, it is first necessary to specify the assumptions regarding the drone, its actions as well as the attackers, their objectives, and their means. This section concerns itself with exactly that.

The drones considered are those in civilian usage (both commercial and non-commercial). The goals are to secure the drone, its private data as well as whatever goods it may carry from attacks by adversaries. This thesis concerns itself more with civilian UAVs consequently the attackers it focuses upon are cyber criminals and lone hackers. This implies that attacks targeting the supply chain of the drone (such as modifying parts before assembly) are not considered for our attacker profile.

The basic drone mission in this thesis consists of a take-off in a secure location, followed by a flight to a specific destination. This leads to an attack only being possible during the UAV's flight, reducing the possibilities to most man-in-the-middle attacks. It is assumed that the attackers are aware of at least some locations the UAV will pass through (for example because it passed there

before), that they are able to follow it and they have access to more computing power than what is available on the UAV.

The goals of these attackers can be split into two categories, the first being getting secret data (e.g., drone take-off location when used by individuals, company drone routes when used commercially), whether stored in the drone, the GCS or sent through communications as well as the goods transported by the drone or the drone's payload (as an example, a delivery from a commercial company). The second one is to use the drone to be a nuisance or even a serious threat through impairing the services it or other elements in the vicinity provide, whether that be by damaging the drone and its payload, by damaging nearby items or worse. The two following parts will detail the different attacks on drones and how they can be used to fulfill these goals.

2.2 Attacks

GNSS systems are vulnerable to different types of attacks by their nature. There are two types of attacks on which one should focus for GNSS attacks, jamming and spoofing attacks [18]. This is especially true when taking into consideration that the third type of attack, eavesdropping, is a non-concern due to the inherently public and one-way nature of GNSS signals.

Jamming attacks have as goal to cut the communication between the UAV and the satellites it uses to geolocate itself. These one-way signals from the satellites are necessary to calculate the UAV's distance from each satellite and then find its position [19]. No signal implies no location and loss of capability for the UAV. This only requires an emitter capable of emitting on the same frequency as the satellites, as used frequencies are publicly available for GNSS services users' access. This kind of attack can have major effects, seeing as it essentially prevents the UAV from accurately knowing its location. The potential impact of such an attack can even reach mission failure [18]. Jamming attacks are thus defined as attacks that stop the UAV from accurately determining its location by sending a stronger signal garbling the satellite-sent ones and can result in mission failure.

Spoofing attacks, on the other hand, have a different goal. Their purpose is to change the GNSS location calculated by the UAV to a different one (not just making it impossible to calculate the real one like in the case of jamming), which is done through sending weak signals aligned with those of the GNSS network, then strengthening them until they are slightly stronger than the satellite ones and the UAV has locked onto these signals and uses their information instead of the

legitimate, satellite-sent one. Such signals contain fabricated coordinates and timestamps leading to a falsified UAV position that the UAV will interpret as drift and will then correct for, which will actually lead it off-course [16, 18, 20].

These spoofing attacks can have all of the impact of jamming attacks in terms of mission failure because the UAV can be rendered unable to reach a destination location [21]. This attack is however even more dangerous because it can mislead the drone regarding its location without either it or the GCS becoming aware of the deception [22]. This can be used to lead the UAV to a specific location and have it land there [9, 10], believing himself to be back at base or even just make the UAV crash into the ground, resulting in not only the failure of the mission but also the loss of the drone [15]. Furthermore, it can lead to the loss of private data, stored in the drone. This attack is also easy to achieve, requiring little in the way of specialized hardware and software [10].

2.3 Defences

There are multiple methods to mitigate the previous attacks and their impacts, with some work being done regarding the strategies used to protect the UAV and its ability to successfully navigate under attack [13, 19, 23]. This section will discuss these, first covering the differences with other types of communications that make some of the defences for such communications unusable. Afterwards, the methods found in the state of the art to detect spoofing attacks will be covered, before finishing with those that are also able to predict the UAV's real position. Finally, this section will present a summary of the defences.

2.3.1 Differences with Other Types of Communication Defences

There are problems with trying to use the existing methods for communications that jeopardize their employment. While the methods to counter eavesdropping are irrelevant to the present concern as the messages are public by design, the methods to counter jamming and spoofing are of great interest as these are potential attacks on GNSS systems.

A method often proposed to counter jamming in UAV communications is frequency hopping [24]. However, as GNSS signals are sent on a small public set of frequencies, the attacker need only jam them to stop the UAV from receiving the information necessary to calculate its location, making this technique unusable.

The method used to handle spoofing in communication networks is often authentication [25], specifically in this case it would be authenticating the satellites' messages. This, however, would require changing the format of the messages sent by the satellites. In other words, this would mean modifying the satellite firmware, making this solution quite expensive for existing satellites [19].

Given the above considerations, GNSS-specific methods are required for an effective UAV protection against spoofing and jamming attacks.

2.3.2 GNSS-Specific Detection Methods

A type of method that is of great interest regarding the mitigation of attacks on GNSS is to first detect a jamming or spoofing attempt and then act based upon that information [12]. The detection methods are of interest mainly with regards to spoofing detection, as jamming can usually be detected by the lack of usable information within the signals [26]. They are varied, with multiple algorithms used and many different parameters taken into account.

One of the simplest methods for this detection work would be to compare the position obtained through GNSS signals with the one updated throughout the flight by the internal measurement unit (IMU), generally consisting of an accelerometer and a gyroscope [27]. A comparison is made onboard the drone and, should the difference between the two positions be bigger than a defined threshold, the GNSS-obtained one is considered fake and thus the UAV under attack. The problem with this relatively simple method is that the calculations done by the IMU are based upon repeated integrations of acceleration data, resulting in a growing inaccuracy over time and rendering this method unusable [27].

A proposal to evade this problem is to work directly with the linear acceleration of the UAV. This information can be obtained with less errors with the IMU and does not suffer from a growing error over time making it usable as a comparison to the GNSS-derived location [27]. This method however has difficulty with repeated direction changes and the same authors proposed another approach using angular variation. It consists in calculating once with IMU data and another time with GNSS data the angle formed by the North pole, the UAV and its direction, deciding whether the drone is under attack or not based on the difference between these two angles [28].

A similar approach has been proposed by working on the Kullback-Leibler divergence between two distributions, the normal one and the one when under attack. These distributions are calculated

during flight with comparison between past datapoints and new ones to evade the need for training data [29].

Other methods have been proposed using machine learning techniques to process more complex comparisons than simply using a threshold for the difference between internal and GNSS data. One of the earlier proposals was the usage of a modified support vector machine (SVM) on the successive GNSS coordinates to separate the outliers from the normal ones [15]. A more recent proposal was to use a linear regression on the UAV positions. The principle would be that, at the beginning of a flight, the UAV would learn to predict the next GNSS-obtained position based on the previous ones (the number of preceding positions used for a prediction is a hyperparameter of the model). Once it has this predicted position, it would compare it with the actual position obtained through GNSS and, should the two be too different (this difference threshold being another hyperparameter), consider itself under attack [12]. A third method proposed was to use a neural network that would be trained on multiple parameters of the signal, notably the signal-to-noise ratio, the amount of Doppler effect, the satellites whose signals are received, and which would take these into account to decide if the signal is considered fake or not [19].

Tree-based classifiers such as random forests were also tested to find out how useful they could be for spoofing detection, with some satisfactory results [13]. Another option was proposed using the fusion of data from multiple sensors to reduce errors from IMU measurements through a convolutional and recurrent neural network. This trajectory is then compared with map data and this information is used to decide whether spoofing is detected or not [30].

Some recent techniques have also been developed using novelty learning to classify GNSS data and identify falsified coordinates. They consist in learning normal data (i.e., not modified by attackers) and then classifying new datapoints based upon whether they are too different from the training (normal) data, with some of the algorithms used for this being one-class SVM and autoencoders [31].

Finally, some methods, instead of working with the signal or the coordinates, use computer vision algorithms to either find the UAV's location or at least to determine whether it is being attacked. One of these has as its working principle to use the Lucas-Kanade algorithm that evaluates the movement of the pixels of a grayscale image over time and is thus used here to estimate the UAV's

speed. This speed is then combined with the IMU-obtained one and this fusion is compared to the GNSS-received information to detect a difference that would indicate spoofing [32].

Though they are less widely useful as they are applicable to far less cases, there are a few methods that can be used specifically when there are multiple UAVs working together. Notably, should only one UAV be attacked within a group of five or more, it becomes possible to identify the compromised one [33]. It can be done by retrieving for each drone its GNSS position. Then, one drone calculates its position based upon a triplet of other drones' coordinates, doing so for every group of three. Should the drone doing the calculations be the one attacked, its GNSS-derived position will not correspond to any calculated from the other drones' coordinates. Should another drone be the one attacked, the initial drone's GNSS coordinates will match the one calculated from a specific triplet, the one without the compromised drone [33]. As well as these approaches, some have also been proposed using other drones as escort to a main UAV where they all would compare the strength of the signal each one received taking into account their relative position and using this information to determine whether said signal is legitimate or not [34].

2.3.3 GNSS-Specific Mitigation Methods

Once an attack is detected, it is necessary to handle its effects. There have been multiple attempts [12, 30] at finding methods to navigate using some semblance of GNSS while under spoofing or jamming attack, whether it be using other drones' information [33] or even the attacker's own [26] with the most basic of methods being to navigate with the IMU measurements as there is a lack of better data for the UAV at that point in time, considering the GNSS module is under attack.

However, it is possible, using methods discussed previously, to use more refined techniques. Notably, in the preceding subsection (2.3.2), a technique using linear regression to predict the next position was discussed. This position can be used instead of the one obtained through GNSS measurements when these are compromised [12].

Another method that had previously been discussed was the usage of other drones with a method akin to triangulation to identify which drone within a swarm is compromised. However, this method also permits finding its actual coordinates, as the ones obtained from calculations with triplets of uncompromised drones are accurate, and thus usable [34].

In the same way as these methods, the usage of DeepPose permits finding the drone's actual position, handling both the problem of detection and determining the UAV's actual location. However, some of its techniques only work within specific types of areas, notably roadside [30].

A similar method had also been proposed, using road data and computer vision. It consisted in using a hyperspectral sensor to find the sides of a road and then fitting curves to the identified road in the UAV's normal camera view. These are then used to follow the road when under attack [35].

Another set of methods proposed to alleviate the effect of GNSS jamming and spoofing is to use the information given by other external sources, whether by third-parties or even the attacker. Recent work has been done in this area, notably using a UAV as a backup out of reach of attackers, thus being able to use its own GNSS information, but keeping in sight other drones and updating and informing them on their position [36].

Some other methods also proposed using third-party cellular wave information to find the drone's actual position in a GNSS-denied environment [37, 38]. They could potentially be applied when the UAV is under attack.

Another source of information that can be used is the attacker itself. If the source of the adversarial signal is stationary, it becomes possible to skirt around the edge of its effect region and then find out the emitter's location. Once this is done, the drone can use this information combined with the variation in the strength of the attacker's signal to calculate its own movement and thus location [26].

2.3.4 Defence Techniques Summary

There exist multiple methods for detecting GNSS jamming and spoofing, ranging from comparison between expected and received positions to complex machine learning algorithms based on signal characteristics. There are also methods to find an estimate of the drone's actual position with some of these algorithms, whether they are used with machine learning, other drones or even the attacker itself.

All of these are presented in Table 2.1 which summarizes the different defences and whether they protect against jamming, spoofing or both. It also notes whether they serve to detect the attack, to localize the drone once it has been attacked or both as well as describing their functioning and how far along their implementations are. The implementation levels are defined as follows: theoretical

when only a theoretical study of the method has been performed, dataset when the method has been tested using flight data, simulation when the method has been implemented on a simulator, and implementation when the implementation on drone has been performed.

Table 2.1 UAV GNSS Defences

Defence	Jamming or Spoofing	Detection or Localization	Method	Implementation
Authentication	Spoofing	X	Authenticate satellite signals	Theoretical
IMU position comparison [27]	Spoofing	Both	Compare GPS position with IMU-estimated one	Practical
IMU linear acceleration comparison [27]	Spoofing	Detection	Compare GPS-estimated linear acceleration with IMU-obtained one	Practical
IMU angular variation comparison [28]	Spoofing	Detection	Compare GPS-estimated angular variation with IMU-obtained one	Practical
Kullback-Leibler comparison [29]	Spoofing	Detection	Compare past and present position distributions	Simulation, Practical
SVM [15]	Spoofing	Detection	Compare present to normal data points	Simulation
Linear regression [12]	Spoofing	Both	Compare present to predicted position based on previous positions	Simulation
Neural network [19]	Spoofing	Detection	Classify signal information based on learned data	Dataset
Random forest [13]	Spoofing	Detection	Classify signal information based on learned data	Dataset
DeepPose [30]	Spoofing	Both	Use sensor fusion and a convolutional recurrent neural network to classify signals	Dataset
Novelty learning [31]	Spoofing	Detection	Classify signal based on learned normal data	Practical

Table 2.2 UAV GNSS Defences (cont'd)

Lucas-Kanade vision [32]	Spoofing	Detection	Compare pixel movement in camera-implied speed to GPS	Practical
Cooperative localisation [33]	Both	Both	Compare GPS-received data with position implied by other UAV locations	Simulation
Trilateration [34]	Both	Detection	Compare each drone's received signal strength	Theoretical
Road following [35]	Both	Localization	Find a road visually and then follow it	Simulation, Practical
Third-party cellular wave information [37, 38]	Spoofing	Both	Detect discrepancies between received GPS signal and third-party cellular information	Simulation, Practical
Parent-child grouping [39]	Both	Both	Keep parent drone out of reach of attackers while keeping track of child drones using vision algorithms	Simulation, Practical
Attacker signal localization [26]	Jamming	Localization	Use received signal strength to determine distance from attacker	Simulation

CHAPTER 3 METHODOLOGY

This chapter covers the methodology. It is divided in two main sections, the first covering the steps followed during a flight used for data collection, and the second presenting the process used for defence implementation, results collection, and comparison.

3.1 Flight Data Collection

This section goes over the collection of flight data used to assess defences afterward, specifically covering in the first subsection the general process followed during a flight to obtain flight traces, and in the second how attacks were implemented. Do note that the parameters for the flights as well as the types of attack implemented are covered in Sections 4.4 and 4.5 of the next chapter respectively.

3.1.1 General Flight Methodology

In order to collect useful data, a set of steps are required. First, the parameters for the flight are needed (wind direction, wind speed, GNSS error, etc.) before the UAV takes off. Then, the UAV takes off vertically until it reaches one kilometer of altitude. Afterwards, once it has reached that altitude, it will fly in a straight line towards its destination at a custom target speed. Once it is 100 meters away from its departure location, the UAV is considered to have begun its flight. If an attack is planned for this flight, a countdown of at least 100 seconds starts before the attack begins. Once the UAV reaches its destination (or believes it has reached its destination in the cases where there was an attack) it hovers at one kilometer of altitude until the simulation is stopped.

The parameters are entered before departure so that during the vertical ascent part of the takeoff, their high-level effects on the flight can be assessed. One of the main things assessed is whether they will probably lead to the UAV crashing during the mission.

The vertical takeoff until a cruising altitude of 1000 meters reduces the risk of a mountain or other elevation in the way of the flight. This can be difficult to verify for long flights without spending a long time verifying topographical maps.

The assumption that the flight only begins once a 1000-meter altitude has been reached and then stays for its whole duration at that altitude is done to concentrate on spoofing regarding longitude and latitude.

The flight is only considered to have started once the UAV is about 100 meters away from its departure location because it should have reached its target speed at that point. This provides a convenient way of having data where the UAV is only going at its intended speed and avoids the need to consider differing flight speeds in the early parts of the flight. In general, the early parts of the flights are used to train in-flight defences.

The decision to give at least a 100 seconds-long buffer between the beginning of a flight and the beginning of an attack is due to the need for many defences to do some training during the flight. To accomplish that training, they require benign flight data, and 100 seconds was proposed in an article [12] as the duration for which the training data should be collected which led to this delay being used here as well.

Once the UAV has reached its destination, hovers at 1000 meters above the ground until the simulation is manually ended. It was decided the UAV should not land to reduce any risks of altitude affecting the results.

Thus, the general flight procedure consists in entering the parameter values, taking off vertically until reaching the 1000-meter mark and, flying horizontally towards the destination where the drone will hover at that same altitude while, when the drone is 100 meters out from its departure location, a countdown of at least 100 seconds begins before the attack may take place, should there be one for this flight.

3.1.2 Attack Simulation

Once a flight has crossed the threshold of departure set in the previous subsection (3.1.1) and the countdown activated at that point has reached zero, the attack begins. It consists in substituting the GNSS positions the UAV receives for others, fake ones. This substitution happens when the GNSS position is received by the UAV but before it is used by other subsystems such as the IMU to ensure only the modified data is used.

The process for doing so is the following. First, the current actual GNSS position is saved, while calculations are made based upon it and the spoofing method's parameters to decide on the current spoofed position. Once this is done, the current spoofed position is relayed to the rest of the UAV's systems, as well as a false heading consisting of the movement between the current spoofed position and the previous spoofed position (if there was no previous spoofed position, such as at

the very beginning of the attack, then the previous real position is used). If there is no attack taking place, the current spoofed position is simply considered to be the same as the UAV's actual position.

Afterwards, the following are logged in “spoofing logs” for use in the next steps of this project: the current real position, the current spoofed position, whether the UAV is considered to have departed (using the criteria explained in the previous subsection), whether the UAV has arrived at its destination and whether the spoofing has started. The information regarding the parameters of the attack is also logged, though not on the UAV directly, instead being on a separate file for each type of attack, with information including the start of the attack's timestamp and attack-relevant parameters (angle of deviation, modification of speed, teleportation destination, etc.) as well as some complementary information which, although unneeded for attack implementation, can be useful for analysis (teleportation coordinates when using distance along the trajectory instead).

3.2 Defence Implementation and Result Collection

3.2.1 Implementation Per Flight

All the defence methods studied are based on in-flight learning, as such, the general process of training the defence during a flight is the same, no matter the flight conditions or the defence applied. When applying a defence with a given value for each hyperparameter on a specific flight, the procedure followed is relatively simple.

First, the “spoofing logs” discussed in Section 3.1 are read for that flight. Then, the logs that will be used as input for the defence are read as well. Should the inputs be composed of multiple logs (e.g., both GNSS and IMU), they are fused to ensure the one-to-one association between their entries and those in the spoofing logs. This fusing can be done each time a defence has to be tested on a certain flight (during a real flight the most recent entry of each log needed would be used). However, in the current case, where the defences are applied on the data once the flights are already finished, it is more efficient to simply fuse the logs and then apply a series of defences on these fused logs.

After the logs' reading and eventual fusion, all the entries in the logs from before the start of the flight indicated in the spoofing logs, are removed. This way, the methods concentrate on the part of the flight where the UAV has reached its cruising speed. Then, the entries corresponding to the

first 100 seconds are used to train the defence, though obviously only if this defence has a training phase. The specificities for the defences' trainings will be explained in more detail in the defences' corresponding sections of Chapter 5. An important note is that some defences use a part of their training data to also find the best values for detection thresholds and other similar tasks. In such a case, the parameterization dataset used for this last task consists of the last 20 seconds of flight of the 100 seconds making up the training data and importantly, this last part is not used when training the method. In that case, the method's training thus uses only the first 80 seconds while the best values for the threshold and other similar parameters are found using the next 20 seconds.

Once the training has been done (if any is needed) the defence is applied to each successive entry in the logs, taking into account previous entries if necessary. Should the defence signal the entry as an attack, the corresponding entry in the spoofing logs is checked to see if that is true. If it is, it is noted as a successful detection, while if it is a false alarm, it is noted as such.

Finally, once all the log entries have been evaluated by the defence, the first entry signaled as an attack is noted, including the real latitude and longitude of the UAV at that point. The number of entries that were accurately singled out as attacks is calculated, while the number of benign entries falsely considered attacks is also noted, as well as the total number of spoofed entries (those where the UAV is under attack) contained in the logs.

This is repeated for every flight and every possible set of values of hyperparameters of every defence. In this way, this procedure tries to replicate the normal operation of a UAV, where each successive log entry corresponds to new information about the state of the drone that is received by the defence, then analysed and, if considered to be an attack, reported as such.

3.2.2 Evaluation Metric

Multiple metrics were considered for the task of measuring the success of the defence and parameter values combinations, however, many amongst these suffered from some problems that made them unsuitable. They will be detailed here, noting for each what they entailed and where they were lacking, progressively heading towards the one that was actually used.

The first metric considered consists in seeing, for a flight, if the defence signalled an attack at any point and to then check whether there had actually been one during the flight. A false positive is an alarm raised during a benign flight (a flight where no attack would take place), while a true

positive happens when an alarm was raised during a flight that does encounter an attack, irrespective of when the alarm was raised compared to the start of the attack. A true negative is a benign flight where an alarm was not raised, while a false negative is one where no attack was detected even though one took place. Once these four categories have been established, a metric such as the number of correct classifications (number of true positives and true negatives) is calculated over each flight. It can then be aggregated, and the results obtained used as comparison measures.

This metric suffers from two major drawbacks, the first of which is that it completely ignores the question of timing. Indeed, suppose a defence signals that a flight is under attack, and that is indeed the case. There is no difference in score between a method that signals before the actual attack, when the drone is clearly not yet under attack, and one that signals afterwards, when it is.

A modified version is one that considers a false positive to be either an alarm raised during a benign flight, or one raised during a flight under attack before the attack takes place, while the true positives are those for flights under attack where the signals were made once the attack has already started. This makes it possible to create a different version that can hopefully be able to solve the first problem.

This modified version, however, cannot handle the second problem encountered, which is that the overwhelming majority of defences report attacks during the flights, even the benign ones. Thus, there is a need for another method to better distinguish between defences who mistakenly report an attack during the last 10 seconds of a benign flight compared to those that do the same in the first 10 seconds.

A more comprehensive version of the previous metrics can be used, keeping the idea of false positives and negatives, but instead of being a single comparison per flight (whether an attack was detected during this flight compared to whether an attack took place during it), this check would be done for every datapoint within the flight. Thus, a true positive would be when the defence signals an attack for a specific log entry in which the UAV's position had indeed been falsified, a false positive would be when the defence would report the position relayed by the log entry to be fake even though it is not, a true negative when no attack is signaled and the UAV is not under attack at that time, while a false negative would be when the defence does not signal an attack even

though it should. The results for a flight would then be aggregated together into a singular metric which would in turn also be aggregated for comparison with other defences.

This idea suffers from another problem however, which is related to the very nature of an attack. An attack always starts by implying, through fake coordinates, that the UAV is at a different position than it believes itself to be in and progressively induces a difference between the UAV's real and believed positions. The faster an attack is detected, the less time an attacker has to affect the UAV's actions. This makes speedy detection highly important, something the current metric does not take into account, rewarding in the same way a defence no matter when it detects an attack compared to the attack's start.

An obvious way to correct this is to use a metric that would instead consider the difference between the time when an attack starts and the time when it is first detected, better taking into account how much deviation there has been during the time where the UAV was unaware of an attack. This idea of difference could also be applied, inversely, to false alarms. The difference could then be between the moment when an attack is erroneously reported and the end of the flight. This is displayed in the next equation, with the difference for a flight i , $Diff(i)$, being either the distance between its first detection, $Det(i)$, and its end, $End(i)$, if there is no attack or between the detection and the beginning of the attack, $Atk(i)$, if there is an attack. Of note is that if there is no attack, then the attack is considered to happen at the same time as the end of the flight.

$$Diff(i) = \begin{cases} Dist(Det(i), End(i)) & \text{if } Det(i) \text{ before } Atk(i) \\ Dist(Det(i), Atk(i)) & \text{otherwise} \end{cases} \quad (3.2)$$

In this way, it is possible to define a metric that explains the efficiency of a flight with a simple axis: the higher it is, the worse the defence is on that flight. However, this formula has some problems, notably that the actual maximum value it can reach depends on multiple flight-dependent factors, notably the length of the flight, how far into the flight an attack took place, and others. This is a problem when trying to aggregate results from different flights with varying distances, durations, etc. A simple way to solve this is to divide the value in the formula by a flight-dependent one. In the case where an attack is reported after it has begun, the metric would divide the previous result by the difference between the start of the attack and the end of the flight. In the case where there is no attack, the value would be divided by the difference between start and end of the flight, while in the case where an attack is signaled before it actually happens, the previous result would

be divided by the difference between the start of the flight and the end of the flight, as if there had been no attack. If the divider is equal to zero, then the result of the calculation is considered zero as well.

$$Diff(i) = \begin{cases} \frac{Dist(Det(i), End(i))}{Dist(Begin(i), End(i))} & \text{if } Det(i) \text{ before } Atk(i) \\ \frac{Dist(Det(i), Atk(i))}{Dist(Atk(i), End(i))} & \text{otherwise} \end{cases} \quad (3.3)$$

The question that remains is how to define the difference between two points in the flight, or rather, which of the multiple possible definitions to use. There are two main definitions that could be used, these being geographical distance between the two points and the time difference between when the UAV was at the first position and when it was at the second one. As an example, in the case of a flight without attack, the two options would be the physical distance between the position of the UAV when an attack was reported and the destination, while the time difference would simply be the time between the false alarm and the UAV's arrival at destination.

Though they are both potentially interesting, the choice was made to focus on the use of distance as a measure of difference, as it offered some advantages. Notably, it can better relay the magnitude of a spoofing attack, as detecting an attack before it accelerates the UAV compared to after would have an impact on how far the UAV deviated before detection that would not be accurately reflected using time-based differences.

For all points (Begin(i), Det(i), Atk(i), End(i)), it was decided to use the real position except for End(i) in false alarm cases where it would use the final spoofed UAV position instead. The choice of using the real positions for Begin(i) was obvious as there is no fake position for it (spoofing does not start before the flight). For Det(i) and Atk(i), using the real locations for both is better because it more accurately represents how much the spoofer was able to make the UAV deviate from its trajectory than the fake ones whose distance can be manipulated by the spoofer and because using a mixture of real and fake positions would rapidly become a problem, one needs only think of the case of teleportation attacks (where the spoofer makes the UAV believe it moved an enormous distance almost instantaneously).

However, for End(i), the choice is more complex. In a flight without attacks, it is quite obvious as in that case both possibilities are the same. In a flight with attacks, however, it is not. If there is a

false alarm, then using the real end position would run contrary to the idea of treating that case as though it were a false alarm on an attack-less flight as that real end position depends on the spoofer who should be ignored (while the fake end position is always the destination). It could also lead to cases where an earlier false alarm gives a better score than a later one thus the fake position should be used (imagine a flight where the UAV goes backwards because of the spoofing). In cases without false alarms however, it is preferable to use the real end position as using the fake one could lead to cases where the distance between $Det(i)$ and $Atk(i)$ is greater than the distance between $Atk(i)$ and the fake end position (if the spoofer makes the UAV overshoot its destination by more than the distance between the start of the attack and the destination and the defence does not notice until the very end).

The choice was finally made to reverse the orientation of the results, with zero being the worst result and one the best, simply for the sake of easier readability and interpretation, which resulted in the following metric.

$$Diff(i) = \begin{cases} 1 - \frac{Dist(Det(i), End(i))}{Dist(Begin(i), End(i))} & \text{if } Det(i) \text{ before } Atk(i) \\ 1 - \frac{Dist(Det(i), Atk(i))}{Dist(Atk(i), End(i))} & \text{otherwise} \end{cases} \quad (3.4)$$

With the metric finally settled, there remains the question of how to obtain the results for a flight and then how to compare them, subjects that will be covered in the next subsections.

3.2.3 Results Collection

Each defence was tested in an identical way. A set of possible values for each of its hyperparameters were decided, and the defence was applied on every flight for all combinations of possible hyperparameter values, unless specified otherwise. In this way, results were collected for every flight with every possible combination of hyperparameter values for the defences.

Once this was done, the next steps are the comparison phase for each defence. It consists in choosing the combination of hyperparameter values that produces the best results overall, before moving on to the next one, which is the comparison with the other defences. Both steps require ways to compare the results from multiple flights with other similar sets of results.

The methodology for the comparison will be explained in the next subsection (3.2.4), as it pertains both to the intra-defence comparison based upon differing hyperparameters as well as to the inter-defence comparison that will be used to discern which defence is best. For this subsection, the main point is to state that the set of parameter values with the highest result score will be used to compare this defence with others.

3.2.4 Comparison Methodology

At this stage, the results collected are composed of a specific score per flight for each defence. This means that comparing two defences would require comparing all these scores and try to deduce from this series of comparisons whether a certain defence is better than another. A far simpler way would be to aggregate a defence's results into a single score and compare this new value to other defences' similarly calculated results. In the rest of this work, the aggregation technique used to do so is a simple mean of the metric discussed in Subsection 3.2.2.

Though some possible defences might be able to learn from some flights and then be applied onto others, no such method was tested here, thus a separation was unnecessary. In a similar way, splitting the dataset in half to compare the defences with one half and then apply the highest-ranked defence on the other half would permit obtaining a better estimation of the best defence's results on unseen data. This however would come as a detriment to the main goal of this project, which was to rank the existing defences, by reducing the size of the comparison dataset. Thus, the choice was made to use the full dataset for the comparison and not partition it as the possible reasons to do so did not prove compelling in the current situation.

CHAPTER 4 FLIGHT SIMULATION AND DATA COLLECTION

4.1 Motivations for a New Dataset

To compare the different defences, it is first necessary to have a common set of flights where they are applied to see their reactions in identical situations and thus enable an accurate comparison. Using a dataset of flight traces permits doing so in a far more efficient way than having a UAV repeat every different flight plan for each defence. It also helps ensure a better comparison, as the flight conditions are completely identical for two different defences, as well as making it easier to use this method to compare future defences with the ones worked on in this project. This, however, obviously requires a dataset of flight logs. However, the existing datasets are often limited, whether in terms of number of flights, types of attacks depicted, or data contained within, leading to the need for a new one.

Amongst these datasets, the UAV Attack Dataset by Whelan [40] contains extensive logs related to multiple components of the UAV (GNSS, IMU, etc.). That being said, it is only composed of two flights and the UAV is under attack in only one of these. This scarcity of flights makes it difficult to properly compare and evaluate the defences. Another dataset, the Perdet dataset [41], also suffers from these issues, as it also only contains one flight, which is not enough for a complete comparison of multiple defences.

The TEXBAT dataset [42], on the other hand, while providing more attack scenarios, has three drawbacks. The first is that it is only composed of the raw GNSS signals (there are no internal logs describing how the UAV reacted during these flights). This problem could at first glance be solved by “playing back” these recordings to a UAV and seeing how it reacts. This solution fails, however, when one considers the fact that the UAV “listening” to these recordings might react differently than the one who recorded them in the first place, thus potentially making the signals wrong. The second limitation of this dataset is that its data corresponds to stationary, hovering flights, a situation which is quite different from the one generally considered by most defence methods. A comparison using these flights is less interesting as it is not the intended use case for most of the defences. Besides those, another problem is the low number of flights (eight). These drawbacks also make this dataset unsuitable for the current project.

Thus, the existing datasets are not suited for the assessment and comparison of GNSS defences and the conception of a new one is required, which should contain more flights and more varied attacks than the previous ones, as well as enough non-GNSS data to enable defences to work.

4.2 Simulators or Real Flights

There are two main options that can be considered for the creation of a new dataset, either using a real UAV or simulating it. The first option is to use a real UAV, have it fly a certain number of missions and use GNSS spoofing in some of them. This option has quite a few advantages over the simulation proposal, its most notable being that it is closer to the final goal: deploying these methods on an actual drone.

This method however has major drawbacks. The first of them is that there are serious limitations on real flights compared to simulated ones in relation to flight length. Real flights require that the flight plan be achievable by the UAV with regards to battery life, thus limiting the length of possible flights and diminishing the applicability of this dataset, and thus the results of the comparison, for UAVs with greater range.

Another problem on this front is related to the flight conditions. There is far less control over flight conditions when flying in the real world with a real UAV compared to a simulation, both with regards to the environment and the UAV's sensors themselves, reducing the range of possible scenarios. As an example of the former, there is no way to control the wind's speed or direction during a real flight. This obviously limits the ability to represent diverse flight conditions, lessening the breadth of real-life scenarios where the dataset's lessons could be applied.

In the same vein, there is the problem of flight location. Flying with a real drone implies taking off from somewhere, and thus the presence of the UAV at said somewhere. If the goal is to obtain data from multiple locations on Earth, this would require quite some traveling.

A fourth problem is related to the logistical challenges involved with collecting the data needed. These challenges notably consist of the limited time window to operate the drone, as it requires access to the premises where it is to be flown as well as constraints regarding the meteorological conditions under which it can be flown.

A fifth problem is the possible requirements of legal authorizations to proceed with spoofing experiments as tampering with GNSS signals may be illegal. These requirements would only add more time constraints onto the dataset creation.

Thus, while using real flights does come with advantages, notably the greater fidelity to real-world use cases, it also implies multiple drawbacks regarding flight lengths, conditions, and feasibility, amongst others.

Simulators are the other option to create the flight dataset, flying the missions within the simulated environment and collecting the flight logs. Though this option has the drawback of lessening the fidelity of the flight logs and consequently the defence comparison, it does possess multiple advantages over real-world flights.

The first advantage is that most of the problems associated with real-world flights are negated by a simulation. There are no concerns regarding the possible length of the flight, its departure location, or the conditions of the flight as all of these can be adjusted within the simulation itself, enabling a far more diverse series of flights and thus a better dataset than what would be possible under real-world conditions.

Simulations also resolve some of the more practical concerns with real-world flights, these being access to the premises and weather that could impede flights. Another question that simulated flights solve is the legal question, as there are no concerns regarding GNSS spoofing in a simulated environment.

Simulations also provide the benefit of being able to execute at a speed faster than real-life, making data generation more time efficient. It also makes the testing of spoofing methods less time-consuming.

After comparing these two options, the choice was made that, though a simulation's fidelity to reality was more limited than real-life flights, its other advantages made up for this concern and would enable a better end result. Thus, simulation was the method used to create the dataset.

4.3 Simulator Selection

Once the choice of simulators over real flights had been made, the different options of simulators had to be compared. Through the same process, a choice of operating system for the UAV connected to the simulator also had to be made. The requirements on these operating systems and

simulators will be laid out in a first subsection which will help in making the choice of setup in the second one clearer.

4.3.1 Requirements

The following requirements for the simulator and UAV operating system were identified.

The first requirement is the ability to specify a flight destination using geographic coordinates instead of having to click on a specific point on a map. This makes reproducing flights far easier, as one would otherwise have to pray that the destination chosen by the click is the same one as in previous attempts at that same flight.

A second requirement is that the flights can be over a long distance, at least several kilometers. Otherwise, most of the methods studied do not have the time to complete their training before the mission ends. Of note is that this requirement is linked to the previous one in the sense that the user interface must facilitate planning flights over these distances.

A third requirement is the realism of the simulation. This requirement essentially constitutes the need for the simulation to be as similar as possible to the real world, mostly in the form of noise on sensors and imprecisions in the UAV's movement. Without this, the drone's future position would be predictable with extremely high or even perfect accuracy and would render the attempts at spoofing useless as they could be immediately detected.

A fourth requirement is the ability to change the environment in which the drone will evolve, to increase or decrease the wind's strength, change its direction, modify the amount of noise in certain sensors, or change the flight's departure location. This enables testing of defences under a far wider range of conditions and thus improve the results obtained by making them more representative of real life.

4.3.2 Setups Tested

The earliest attempt was made with PX4 [43] as the autopilot operating system and Gazebo [44] as the associated simulator as these are well-known open-source systems. However, they were lacking in many aspects that were needed for the dataset. Notable amongst these were the lack of both documentation and focus on the command-line interface and programmable missions, as the focus of the system seemed to be towards a graphical interface and manually placing targets on a zoomed-in map. These two factors were problems as they made reproducing specific missions far more

difficult, as well as making longer-ranged mission harder to program. On top of these problems, the method of GNSS simulation was unclear and finding documentation and explanations proved complex. Multiple attempts to use it were made but all failed, as such, all of these problems combined made using this setup quite unappealing as it did not fulfill even the first requirement mentioned previously.

The second setup tested was the Ardupilot [45] open-source operating system, along with its companion simulator (which is bundled with it). In the rest of this subsection, this setup will be called the Ardupilot-only setup. Missions could be given using the command-line, specifying the destination and speed, and this seemed to be the primary interface, thus it had ample documentation explaining how it worked which proved useful in preparing the missions. It also had a wide range of parameters for the simulation regarding the wind, the measurements' uncertainties and others which made it quite interesting. However, before settling on it, some other attempts were made to link Ardupilot (the operating system) with another more well-known simulator to more fully fulfill the realism requirement.

While Gazebo was also tested in conjunction with Ardupilot, this combination suffered from some problems that made it unsuitable for the current use case. Chief amongst these was the management of wind, which the pairing could not handle. The wind settings in Ardupilot's console would not affect the Gazebo simulation, and the Gazebo simulation provided did not include a wind module. Adding a wind module to this simulation was attempted, however, it unfortunately seemed to fail as the behaviour of the drone when the wind was applied was incompatible with real-life actions. This was on top of the fact that some parts of its behaviour also seemed incongruous even without adding the wind. Consequently, as these problems were becoming more obvious, the flights that had been attempted with it had to be redone using the Ardupilot-only setup as the realism criteria was not being met in this case as well.

Thus, in the end no setup apart from Ardupilot-only was fitting for the dataset's creation. It was thus the one used as it proved both easy to handle and able to modify an important range of the flight's factors.

4.4 Dataset Conception

Once the simulator has been chosen, the actual flights' missions and their conditions must be planned out to capture the full diversity of possible missions.

4.4.1 Missions

The UAV's mission is characterized by its departure location as well as the distance of the flight, its direction, its destination, and its speed. These variables were modified in different ways for each flight in order to cover most of the possible combinations.

The flights' departure locations were spread across the globe, so as to encompass more of the possible locations of flights and to ensure that, for defences relying on the geographical coordinates obtained through GNSS, should the effectiveness be affected by the location, it could be noticed when applying the defence on the dataset.

The requirements on the flight distances were that they would take more than one hundred seconds for the drones to finish the flight as this duration is the one used to obtain the training data during in-flight learning. Varied values were used for this parameter, so that should the effectiveness of a defence depend on the distance traveled during the flight, it could be noticed.

The flight's heading was also varied from flight to flight. This was done to ensure, as with departure and length, that, should a defence's effectiveness vary based upon the direction, when it is applied onto the dataset it can be detected.

The flight's intended speed (i.e., the one it was ordered to go at, which is not necessarily the same as the one it actually flew at) was modified between flights. This was done to see if it could have an impact on the accuracy of the defences, particularly its potential interplay with the flight conditions, such as the wind or the sensors' precisions.

4.4.2 Flight Conditions

The flight conditions were also varied over the whole series of flights to portray a more diverse range of scenarios. There were three main sets of parameters, these being those regarding the wind, the sensors and the GNSS signals.

For the wind, there are three distinct parameters, with the first representing the speed of the wind during the flight. The second one represents the direction of the wind's source while the third is the amount of variation in the wind's speed. These three parameters make it possible to affect the UAV with different winds, whether these be calm headwinds slightly slowing it, stronger tailwinds reducing the amount of power it requires or turbulent side winds that force the drone to often correct its direction.

For the sensors, the model UAV is equipped with barometers, gyroscopes, accelerometers, and magnetometers. Each of these has an associated randomness variable that defines the amount of noise within. These values are modified from flight to flight to create a broader range of possible situations.

A third set of parameters is composed of the noise in GNSS signals. Though these are divided by coordinate axis, they were all set to an identical value, which changed between flights. This was done to better concentrate on how the defences would manage more uncertain GNSS positions.

4.4.3 Plane or Copter

The simulator-autopilot combination chosen provides multiple possible UAV to simulate. It is possible to simulate a plane or a copter [45]. The choice was made to use a copter, more specifically a quadcopter, instead of a plane because it is simpler to recreate flights with a copter compared to a plane. This is because a copter can takeoff vertically and hover, unlike a plane.

4.5 Types of Attacks

One of the problems with the existing datasets is that they are often lacking in number of flights logs, which necessarily leads to a limited quantity of attacks recorded. To further compound these problems, these attacks often consist of a “teleportation” style of attack, where the UAV is told it has moved an enormous distance within a short amount of time [41]. Though these attacks are interesting to test the accuracy of defences, more sophisticated and subtle ones are necessary to fully assess said accuracy.

There are two kinds of falsified signals. The first is made up of those that claim the UAV moved in a different manner than what it actually did such as a “teleportation” attack. The second kind is composed of those that simply reproduce the UAV’s real movements but still tell the UAV it is in a different position than where it truly is, a mirroring of its movements. This is the spoofing that happens after, for example a teleportation attack, where the UAV’s real movements are accurately reflected by the new signals, but with the offset associated with the teleportation. The next sections will cover the possible variations of the first kind of attack, where the UAV’s real movements and the ones implied by the faked GNSS signals are actually different .

To better define the different possible attacks, some definitions could help in enhancing the clarity of the explanations. Let B (beginning) be the position of the drone when it receives a GNSS signal,

F (final) the position of the drone when it next receives its updated position (i.e., the expected position of the drone), S (spoofed) the GNSS location that the drone receives when he reaches F (without spoofing, F and S are the same) and D (destination) the drone's destination. Thus, when there are no interferences, BF is the trajectory between B and F, the actual one followed by the drone, BS is the trajectory that the spoofer claims the UAV followed, FS is the divergence implied by the spoofer, and FBS is the angle of said deviation. As well as these, FD is the real trajectory the UAV should follow to reach the destination while SD is the one it believes it should follow. For this analysis, if there is no spoofing, one can consider that F and S are identical.

It is also beneficial to define two axes of inquiry, the first being a question of movement, that concerns itself with the UAV's divergence from its expected position, while the second would be a question of perception, focusing more on the subtlety of the attack and how obvious it is.

4.5.1 Movement

The movement axis of analysis consists in studying the types of attack based upon the movement the spoofed signals claim the UAV has done. In other words, what is the change between the expected trajectory and the spoofed one (between BF and BS, so FS). There are two basic variations from this point of view, a variation in angle and a variation in distance as well as a composite of the two.

The variation in angle, a deviation or heading attack, consists of the cases where the spoofer claims the UAV simply deviated from its trajectory, in other words, when BF and BS are of the same length, but FBS is a non-zero angle (i.e., the drone drifted to the left or to the right of its intended trajectory). Such an attack would cause the drone to try and correct its position by responding to the deviation and, should the attack be repeated multiple times, cause the drone to deviate out of its trajectory, trying to correct for a deviation that does not actually exist.

This type of attack is illustrated in Figure 4.1 (drawn, like all other figures in this thesis, with matplotlib [46]). In it, the flight began in the top left corner, heading to the bottom right one (the flight was heading southeast). The blue line represents the initial trajectory the UAV followed, before the attack, the red trajectory the one followed once the attack began, the green one the one it was actually following. The black cross is the intended destination, the black Y the beginning of the flight and the black X the position the UAV actually stopped at. While this flight began normally, with the UAV heading towards its destination with the blue trajectory, a divergence soon

took place when the attack began. At that point, though the UAV believed it was following the red line, a continuation of the blue line with some small hiccups that it rapidly corrected for, it was actually following the green one and ended up at a different place than its destination.

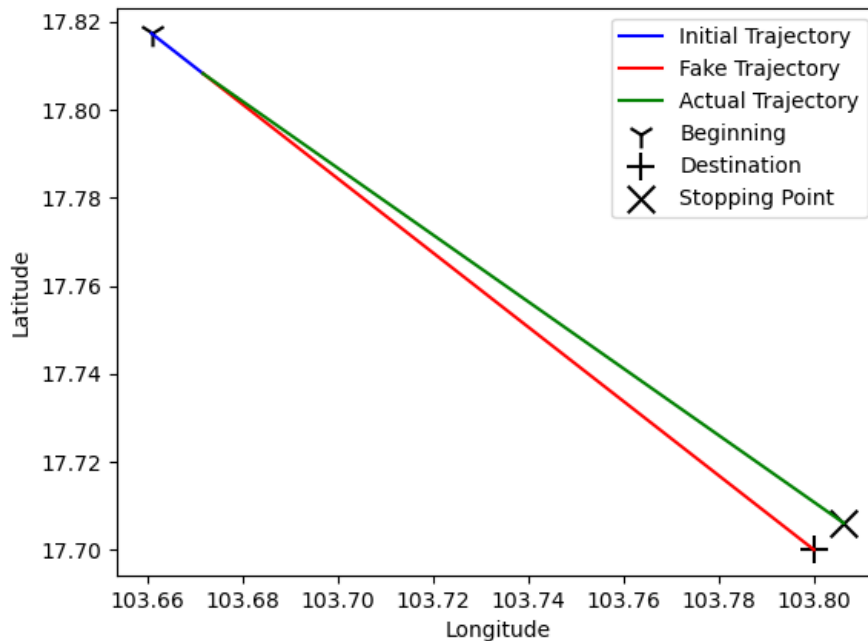


Figure 4.1 Deviation Attack (Flight 85)

This attack was implemented with a few parameters, these being the angle offset by which to modify the trajectory, the frequency at which this modification should be repeated, when the modifications should begin and when they should stop, the remaining trajectory becoming a simple mirroring of the UAV's movements. These parameters together enabled some interesting types of attacks to be added to the dataset.

The distance (or speed) attacks consist of the ones where the spoofer, instead of claiming that the UAV deviated from its intended trajectory, instead overshoot or undershot its expected location, going faster or slower than expected, respectively, in other words, B, F and S are aligned (FBS is zero), but BS is not the same distance as BF. This attack would lead the UAV, at a most basic level, to believe it has reached its destination potentially too early or too late, depending on if it thought it was going faster or slower than reality, respectively. However, should its initial orders be to head towards its destination while maintaining a certain target speed, the UAV could also be led to

believe that it is not going at its intended speed. This could lead it to correct for the perceived speed problem by either accelerating or decelerating, depending on if it thought it was going too slow or too fast respectively, which would add another growing level of divergence between its actual position and the one it believes itself to be in.

This attack can be repeated multiple times consecutively to enhance its effects and this was done with a certain set of parameters, these being the moment at which the attack starts, when a multiplier starts being applied to the UAV's speed, this multiplier changing at every set amount of time with a certain increment (both of these being parameters), to reach a first target value (also a parameter), then potentially changing towards a second target value with another set amount of time and set value to change the multiplier by (the target value, amount of time per change and value per change all also parameters). This provides a great deal of control over the attack's effects.

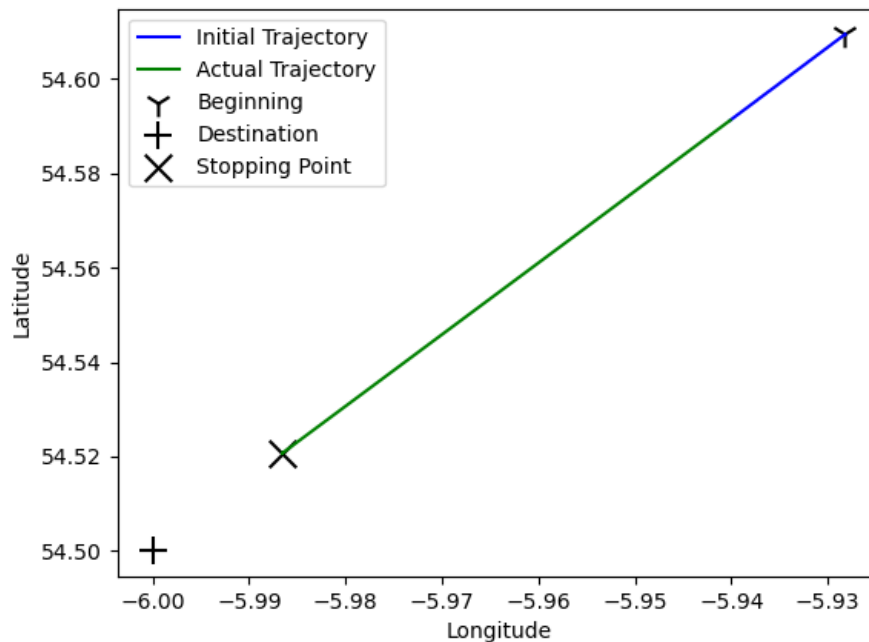


Figure 4.2 Speed Attack (Flight 64)

Figure 4.2 presents an example of this attack. As before, the blue line is the initial trajectory before the attack while the green one is the one the UAV followed after the attack (the one it actually followed, not the one it believed it followed), with the black cross being the flight's destination. At the beginning the UAV flew towards its destination, from the top right to the bottom left, following

the blue line but, an attack soon took place that caused the UAV to slow down. It then continued onwards on its trajectory, following the green line. However, it believed it had reached its destination too soon, and thus actually stopped on its way.

The composite (or combination) attack would naturally consist of those cases where the first two are mixed, those where the spoofed position is both not at the same distance from the initial position than the expected one but also where the spoofed position also implies a deviation from the trajectory. In other words, in this case, not only are BF and BS not of the same length, but FBS is also non-zero. This case can combine the effects of the previous two on the UAV's reaction to the spoofed signals, causing both a change of speed but also a deviation from the intended trajectory. It involves all of the parameters of the previous two, though both parts of the attack begin and end at the same time.

4.5.2 Subtlety

The subtlety axis of reflection is based on how subtle the attack is rather than on how it affects the UAV's movement. As can be guessed, it is far less formal. There are two main cases for it, the first being an obvious attack where the difference between the expected position and the spoofed one are important (where the length of BS is large, orders of magnitude greater than the length of BF). The second option is where the attack is more discrete and tries to avoid detection by having multiple small divergences cause multiple corrections which will slowly add up.

This notion of subtlety helps in specifying the attacks that can be mounted against the UAV as now, instead of focusing on the movement applied to the UAV, one can instead consider solely the size of this movement. This can be used to separate the attacks into two main categories, depending on if an attack aims to be discreet and hide itself amongst the normal movement of the UAV or not.

These unsubtle attacks are in a way akin to teleportation, as their goal is to make the UAV believe it has traveled an impossibly large distance between two successive GNSS signals. Amongst these, it is possible to differentiate between attacks that simply claim the UAV just reached its destination, those that act as though it has advanced far forward on its path and those that claim it was teleported out of its trajectory, far away from its actual location.

All of these take differing parameters, though they have in common a start time parameter that indicates when the teleportation happens. The teleportation to destination attack does not have any other parameters while the teleportation forward one also includes as a parameter the distance to travel. The teleportation out of the trajectory has the position to which the UAV is teleported as a parameter instead of the distance.

Figure 4.3 is an example of an attack teleporting the UAV out of its way. As it was heading to the southeast from the top left corner, following its initial trajectory, the UAV was attacked, making it believe it had been teleported to the bottom left. It then acted upon that information which made it veer off to the right and wildly off course, following the green line to the right even though it thought it was following the red one.

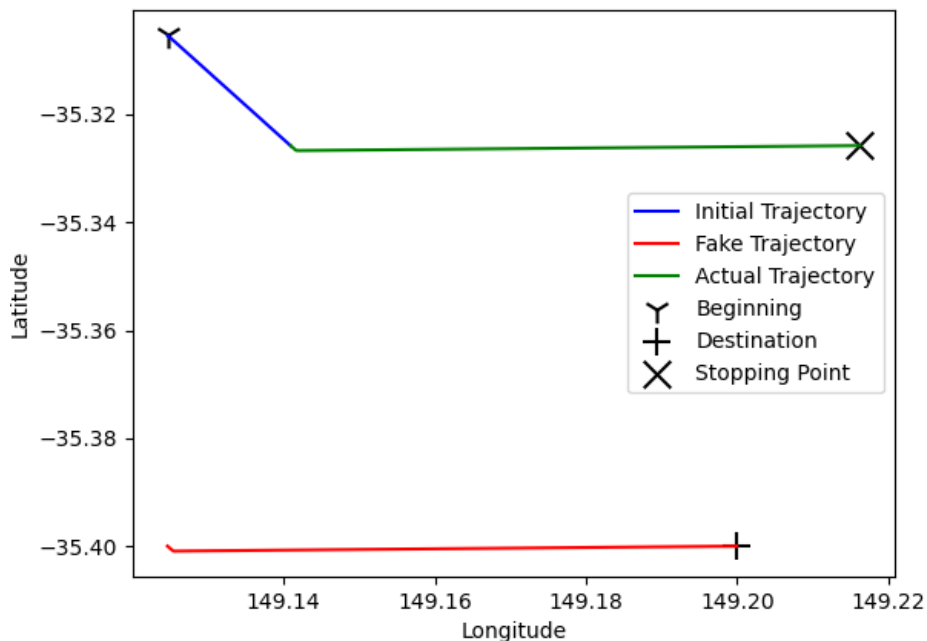


Figure 4.3 Teleportation Attack (Flight 53)

4.6 Dataset Description

Now that the dataset's theoretical conception has been explained, it is of interest to describe its content and composition. The dataset is composed of 100 flights, with flights 1 to 50 without spoofing and flights 51 to 100 with spoofing.

The 50 flights with spoofing comprise all of the types of attacks mentioned in the previous section (4.5). Of these, the first 10 consist of a spoofed signal indicating to the UAV that it has moved an enormous distance between two signals (teleportation attacks, flights 51 to 60), 20 consist of repeated spoofing telling the drones that it is going too fast or too slowly, with some of these doing one and then progressing to the other over the course of the flight (flights 61 to 80), 12 that it is deviating (flights 81 to 92) and a final 8 mixing the speed and deviation attacks (flights 93 to 100). Amongst the teleportation attacks, four consist of moving the UAV out of its path (flights 51 to 54), four to move it forward along its path (flights 55 to 58) and the final two to move it directly to its destination (flights 59 and 60). The beginning of each attack was between 100 and 500 seconds after the beginning of the corresponding flight. Figure 4.4 illustrates this distribution.

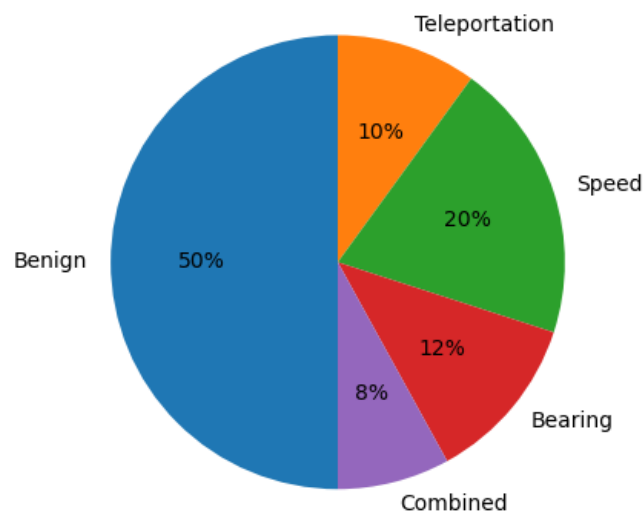


Figure 4.4: Flight Types Distribution

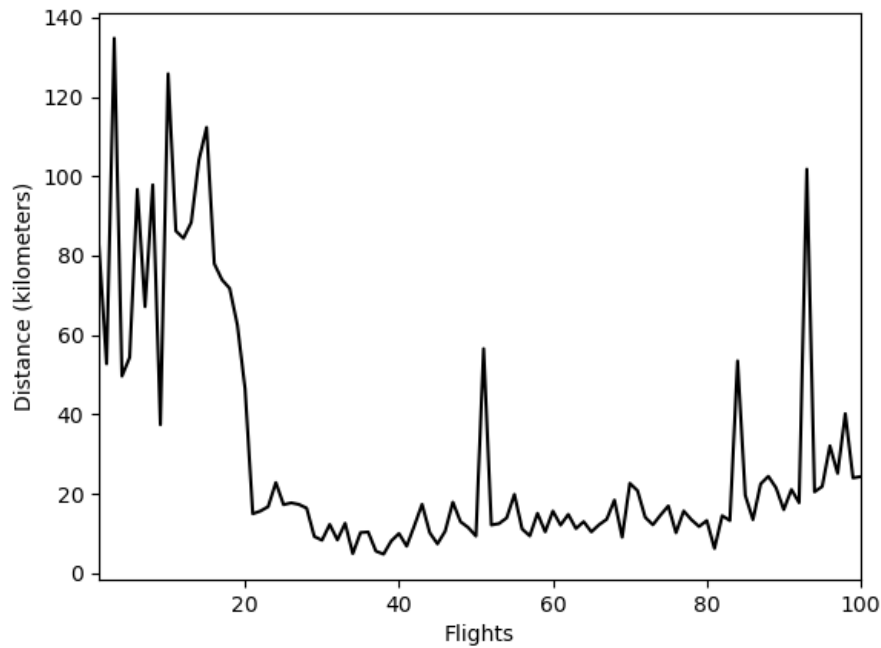


Figure 4.5 Flight Lengths

The flights use a diverse set of departure locations, these being based upon the preset list distributed alongside the autopilot, with added ones being world capitals or landmarks. These flights were generally with a speed of 10 kilometers per hour. About three quarters of those are between five and 20 kilometers in length with the rest going from 20 to 130 kilometers, as displayed in Figure 4.5.

The flight headings were spread equally between four quadrants, these being 0 to 90 degrees, 90 to 180, 180 to 270 and 270 to 360, with each quadrant being as present in the spoofed flights as in the normal ones.

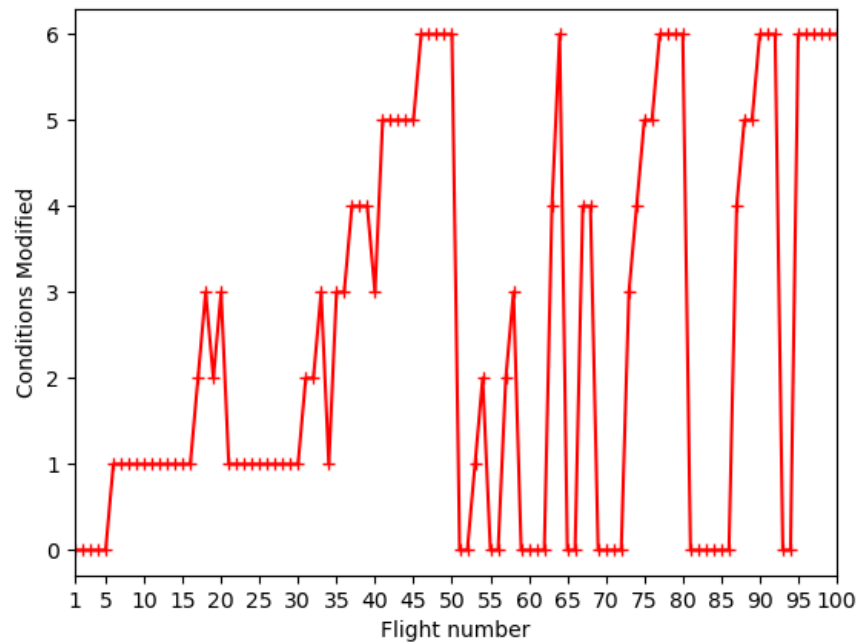


Figure 4.6: Flight Conditions

Regarding the flight conditions, for the non-spoofed flights, the first five were with no particular flight conditions, no winds or added errors on sensors, while the next ten included wind. The fifteen next ones had no wind but one or two of the sensor error parameters were modified, to try and reflect each sensor's effect on UAV positioning while the final twenty combined wind with tweaks to multiple sensor parameters. For the second half of the data, it was ensured that for each type of attack, the broad range of possible conditions would be represented. This was done by ensuring that for each type of attack, there would be flights with neither wind nor heightened sensor error as well as flights with either one or even both. Figure 4.6 serves to display the number of conditions modified, these being the sensors and the presence of wind, in a clear way.

The three tables in appendix C (Tables C.1, C.2, and C.3) provide information regarding the flights, noting for each flight all its information. This discussion of the dataset is now finished. The next chapter will cover the different defences as well as their results.

CHAPTER 5 DEFENCES ASSESSMENT

This chapter presents the different defences as well as their assessment. These defences are implemented using the scikit-learn library [47] except for the autoencoder and LSTM defences which used PyTorch [48]. An important note is that for some defences, several implementations were performed.

5.1 Implemented Defences

For some defences, the simulator setup does not simulate some of the data these defences need. This problem is present for defences using vision to handle spoofing, notably the Lucas-Kanade [32] and the road following [35] algorithms. It is also the case for the Kullback-Leibler algorithm [29] as it requires the number of GNSS satellites whose signals were received as well as for the random forest [13] and neural network [19] methods which require information on the signals' characteristics. This is also a problem for the methods using third-party cellular information [37, 38]. As all of these defences require data that is not simulated, it was not possible to implement and test them.

On the other hand, some defences require data that, though handled by the simulator, cannot be logged to be used afterwards to test the defence. This is a problem for approaches like DeepPose [30] or the road following algorithm [35] that require map information regarding the UAV's surroundings, something that, though simulated during the flight, is not logged and would be quite complex to recreate based upon the information present in the logs. This makes implementing them impractical in the current context.

Some other defences require the use of multiple UAVs, something that could not be tested. Indeed, defences like cooperative localisation [33], trilateration [34] and parent-child grouping [39] require more than one UAV for a single mission, something that was not considered here.

Also, some defences simply cannot be tested with the current setup, notably the idea of adding authentication to GNSS signals. This defence works in a wholly different way to the others implemented here, following a preventive approach instead of using a detection or mitigation one. This makes it unsuitable for testing using logs where attacks have already taken place.

A final point against some defences is that they are variations on others that are already implemented. Notably the IMU position comparison [27] and the IMU angular variation

comparison [28] are both somewhat similar to the IMU linear acceleration method [27], thus lessening the need to implement them.

The following sections present the implemented defences, by describing for each, its functioning, its hyperparameters, the different versions implemented, the results obtained, and finally discussing these results.

5.2 EKF: Extended Kalman Filter

5.2.1 Description

The first defence used was an extended Kalman filter, as the UAV autopilot came with one built-in [45]. On top of measuring the results of its reports of GNSS anomalies, some versions of that defence were made that use the on-board log outputs of the built-in EKF to also determine whether the signals were attacks. An extended Kalman filter is, as its name indicates, a variation on a Kalman filter, an approach that predicts the state of a system based upon successive noisy observations by keeping a running estimate of the current state as well as its uncertainties and updating both with information from newly obtained data.

The autopilot used to collect the flight traces had an extended Kalman filter by default to keep track of its state, including the GNSS position, and would send error messages signalling GNSS glitches during flights when it considered the GNSS signals to be erroneous, more precisely when the innovation (change between predicted value and expected one) was greater than some threshold value [45].

This naturally led to questions regarding whether it could be built upon to produce better defences. A few defences were thus constructed that would look directly at the innovation during the training phase, multiply the maximum value obtained with a multiplier hyperparameter, and then compare the innovation for the rest of the flight with this threshold. These approaches, derived from the base system implemented in the autopilot, would enable building on it without the requirement to code an actual Kalman filter and were thus a welcome addition to this project.

Different versions of this general defence were made, with the first one consisting of only the alerts sent by the UAV during the data collection phase found in the error log which would serve as a default defence that can be used to compare the other ones constructed afterwards. The other

versions applied the idea of a threshold to either position, speed, or both (with a version requiring both being too high while another only one of them) using the EKF logs.

This threshold idea consisted in using the training data to find the greatest amount of innovation in a given entry and using this maximum value, multiplied by a hyperparameter, as a threshold. Should that threshold be exceeded during the flight, the defence will signal the exceeding entry as anomalous and the sign of a spoofing attack upon the UAV.

5.2.2 Hyperparameters

There is only one true hyperparameter in these defences, the multiplier. It is used in the calculation of the threshold used as a boundary for the allowed variance in the data, boundary beyond which the UAV is considered under attack. It is used, as described previously, by multiplying it with the maximum innovation value obtained in the training phase to calculate the threshold. This adds a level of tolerance for anomalies in the defence and thus improve the results. The values tested were 1, 1.25, 1.5, 1.75, 2, 3, 5, 10, 15, 20 and 50. These values were chosen to cover a wide range of possible cases. This will be the case for all other hyperparameters throughout this work.

5.2.3 Implementations

Five different versions of this defence were implemented, the first of which was the onboard EKF's error log, where the GNSS glitches signaled were interpreted for the current purpose as spoofing detection. This defence provided a baseline of what the UAV currently offered that could be used for comparison with other defences.

The second version consisted in calculating the maximum position innovation during the training phase and using this value, multiplied by the multiplier hyperparameter, as the maximum threshold for any entry's position innovation during the rest of the flight. Should any entry be greater than this threshold, it would be signalled as an attack. The third version was much the same, with the only difference being the usage of the velocity innovation instead of the position one. The fourth version used both the position and velocity innovations, calculating distinct thresholds for each and, should either of the two be exceeded, the entry would be considered spoofed. The fifth was quite similar to the fourth, though it required that both thresholds be exceeded instead of only one before the position was signalled as an attack.

5.2.4 Results

For the first version, the one based solely on the UAV's error reports, the score obtained using the metric defined in Section 3.2 was 0.598491. This built-in version does not detect any errors during the fifty normal flights (flights 1 to 50) and detects almost immediately all 10 obvious attacks (51 to 60). However, it fails to detect any attack more subtle than these basic ones (61 to 100).

In Figure 5.1, the results of the first version for each flight are displayed, with the deviation being a representation of how far off from the expected detection the defence was, this expected detection being either the attack's beginning when there is an attack in the flight, or the end of the flight when there are no attacks within the flight. If the score is positive, then it means the first detection was after the expected one, while if it is negative then the detection was earlier than expected. The value used for this deviation is 1 minus the score for the specific flight. As an example, for the fifty-eighth flight, the deviation is slightly above 0, which means that this version of the defence first detected an attack slightly later than when it actually happened. For the rest of this work, a false alarm is counted when its deviation (the value on the graph) is lower than -0.25, while an attack is considered detected quite late when its deviation is greater than 0.25 and completely missed when it is close to 1.

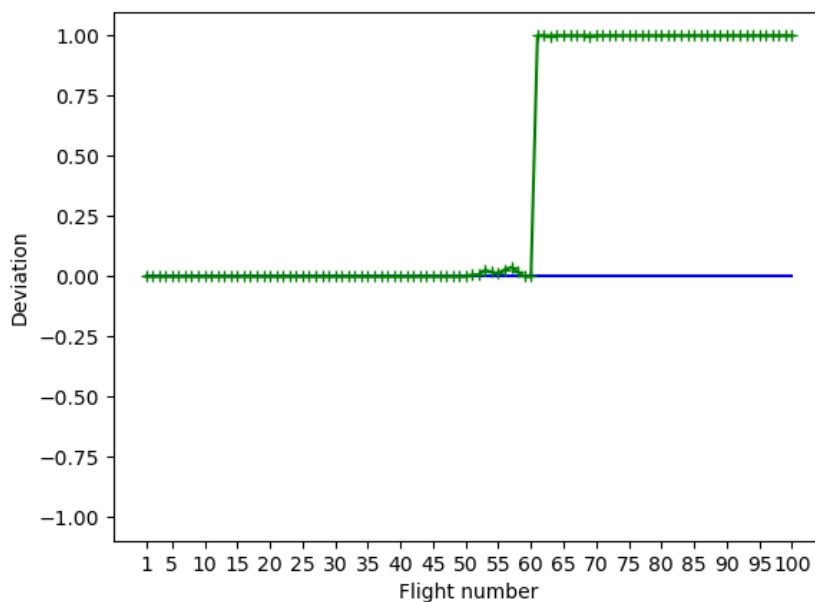


Figure 5.1 EKF defence graph, first version (baseline)

For the next versions of this defence, as well as the next defences' versions, the graphs displaying the flight-by-flight results of the top configuration are in appendix A, displaying the results of the version's configuration with the highest score (Figure A.1 for the second version of this defence, for example).

Table 5.1 EKF defence results, second version (position)

Multiplier	Score
1.5	0.751424
1.75	0.727300
2	0.704283
3	0.628272
5	0.599832

Table 5.1 presents the results of the five best hyperparameter configurations for the EKF defence's second version (each subsequent table will do the same for its corresponding version). Its top configuration falsely considers as under attack 4 normal flights, but it also correctly detected multiple attacks, detecting almost immediately all obvious attacks and rapidly 17 non-obvious ones. The attacks missed are concentrated amongst the heading and combined attacks, generally with heavily affected flight conditions.

Table 5.2 EKF defence results, third version (velocity)

Multiplier	Score
1.75	0.6880657911513459
1.5	0.6800430792694658
5.0	0.6765741066017845
3.0	0.6750412004022789
2.0	0.6718094458262434

Table 5.2 presents the results for the third version. This version incorrectly reacted (signaled an attack) during nine normal flights. It also failed to react correctly to five of the obvious flights, only detecting rapidly half of them and reporting an attack too early in one of the five misclassified ones. It however correctly detected 20 of the non-obvious ones. The attacks it missed were relatively evenly spread out between the different types of attacks and were often under heavily affected

flight conditions. The false alarms were not, however, with only one or two affected flight conditions at most.

Table 5.3 EKF defence results, fourth version (position or velocity)

Multiplier	Score
2.0	0.7394572667561298
3.0	0.7289780801731829
5.0	0.7265761518698133
1.75	0.7093742031861676
10.0	0.70329616202053

Table 5.3 presents the results for the fourth version. This version had only 12 false alarms on normal flights but failed in correctly detecting one of the obvious attacks and 18 of the non-obvious ones. Its missed attacks were amongst the flights with the heavier conditions, relatively evenly spread out between the types of attacks. Its false alarms were in flights with little in the way of modified flight conditions, however.

Table 5.4 EKF defence results, fifth version (position and velocity)

Multiplier	Score
1.25	0.7426796606823656
1.0	0.7331669666554985
1.5	0.7102363620314511
1.75	0.68603942304466
2.0	0.6521192072177387

Table 5.4 presents the results for the fifth version. This version only signaled an attack at the halfway mark of a single normal flight. It however, incorrectly assessed five of the obvious flights and only did so correctly for half (20) of the non-obvious ones. Out of the missed attacks, almost all of the subtler ones were under heavily affected flight conditions and spread out across the different types. However, the obvious attacks missed were bit under said heavy flight conditions.

5.2.5 Discussion

The results of the first version can be used as a baseline to compare with the other defences, where the acceptable minimum is to minimize the number of false alarms detected while still detecting the obvious attacks. Its score of 0.598491 can be used to assess if a defence is considered an improvement on the status-quo or not.

The second version, which uses only the position innovation, obtains better results than this baseline. It did not fail to detect a specific kind of attack, but was, in all likelihood, mostly affected by the inaccuracies caused by the flight conditions, as these missed attacks mostly took place in flights affected by multiple flight conditions. Its results make this quite an interesting approach and an improvement over the baseline. These improved results were obtained through its customized threshold computed on a flight-by-flight basis, thus allowing for more flexibility than what is possible in the previous version with one set before departure.

The third version obtains inferior results to the previous one, though still better than the baseline. It seems to have had the most difficulty with flights where there were multiple sensors, particularly the GNSS one, whose errors were increased. Its usage of velocity over position probably penalized it as it was harder for it to establish a good threshold, as evidenced by its greater amounts of false alarms.

The fourth version was the one where, should either the position innovation or the velocity one be too great an attack would be signalled. It had more false alarms on normal flights than the second version and failed in correctly detecting one of the obvious attacks and 18 of the non-obvious ones, contributing to its worse score than the second version's best attempt. It also had difficulties in some of the same areas as the previous versions, in the flights with heightened errors for the sensors. Though it improved upon the velocity version, it did not manage to fully reach the level of the position version, weighted down by the usage of velocity.

The fifth version, which would only signal an attack if both position and velocity detected one, had better results than the fourth though it was somewhat worse than the version considering only the position innovation, though better than only using the velocity innovation. It, as the others, suffered most in flights with multiple sensors affected by attacks. It also had difficulty handling the velocity innovation as it repeated some of that version's mistake, missing some of the same obvious attacks.

From these results some observations can be made, most notably that using the position innovation seems to be the best method for detecting spoofing attacks. This can be seen as, not only does the version using solely the position information with a multiplier value of 1.5 obtained better results than all other versions. Furthermore, whatever the multiplier value, most of the best variations of the fourth and fifth versions, which take into account the position innovation, perform better than the third version (which does not), though not as well as the best of the second version (which ignores the speed).

In conclusion, this defence provided both a baseline through its first version and, through its second version, an improved version that serves as the best of what can be accomplished using only the current built-in defence.

5.3 Autoencoder-based Defence

5.3.1 Description

One of the defences that has been proposed to counter GNSS spoofing against UAVs was an autoencoder [31]. An autoencoder consists of a neural network trained to recreate its input as well as possible, doing so by taking in its input, returning something of the same dimension and being trained based on the difference between its input and its output, called the reconstruction error [31].

The interesting part of this construct is that it cannot just transmit directly the information between an input node and an output node with no transformations, as there are one or more neuron layers between the input and output layers, with those intermediate layers made smaller than the input and output ones to force some compression of the input information.

This forced compression means the neural network must try and learn some of the correlations between different parts of the input. To train it, a series of inputs are provided and fed through the network. The outputs are then graded based upon their reconstruction errors, encouraging the network to learn these correlations. This is repeated for a certain number of times, said number being set before training. This training enables it to reproduce input similar to what it has been trained on. However, if the entry is different from the training data the network will have a harder time recreating this new input.

This approach can be used in anomaly detection by training an autoencoder on normal data during the training phase before any attack takes place. Once the training is done, a second part of the

data, the parameterization dataset, is used to ascertain the maximum reconstruction error when not under attack, which is then used as a threshold once the multiplier hyperparameter is applied to it. Afterwards, and for the rest of the flight, each new datapoint goes through the autoencoder and is rated based on the difference with its output. If this difference is greater than a set threshold, then it is considered anomalous and the UAV under spoofing attack. If it is below the threshold, it is ignored, and the UAV keeps flying. The data used as input is a fusion of the GNSS and IMU flight logs, with principal component analysis (PCA) potentially applied onto it before the training.

5.3.2 Hyperparameters

The autoencoder defence has multiple hyperparameters that influence its functioning as well as its success. Amongst these are the number of epochs, the size of the internal layer, the learning rate, the multiplier used to obtain the threshold and the number of features in the PCA.

The number of epochs is a proxy of the amount of training done, consisting of the number of times each entry in the training part of the flight is used for training. It is directly related to the capacity of the neural network to correctly reproduce data on which it has not been specifically trained, in other words, its capacity to reproduce data during the subsequent parts of the flight. The values tested are 10, 20, 30, 50, 75, 100, 150, 250, 500 and 1000.

The size of the internal layer consists in the amount of compression applied to the data, the smaller size implying a greater compression and inversely. This affects the capacity of the autoencoder to successfully reproduce existing data. Of note is that while it would have been possible to use multiple internal layers within the autoencoder, this was not done for simplicity's sake. The possible sizes are 1, 2, 4, 6, 8, 12 and 16.

The learning rate controls the speed at which the autoencoder learns from a training round, with a greater learning rate implying a greater impact of the current training round on the current values of the weights within the neural network. The values studied are 0.0001, 0.001, 0.01, 0.1 and 1.

Another hyperparameter is the multiplier used in the threshold calculation. This threshold is used to decide whether the reconstruction error for a particular entry is too great or not, and thus whether it should be reported as an attack. This threshold is calculated by multiplying the greatest maximal error obtained after the training with the multiplier, as described in the previous part. As can be

guessed, this multiplier acts as a controller on the defence's tolerance for anomalous behaviour. Its possible values are 1, 1.25, 1.5, 1.75, 2, 3, 5, 10, 20 and 50.

The number of features hyperparameter corresponds to the number of features extracted using PCA, which affects the amount of variance in the original data that the PCA can conserve through its process. The possible values are all integers between 2 and 8 included, as well as -1, which meant that PCA is not used.

5.3.3 Implementation

Only two versions of this defence were implemented, where the inputs were the differences between data entries, and where all previously listed hyperparameters were used as described. The difference laid in the data entries. For the first version, they consisted of a fusion of the GNSS and IMU logs, for a total of 29 different inputs to the autoencoder while for the second one they were restricted to solely the GNSS log.

5.3.4 Results

Table 5.5 Autoencoder defence results, first version (fused logs)

Multiplier	Internal Layer Size	Epochs	Learning Rate	Features	Score
5.0	8	30	0.1	-1	0.7502604887413682
50.0	2	150	1.0	6	0.7261207739536042
3.0	2	75	0.01	4	0.7225897987274021
50.0	2	500	1.0	5	0.721225846031101
20.0	2	20	1.0	7	0.7211705285950051

Table 5.5 presents the results for the first version. Its the top configuration had 22 false alarms in the benign flights, correctly detected all obvious attacks, and rapidly detected 30 of the remaining 40 subtler ones. The vast majority of false alarms were in flights with only a few affected conditions. The attacks missed, on the other hand, were in flights with heavily affected conditions, mostly facing speed attacks.

Table 5.6 Autoencoder defence results, second version (GNSS only)

Multiplier	Internal Layer Size	Epochs	Learning Rate	Features	Score
3.0	2	1000	0.001	7	0.8162065503688939
3.0	4	250	0.1	7	0.8093107438261902
3.0	2	500	0.001	7	0.8078658624312449
3.0	4	100	0.01	-1	0.8060975513137207
3.0	6	250	0.001	-1	0.8051201839884123

Table 5.6 presents the results of the second version. The top configuration had 15 false alarms in the normal flights and 7 missed subtle attacks. Most of the false alarms were in flights with only wind while the missed attacks were spread out between speed and heading attacks, though in both cases the flights had heavily affected flight conditions.

5.3.5 Discussion

For the first version, the most performing parameter combination seems to have been a somewhat moderate multiplier applied to the error rate, alongside a small number of epochs, a moderate learning rate and no prior PCA, as well as a somewhat moderate internal layer size. This differs from the rest of the top combinations, which all used a small internal layer size coupled with PCA being applied before inputs were provided to the autoencoder.

Specifically for the top combination, the results obtained are interesting, as some observations can be made from its results. Notably, it had no difficulty detecting any of the basic attacks and detected rapidly three quarters of the subtler ones mostly missing attacks with heavily impacted sensors. For the flights without attack, it had false alarms slightly less than half the time. Overall, it had perfectly identified more than half of the flights and correctly detected most of the non-obvious attacks, a good result that beat the baseline.

For the second version, the results improved significantly, passing the 0.8 threshold for the first time. The top configuration in particular managed to detect the vast majority of attacks while only having a limited number of false alarms. These improved results indicate that using the GNSS log alone may be better than adding the IMU one to it. This is probably due to the reduced size of the input data as the most likely values to be directly affected by spoofing are latitude and longitude

which have a greater impact on reconstruction when using GNSS-only compared to the fused logs as the latter have more variables per datapoint thus lessening the impact of each individual one.

Another interesting observation for this GNSS-only version is that, even though the top result uses the maximum number of epochs and PCA, the fourth is close to it but has ten times less epochs and no PCA, probably making this fourth configuration far faster to train. This trade-off is a potentially interesting one that should be kept in mind when a choice has to be made regarding implementation.

Finally, two interesting observations common to both versions can be made. The first is that the usage of PCA does not seem to particularly help obtain better results. After all, the top result of the first version does not use it and, while the top result of the second version does, the difference between it and the fourth top result, which does not, is quite small. The second is that all of the top configurations for both versions significantly outperformed the baseline, the UAV's onboard EKF with no modifications (first version of the EKF defence), indicating that this defence seems quite promising.

5.4 Linear Regression-based Defence

5.4.1 Description

Another defence used was an application of linear regression [12]. A linear regression is a technique of statistical analysis that tries to find the relation between two variables, for simplicity's sake X and Y , by considering one, Y , as a linear combination of the other, X . In other words, the first variable, X , can be multiplied by a constant factor (that does not change, no matter the value of X), and to that result a constant offset can be added to obtain Y . Though these relations cannot be found exactly, because of the impacts of noise and imprecisions in measurements, it is still possible to find the one that fits the most some dataset by minimizing the difference between the predicted values and the observed ones. The relation is displayed in equation 5.1, with A the constant factor, B the constant offset and N the noise (N changes every time X does).

$$Y = A \times X + B + N \quad (5.1)$$

This can be applied to the present project, where another defence proposed was one based on linear regression [12], in which the UAV's movement since the last datapoint is predicted based upon the time elapsed since that datapoint was evaluated. More precisely, once a new datapoint is obtained,

the changes in latitude and in longitude since the previous one are predicted based on the delay between the current datapoint and the previous ones. These predicted changes are then compared to the observed ones between the two entries in the GNSS log and, if one of the differences is greater than a threshold, an alarm is sent, signaling that the UAV is currently under attack.

The defence is trained on the beginning of the flight, where a series of datapoints, the training set, is used to calculate the delays and the associated changes in latitude and longitude. These are then used as the variables in two linear regressions, with the delay as the input variable while the changes in latitude and the changes in longitude are the output variables for the two different linear regressions. Then, either the training data itself or the parameterization dataset (a part of the training dataset kept aside) are used to determine the maximum allowable error which is then multiplied with a hyperparameter to create the detection threshold. Should the difference between the expected changes in latitude or longitude and the observed ones be greater than the threshold, then an alert is issued.

An idea for this defence was to look at more than the latest span of time between data entries, and instead look at a series of the latest timespans. In other words, to look at the time elapsed between the current entry and the previous one, but also at the time between said previous one and the one before it, and so on. This added information could help to better predict the actual movement, and thus better detect anomalies, using a multivariate linear regression. The main difference between this approach and the one described previously, with a single-entry variable, would simply be that each input variable would be multiplied by a constant but different coefficient to better reflect each one's specific impact on the actual movement.

5.4.2 Hyperparameters

There are two hyperparameters for this approach, the multiplier used in the threshold calculation and the number of previous time differences used.

The multiplier is, as in the previous defence, used to construct the threshold. The highest error obtained in the second phase is multiplied by this multiplier to obtain the actual threshold. Thus, a higher multiplier leads to a larger allowance for deviation, to help separate imprecisions in the UAV's movements from an actual attack. The possible values are 1, 1.25, 1.5, 1.75, 2, 2.5, 3, 4, 5 and 10.

The other hyperparameter is the number of previous time differences used as input variables to compute the UAV's expected movement, the *lookback*. Varying this number can affect the precision of the defence and thus its effectiveness. The values tested are between 1 and 5 included.

5.4.3 Implementation

Three different versions of this defence were implemented.

The first version uses no parameterization dataset. It exploits the maximum difference between the expected values of the changes in latitude and longitude compared to the actual ones over the training set as a threshold, with these expected values obtained by training the model on the same training set. Thus, in this version, the training set used to train the regression model is bigger than in the next version, as there is no need to set aside a part of it for parameterization here, unlike in the next one.

The second version used a parameterization dataset, thus the threshold was the maximum difference between the expected values of the changes in latitude and longitude compared to the actual ones over the parameterization set. The expected values were obtained by training the linear regression over the training set.

In the third version, no parameterization set was used, and the model was trained on the training set (including the parameterization dataset). The threshold was constructed by taking the maximum observed changes in latitude and longitude (instead of the differences between these observed changes and the predicted ones). This version is the closest to the one present in the paper that proposed this defence [12].

5.4.4 Results

Table 5.7 Linear regression defence results, first version (no parameterization)

Lookback	Multiplier	Score
5	1.75	0.7583316200872072
4	1.75	0.7583301832381794
2	1.75	0.7573637873033746
3	1.75	0.7573637873033746
1	1.75	0.7573615346836908

Table 5.7 presents the results of the first version. Its configuration with the highest score had a lookback of 5 and a multiplier of 1.75, mistakenly reported attacks on 20 of the 50 normal flights, mostly ones which only had wind, but on the other hand, it detected immediately all obvious attacks and only missed nine of the 40 non-obvious ones, spread out amongst the speed and deviation attacks, generally with heavily affected flight conditions.

Table 5.8 Linear regression defence results, second version (with parameterization)

Lookback	Multiplier	Score
5	2.5	0.7442071204573331
3	2.5	0.7395442329883146
4	2.5	0.7389322032902993
4	3.0	0.7358608153147118
5	3.0	0.7277026600837719

Table 5.8 presents the results of the second version. The configuration with the top score, with a lookback of 5 and a multiplier of 2.5, incorrectly reported attacks on 20 of the 50 normal flights, quite similar to the previous version with false alarms concentrated on flights with only wind. Though it correctly detected all the obvious attacks, it failed to identify nine of the 40 non-obvious ones and it took some time as well to detect a tenth.

Table 5.9 Linear regression defence results, third version (paper)

Lookback	Multiplier	Score
5	1.5	0.6062167639484588
1	1.5	0.606216729846215
4	1.5	0.6062165827456293
3	1.5	0.6062164164507784
2	1.5	0.6062162934502736

Table 5.9 presents the results of the third version. The top configuration of this version had two false alarms in benign flights, though near their end, and detected all obvious attacks and one non-obvious one halfway through thus missing the 39 other subtle attacks.

5.4.5 Discussion

The scores obtained in the first version indicate that the amount of lookback seems to have little effect on a method's results, with the multiplier having far more impact as all five best hyperparameter combinations have a different lookback value but the same multiplier with very little score difference. All five of these results also comfortably clear the baseline, being better than the UAV's onboard EKF's warning system. This good performance is due to the fact that this problem boils down to finding the linear relationship between the time elapsed and the changes in latitude and longitude, something linear regression is meant for.

The results of the second version are slightly worse than the ones of the first version, possibly indicating in this case that the gains from using different data to construct the threshold does not outweigh the greater amount of training data of the first version. However, the difference in scores between their respective top configurations is rather small, indicating that perhaps these impacts are not that important.

The third version obtained far worse results, however, barely beating the baseline, with a best score of around 0.6062 compared of around 0.5985 for the baseline. The reason for this is undoubtedly the change in threshold calculation which made it far more permissive as the difference between two successive datapoints, used in this third version, is far greater than the difference between the predicted changes in latitude and longitude and the actual changes, used in the previous versions. As in the first version's case, the multiplier is the same for all 5 best hyperparameter combinations, reinforcing the idea that the lookback, the other hyperparameter, does not matter much.

All of these results indicate that the first version is the better one, beating out the second by a small margin. Both of these are marked improvements over the third one, the closest to the proposal of the original paper [12]. The observation that the lookback has little impact seems confirmed not only by the fact that it is the hyperparameter that varies most, but also that when it varies, the change in score are minimal as the difference between the best and worst performing configurations displayed can attest (the biggest difference is for the second version, 0.01, while the first version's is below 0.001 and the third is below 0.000001).

5.5 LSTM (Long Short-Term Memory)-based Defence

5.5.1 Description

One of the other defences implemented was one using a long short-term memory network (LSTM). This defence consists in using a LSTM to predict the UAV's future position, its latitude, longitude, and speed, based upon its current one. It does so through the use of a LSTM network, a type of recurrent neural network that is able to learn some of the sequential patterns connecting successive entries, thus offering a certain predictive ability. A LSTM was used as it is a relatively common type of recurrent neural network and it was used as a benchmark in a study for another defence [12], thus enabling study replication as an added bonus.

This can be used as a defence in a similar way as the previous linear regression defence, by predicting the expected movement of the UAV in latitude and longitude as well as its speed, then comparing it to the actual movement and speed. If the difference between these is greater than a set threshold, then it is signalled as an attack.

The LSTM network is trained on the beginning of the flight with a set of datapoints (taken from the log data), the training set, used to create series of successive changes in latitude and longitude, as well as speed, where the last entry in a series is the one to predict while the previous ones are the ones used to predict it with their number being decided through a hyperparameter, the lookback. These samples are then all used to train the LSTM, with all series run through the network that then corrects its weights to better predict, with the size of its corrections depending on the learning rate, and with these rounds of training being repeated a set number of time, depending on the epochs hyperparameter.

The parameterization dataset is run through the network once the training is done and the differences between the predicted values and the observed ones are measured. The maximum of these differences is then multiplied with a multiplier hyperparameter to obtain the threshold value mentioned in the previous paragraph that will be used to identify attacks.

5.5.2 Hyperparameters

There are four hyperparameters used for this defence: the number of epochs during training, the multiplier, the lookback, and the learning rate, all of whom have an impact upon the defence's results. The first, the number of epochs, is used to decide on, as the name implies, the number of

epochs of the training, in other words how many times should each datapoint in the training part of the data be used to train the LSTM model, something that directly affects the results obtained through the potential of overfitting to the training data. The values tested are 10, 20, 30, 50, 75, 100, 150, 250, 500 and 1000.

The multiplier is used, as in other defences, as a multiplier applied to the worst error observed on the second part of the dataset, with the result used as the threshold for anomaly detection, enabling modifications to the allowed margin of error before an entry is declared anomalous during the rest of the flight. The possible values were 1, 1.25, 1.5, 1.75, 2, 3, 5, 10, 20 and 50.

The lookback consists, as in the previous defence, in the amount of previous datapoints taken into consideration to predict the current change in position. This hyperparameter thus plays on how much data is considered for prediction and consequently, the accuracy of those predictions. The values tested were all integers between 1 and 9 included.

The learning rate consists in how much the LSTM should learn from a round of training, with greater values implying faster change in the network while smaller ones would make those changes take longer. This is used in combination with the number of epochs to manage the risk of overfitting to the training data as well as the one of underfitting, where not enough would be learned. The values tested were 0.0001, 0.001, 0.01, 0.1 and 1.

5.5.3 Implementation

Only one version of the LSTM was implemented, in the way described in the Subsection 5.5.1. It used all four of the hyperparameters covered in Subsection 5.5.2.

5.5.4 Results

Table 5.10 LSTM defence results

Multiplier	Epochs	Lookback	Learning Rate	Score
10.0	100	1	0.0001	0.5828929557348852
10.0	75	1	0.01	0.5825727644206959
10.0	30	1	0.1	0.5825683281873306
10.0	1000	1	0.001	0.5819468473596562
10.0	50	1	0.001	0.5813801967709549

Table 5.10 presents the results of this defence. The hyperparameter combination with the best score had false alarms far from the destination on 15 of its 50 normal flights. It also had false alarms on all 10 obvious attacks before those attacks even took place. It only identified 16 of the 40 non-obvious attacks within a reasonable timeframe, as 16 others were not detected, spread between speed and deviation attacks, though all under heavily affected conditions. The 8 remaining had false alarms before their attacks could take place. Out of these 33 false alarms, the majority were with little modifications of flight conditions.

5.5.5 Discussion

The results obtained with this defence are, as noted in the paper where it was mentioned [12], worse than those obtained with the third version of the linear regression defence (the version proposed in said paper). They are also worse than the baseline. These results confirm the view expressed in the paper [12] that a LSTM is an inadequate tool for this kind of task due to its unwieldiness with linear inputs.

5.6 Linear Acceleration Defence

5.6.1 Description

The linear acceleration defence was proposed in [27]. This defence calculates the drone's acceleration in latitude, longitude, and altitude, converts it to the drone's frame of reference and then compares it with the one observed by the drone's accelerometer, using a fusion of the GNSS and IMU logs to obtain the data it needs.

This conversion is done by first calculating the acceleration from the successive GNSS positions of the drone. Then, a transformation matrix is used, with its components being updated through the UAV's gyroscope measurements before the matrix's multiplication with the previously obtained acceleration. The detailed explanation can be found within the original paper [27]. The method uses two sliding windows. The first is used to compute the average acceleration difference between the predicted and observed speed. The second is used to calculate the average, or the variance of the differences computed over the first.

Once the GNSS-obtained acceleration has been transformed to the UAV's frame of reference, the difference between it and the one obtained with the accelerometer is calculated. This difference is

calculated for every previous datapoint still in the first sliding window, with the average of these differences used afterwards.

Once this current average difference is obtained, its second element is compared to the first threshold (Epsilon 1). Should its absolute value exceed this threshold, then the average of the second elements of the average differences of each datapoint in the second sliding window will be compared to the second threshold (μ_1) and, should this threshold be exceeded, the UAV will be considered under attack, while if it is not, the UAV will not be considered under attack.

If the first threshold is not exceeded, however, a different process will be followed. The absolute value of the current average difference's first element will be compared to the third threshold (Epsilon 2) and, if this threshold is exceeded, the variance of the first elements of the average differences of each datapoint in the second sliding window will be compared to the fourth threshold (μ_2). Should this threshold also be exceeded, the UAV will be considered under attack while otherwise it would not be.

5.6.2 Hyperparameters

There are multiple hyperparameters associated with this defence, namely the two sliding windows' sizes and the four thresholds used in the defence.

The first sliding window is the one used to calculate the average acceleration difference between the predicted and observed speed, parts of which are then compared with the first and third thresholds. Its size thus affects the effect any single variation will have on the average and thus the likelihood of triggering either of the previously mentioned thresholds which would lead to testing for the second or fourth thresholds, respectively and consequently closer to a report of an attack. The values tested were 5, 10, 20 and 50.

The second sliding window is the one used to compute the average, or the variance of the differences calculated over the first sliding window, depending on whether it is for the second or fourth threshold test. As with said first sliding window, the size of this second one will affect the impact any single value can have upon the average and variance, and thus the chance that either of the second or fourth thresholds will be exceeded, which would trigger an attack report. The possible values were 50, 100, 250 and 500.

The four thresholds have been explained in Subsection 5.6.1 (the previous one) and are, as could be deduced, threshold values used to find out if the UAV is under spoofing attack or not, with different ones applied to different values calculated by the defence. The possible values for each were 0.01, 0.1, 0.5, 1, 10, 25 and 50.

5.6.3 Implementation

Only one version of this method was implemented, the one described in the original paper [27] and in the preceding subsections (5.6.1 and 5.6.2). An important note is that due to the large number of hyperparameters, trying every combination of values would have been too time-consuming, thus some (the first and third threshold, as well as the second and fourth thresholds) use the same values.

5.6.4 Results

Table 5.11 Linear acceleration defence results

n	N	Epsilon 1	Mu 1	Epsilon 2	Mu 2	Score
50	500	1.0	0.01	1.0	0.01	0.5314006509821438
50	500	50.0	50.0	50.0	50.0	0.5200689049659086
20	500	50.0	1.0	50.0	1.0	0.5200689049659086
20	50	25.0	1.0	25.0	1.0	0.5200689049659086
20	100	25.0	1.0	25.0	1.0	0.5200689049659086

Table 5.11 presents the results of this defence. Its top configuration had 22 normal flights with attacks wrongfully signaled within them, while only 3 of the obvious attacks and 18 of the non-obvious ones were detected somewhat on time, with 13 false alarms (too early) and 17 far too late or completely missed. All deviation attacks were missed or had false alarms as well as half of the combined ones. Of note is that the second through fifth top configurations obtained the same score and used some of the most permissive settings for the thresholds possible.

5.6.5 Discussion

This defence obtained bad results, worse than the ones of the baseline. This can further be observed when considering the flight-by-flight results of the best hyperparameter combination that are far worse than those of the baseline with far more false alarms and only a moderate increase in detection. This defence thus does not seem suited for the current task. The fact that almost all

configurations in the top 5 have the same score while using extremely large thresholds reinforces this view as these values seem to indicate that very little emphasis was put on detection and far more on evading false alarms.

5.7 LOF (Local Outlier Factor)-based Defence

5.7.1 Description

One approach for defence is to use the local outlier factor detection method [31], a variant of the nearest neighbours algorithm, though used to detect anomalous values in this case. Here, for each point, a metric called local density is calculated (using a specific distance metric) and said density is compared to that of the point's neighbours. If the density of the point is far lower than the ones of its neighbours, then it is considered an outlier and labeled as an anomaly.

This can be applied as a defence by classifying the entries in the logs as attacks or normal depending on what the local outlier factor method says. There are however multiple different approaches for applying it which must thus be detailed, depending on how new data is provided, how the data is prepared, how the outlier cutoff is chosen and others.

The approach chosen is to use the first part of the flight, the training set, as a comparison dataset and have each new point being compared to it. A related choice being whether to use the datapoints themselves or the differences between successive ones instead as arguments since that may heavily affect how the defence works and thus the results it will obtain.

The second question is whether to use the standard cutoff for outlier allowance of the local outlier factor method or instead use a custom value. This value would be found by testing some datapoints through the model and then the worst result would be multiplied by a hyperparameter factor. A related question is whether to use a part of the training set, the parameterization dataset, to find the best values for the multiplier in the thresholds instead of using said part to train the model. The other option would be to train with the whole training dataset and then run it all through the model it was used to train.

Another question is whether to use PCA to modify the data, taken from a fusion of the GNSS and IMU logs (unless stated otherwise), fed to these methods. If the choice is made to use it, a second question then comes, which is how many features to use to best conserve the information. Another related question to these is whether to normalize the data.

5.7.2 Hyperparameters

There are multiple hyperparameters within this defence, though not all of them are always used, depending on the version (for example, if the standard cutoff for outliers is used, the multiplier hyperparameter is not used). Those hyperparameters are the multiplier for the cutoff, the number of features to create using the PCA and the number of neighbours considered in the local density calculation.

The first of these, the multiplier is, as described in the previous defences, a multiplier applied to the maximum outlier factor obtained. It serves to provide some room between the maximum value observed and the maximum allowable one. The values tested were 1, 1.25, 1.5, 1.75, 2, 3, 5, 10, 15, 20 and 50.

The second one, the number of features, is used in PCA as the number of features outputted and thus, indirectly, as the amount of variance and information conserved once this preprocessing is done. The values tested were every integer between 2 and 9 included, though some versions had less types of data and thus only used up to 3 included, with -1 indicating that PCA was not used.

The third one is the number of points considered neighbours of a certain point, in other words, the number of other points that will be used to calculate its own local density, as well as the number of points its density will be compared with (note that those neighbouring points are the same in both cases). The possible values are 1, 2, 3, 4, 5, 10, 20, 30, 50, 100, 150 and 200.

Though these are not made into explicit hyperparameters, the choice of differences or actual datapoints and the usage of the parameterization dataset to find the maximum outlier factor are also aspects to take into consideration, as the impact of these would be considerable.

5.7.3 Implementation

Multiple implementations of this defence were done, and they will be explained within this subsection.

The first one used the data entries themselves with PCA. It also used the built-in classification algorithm to decide if a datapoint was anomalous or not. The process was rather simple: first the PCA was run on the dataset after having been fitted on the training set, then the defence was trained on the modified training set. Afterwards, each successive datapoint was run through it and, if it was detected by the defence as anomalous, then it would be signalled as such.

The second version uses a restricted part of the differences between successive data entries, composed only of the changes in latitude, longitude, and timestamps.

The third version was the same as the first but with a normalizer added as a preprocessing before the PCA. The fourth was a fusion of the second and third, using only the changes in latitude, longitude, and the elapsed time as data (like the second version), and adding a normalizer (like the third).

The fifth version was a variation of the third (normal dataset and normalizer) that also added a modified classification that instead used the worse score obtained by a training sample multiplied by a multiplier as a threshold, without any use of a parameterization dataset. In other words, the training set was first used to create a model and then again to see what its maximum error was. The process was modified by simply seeing the score the defence gave to the datapoints, comparing it to the threshold and, if it was worse, signal it as an attack.

The sixth was similar to the fifth (normal dataset, normalizer and custom threshold), but instead of using the whole training dataset to train the model and obtain the maximum error, it instead uses the training set without the parameterization dataset to train the model and obtain the maximum error on the parameterization dataset.

The seventh version was another variation of the fifth version which used the changes between successive points instead of the points themselves. The eighth was similar and simply added a parameterization step.

The ninth was the combination of the fourth (restricted dataset) and the seventh (normalizer, custom threshold). It used the restricted part of the dataset, only the differences in latitude, longitude, and timestamps. It also used the modified classification algorithm but no parameterization step. The tenth was similar, simply adding a parameterization step.

The eleventh and twelfth were also like the seventh and eighth respectively, using the differences between successive points. The difference between the eleventh and twelfth compared to the seventh and eighth was that while the seventh and eighth used a fusion of the GNSS and IMU logs, the eleventh and twelfth used only the GNSS logs.

5.7.4 Results

Table 5.12 LOF defence results, first version

Features	Neighbours	Score
9	200	0.0616167331755775
7	200	0.0616167331755775
4	200	0.0616167331755775
5	200	0.0616167331755775
3	200	0.0616167331755775

Table 5.12 presents the results of the first version. Its top configuration had false alarms in every flight with each of the top configurations obtaining the same score.

Table 5.13 LOF defence results, second version

Features	Neighbours	Score
3	30	0.0604884421900145
-1	30	0.0604857378290221
2	1	0.0596788874149973
-1	1	0.0595494534890075
3	1	0.0595435534921148

Table 5.13 presents the results of the second version. As with the previous version, the top configuration had false alarms in every flight. Though the scores were not identical here, they are extremely close.

Table 5.14 LOF defence results, third version

Features	Neighbours	Score
5	200	0.0979408443639658
4	200	0.0966826742320765
-1	200	0.0938430619345394
9	200	0.0934287274923779
8	200	0.0928664872066788

Table 5.14 presents the results of the third version. The best configuration of this version had 97 false alarms and three detections. An interesting observation is that even though the scores are not identical, they are here quite close to one another.

Table 5.15 LOF defence results, fourth version

Features	Neighbours	Score
2	30	0.05971713600455
2	20	0.059452933590471
-1	30	0.0590872733722155
3	30	0.0590392502329109
3	20	0.0588506209714278

Table 5.15 presents the results of the fourth version. Its top configuration had 100 false alarms. Of note is that the scores obtained by the top configurations are almost identical.

Table 5.16 LOF defence results, fifth version

Features	Multiplier	Neighbours	Score
-1	15.0	200	0.6036618939286186
-1	10.0	150	0.6020764300190138
6	15.0	20	0.6013453128436749
9	15.0	200	0.6006970494751813
9	15.0	150	0.6004333133216476

Table 5.16 presents the results of the fifth version. Its top configuration had false alarms on 30 normal flights and 2 obvious attacks were missed. Amongst the less obvious ones, 25 took quite some time to be detected if they even were. These missed attacks were most often speed attacks, though multiple deviation and combination attacks were also missed. Almost all of these were under heavily degraded flight conditions however, something that is not really the case for the false alarms whose flights were generally windy.

Table 5.17 LOF defence results, sixth version

Features	Multiplier	Neighbours	Score
9	10.0	3	0.6005286334834089
-1	10.0	150	0.5993362362383586
6	10.0	150	0.5991639571218328
-1	10.0	100	0.5983622455263312
9	15.0	3	0.5972936722318508

Table 5.17 presents the results of the sixth version. This version's top configuration had false alarms on all but 3 benign flights though one of these 47 was quite late in the flight. One obvious attack and 6 subtler ones took some time to be detected. These missed ones were all speed attacks.

Table 5.18 LOF defence results, seventh version

Features	Multiplier	Neighbours	Score
-1	1.5	50	0.7273295101359879
-1	1.5	30	0.7218024934256172
8	2.0	10	0.7185919270112235
-1	1.5	200	0.7181637973244657
8	5.0	2	0.7167864778408739

The results of the seventh version are presented in Table 5.18. The top configuration had 21 normal flights with false alarms while a significant part of the flight remained but only 14 attacks detected late or not at all, none of them obvious. The missed attacks are spread between the speed and deviation attacks relatively equally. While these were generally under heavily affected flight conditions, the false alarms were for the majority not.

Table 5.19 LOF defence results, eighth version

Features	Multiplier	Neighbours	Score
8	1.75	4	0.7316120031426655
8	2.0	4	0.7227007691587438
9	1.75	4	0.719153778167043
3	3.0	4	0.7177695342426925
8	1.5	5	0.7150753214360229

Table 5.19 presents the results of the eighth version. The top configuration had 21 normal flights with false alarms while a significant part of the flight remained but only 12 attacks detected late or not at all, none of them obvious. The false alarms and attacks missed are almost identical to those in the previous version.

Table 5.20 LOF defence results, ninth version

Features	Multiplier	Neighbours	Score
-1	1.5	200	0.7849923135642805
3	1.5	200	0.7849923135642805
3	1.25	200	0.7828606245599206
-1	1.25	200	0.7828606245599206
-1	1.5	150	0.7734468657506457

Table 5.20 presents the results of the ninth version. The top configuration mistakenly detected attacks far too early in 16 of the 50 benign flights and only missed 9 attacks, all subtle ones, which were spread out between speed and deviation attacks, though for the vast majority were under heavily affected conditions, something that was not the case for the majority of the benign flights which were simply windy.

Table 5.21 LOF defence results, tenth version

Features	Multiplier	Neighbours	Score
-1	1.5	150	0.7865818553033713
3	1.5	150	0.7865818553033713
3	1.5	200	0.7775198026983776
-1	1.5	200	0.7775198026983776
-1	1.5	100	0.7699831454836137

The results of the tenth version of the LOF defence are presented in Table 5.21. The best configuration had 16 false alarms in normal flights and missed 9 attacks, all of which were non-obvious ones. Its results are almost completely identical to those of the previous version (the ninth version)

Table 5.22 LOF defence results, eleventh version

Features	Multiplier	Neighbours	Score
-1	2.0	20	0.8087331786192508
9	2.0	20	0.8087331786192508
8	2.0	20	0.8087331786192508
7	2.0	20	0.8087331786192508
9	2.0	30	0.8086556594821126

Table 5.22 presents the results of the eleventh version. The top configuration had early misdetections on 15 normal flights and missed 8 attacks, all subtle ones, half of them speed attacks while the other half were deviation attacks, though all of them were under heavily affected conditions. Two-thirds of the false alarms were during flights with only the wind, while the remaining third were in flights with more heavily affected conditions.

Table 5.23 LOF defence results, twelfth version

Features	Multiplier	Neighbours	Score
9	1.75	30	0.7973556994270834
-1	1.75	30	0.7973556994270834
8	1.75	30	0.7973556994270834
7	1.75	30	0.7973556994270834
-1	1.5	100	0.7964557608518759

The results of the twelfth version are presented in Table 5.23. Its top configuration had 17 false alarms and 7 missed attacks, all subtle ones. Its false alarms and missed attacks were almost all the same as the eleventh version.

For quite a lot of these versions, the different configurations of a version were often quite close in score, even sharing the same results multiple times.

5.7.5 Discussion

The first four versions' results were very bad, being far worse than the baseline with the only one not having false alarms in every flight being the third version. This is undoubtedly because the standard cutoff used in local outlier factor is unsuited to the amount of variation present in these datasets. The almost if not completely identical scores in each version's top 5 reinforce this

interpretation, as these similarities imply that modifying the hyperparameter values changes nothing.

The fifth version's results were better, reaching scores slightly higher than the baseline. This indicates that the usage of a multiplier instead of the built-in classification algorithm significantly enhances performance. This is due to its greater flexibility as it is able to adapt itself to the flight conditions better than the built-in threshold. The results also indicate that the multiplier is by far the most impactful setting as it almost did not change over the top 5 results while the number of neighbours and the number of features did so far more. All that being said, the performance was still not that impressive compared to other defences, as there were many false alarms and multiple obvious attacks missed.

The sixth version's results were slightly worse than the previous version's ones, though still slightly better than the baseline with more false alarms but slightly less missed attacks. This indicates that the usage of a parameterization step does not help as it probably takes away valuable data from the training.

The seventh version's results improved on those obtained by the fifth with a higher score with less false alarms and less missed attacks, in all likelihood because the changes between successive datapoints provide a better repartition of the data thus the defence can isolate outliers more easily. The eighth version improved even more, though only slightly, a benefit brought to it by the use of the parameterization step to improve its results.

The ninth version obtained even better results than the previous one, with less false alarms and less missed attacks. This is due to the restricted dataset that enabled the defence to concentrate on the more important information. The addition of the parameterization step in the tenth version improved very slightly on these results, though its impact is arguably negligible.

The eleventh version had even better results, having scores over 0.8 and obtained better results than those achieved by almost every defence until now. This is due to the usage of the whole GNSS logs without the IMU data but with the other information that was discarded when using only latitude, longitude, and timestamps. This middle ground between the two extremes enabled this version to only focus on relevant information without discarding too much. The twelfth version highlighted this pattern, obtaining in turn better results than almost all others, though its addition

of a parameterization step clearly did not improve upon the eleventh version, a clear sign that as much data as possible should be kept for training.

This defence has obtained one of the best scores so far with its eleventh version, with no PCA, 20 neighbours and a multiplier value of 2. It achieved a better score than the baseline and is the only one to have even passed the 0.8 mark since the autoencoder.

There are multiple other conclusions that can be drawn from these results. One of them is the low importance of the PCA and its number of features, as the top results were generally with no PCA or a high number of features, with the results being often similar if not identical. Another observation is the mixed success rate of the parameterization step. It seems to have been detrimental in the sixth and twelfth versions, but helpful in the eighth and tenth. Some other important observations are that using only the GNSS logs but not both GNSS and IMU proved beneficial and that the modified decision algorithm as well as the choice of using changes in data entries instead of the entries themselves heavily impacted the results for the better.

5.8 OCSVM (One-Class Support Vector Machine)-based Defence

5.8.1 Description

Another method attempted was the use of a one-class support vector machine to classify flights as under attack or not [31]. A one-class support vector machine is, as its name indicates, a variation of a support vector machine with only one class. A support vector machine is a classification algorithm which operates by creating a linear boundary (a hyperplane) separating two different classes and maximizing the distances between it and the points closest to it of each class, with the possibility of varying how much data from one class is allowed to be in the other's "half" (on the other side of the boundary) and to use kernels to change the data's representation [31].

In the current case, though the two classes are quite obvious (under attack and not under attack), there are no datapoints where the UAV is under attack within the training part of a flight (as stated in Chapter 3). It is thus impossible to directly apply such a method. However, a variation called a one-class support vector machine can be used where instead of creating a separation between two different classes it instead creates the smallest region including a given proportion of the data, thus naturally selecting the set of datapoints closest to each other that will then be considered the normal data [31]. Consequently, some normal points in the training set will be labeled as anomalous even

though they are not, which can be compensated by introducing a threshold for the maximum deviation allowed.

The data used was normally a fusion of the IMU and GNSS logs except versions where other data is specified.

The general procedure followed was to train the defence on the training part of the flight and then classify the rest of the flight based upon whether the OCSVM would signal a particular datapoint as anomalous.

This however masks a number of choices that complexify it. Amongst these is the choice of whether the datapoints should be the entries in the logs themselves or the differences between successive entries. Other issues to consider are whether to use a part of the training data of the flight as a parameterization dataset to find a threshold value for a datapoint's outlier score before classifying it as an anomaly, use the whole training data itself for that task, or to instead use the boundary itself to do this separation. Another question, as with other methods, is whether to use PCA on data before classification to potentially improve results and, if so, how many components to keep afterwards. A final one is whether to use the built-in classification algorithm or to use a modified one that makes use of multipliers to construct a custom threshold.

5.8.2 Hyperparameters

There are multiple hyperparameters used with this method, notably the type of kernel, the number of features to keep in PCA and the multiplier in the threshold calculation.

The kernel types in a support vector machine serve to change the representation of the data, which can make boundaries that are nonlinear in a representation become linear in another. The different possible kernel types are a linear kernel, a polynomial one, a sigmoid or a radial basis function.

The number of features hyperparameter is the number of features extracted using PCA, which affects the amount of variance in the original data that the PCA can conserve through its process. The possible values are all integer between 2 and 9 included, with -1 also used to denote that PCA is not used.

The multiplier is the multiplier applied to the maximum outlier score obtained when running the defence on the parameterization dataset to obtain a threshold that is thus potentially more tolerant

of outliers, depending on the value of that hyperparameter. Its possible values are 1, 1.25, 1.5, 1.75, 2, 3, 5, 10, 15, 20 and 50.

Another hyperparameter, but only used for one version, consists in the degree of the polynomial in the polynomial kernel. Its possible values are all integers between 1 and 10 included.

5.8.3 Implementation

Multiple implementations of this defence were done and will be covered within this subsection. The versions implemented were quite similar to those made for the LOF defence. However, a problem often encountered was that the defence did not mesh well with PCA and would thus either hang (freeze) or take an extremely long time to run. Thus, PCA could not be used nearly as often.

The first version used the data entries themselves with PCA. It also uses the built-in classification algorithm to decide if a datapoint should be considered anomalous or not. The process was as with the LOF defence: the PCA was first run on the dataset after having been fitted on the training set, the defence was then trained on the training set, each successive datapoint was afterwards also run through this process and, if it was detected by the defence as anomalous, then it would be signalled as such. In the end, PCA did not work with this version (the program froze).

The second version uses a restricted part of the differences between successive data entries, composed only of the changes in latitude, longitude, and timestamps. This one was also implemented without a normalizer or a validation step.

The third version was the same as the first but with a normalizer added as more preprocessing before the PCA.

The fourth version was the same as the second but with a normalizer added as preprocessing before the PCA. Here again, PCA did not work in the end.

The fifth version is a variation of the third that also added a modified classification that instead used the worst score obtained on the training set multiplied by a multiplier as a threshold, without any use of a parameterization dataset. In other words, the training set was first used to create a model and then again to see what its maximum error was with said model. The process was modified by simply checking the score the defence gave per datapoint, comparing it to the threshold and, if it was worse, consider it an attack.

The sixth was also similar to the fifth, but instead of using the full training dataset to train the model and obtain the maximum error, it would instead use the training set without the parameterization dataset to train the model and obtain the maximum error through the parameterization dataset.

The seventh version was another variation of the fifth version which used the changes between successive points instead of the points themselves. The eighth was almost exactly the same, except with an added parameterization step on top.

The ninth was the combination of the fourth and the eighth. It used the restricted part of the dataset, only the differences in latitude, longitude, and timestamps. It also used the modified classification algorithm but no parameterization step. PCA would not work with this version either. A tenth version was almost exactly like the ninth but with an added parameterization step and with working PCA.

The eleventh and twelfth versions were done as the seventh and eighth but instead of using the fusion of the GNSS and IMU logs they only used the GNSS one.

A thirteenth version was also done that instead focused on the polynomial kernel, thus only using said kernel. It was the eighth version but with a new hyperparameter, the degree of the kernel's polynomial used as an argument. PCA also did not work here.

5.8.4 Results

Table 5.24 OCSVM defence results, first version

Kernel	Features	Score
linear	-1	0.5200689049659086
poly	-1	0.5200689049659086
sigmoid	-1	0.5200689049659086
rbf	-1	0.056127366638963

The results of the first version are presented in Table 5.24. Its top configuration had no false alarms and only detected two attacks, both in obvious flights, those where the UAV was teleported to its destination.

Table 5.25 OCSVM defence results, second version

Kernel	Features	Score
linear	2	0.0658885176591396
poly	3	0.0652433869169743
poly	2	0.0652148715850157
linear	3	0.0568171603841175
poly	-1	0.0565135888622808

The results of the second version are presented in Table 5.25. The top configuration had false alarms in all but two flights, one benign one and another with an attack, attack that it did not detect. Both of these flights did not have any modified flight conditions.

Table 5.26 OCSVM defence results, third version

Kernel	Features	Score
linear	4	0.2435604368805021
linear	7	0.2282668508183095
linear	8	0.2254679841078185
linear	9	0.2254679841078185
linear	-1	0.217023652628829

Table 5.26 presents the results of the third version. For its top configuration, 16 benign flights were not mistakenly flagged as attacks, while two obvious attacks were missed and seven flagged too early with one detected on time. Out of the other, subtler attacks, two were accurately detected, with eight detected too late and thirty before they even took place. Both the flights without false alarms and those with correctly detected attacks were an even spread of slightly and heavily affected ones, with these accurate detections not being specific to a certain type of attack either.

Table 5.27 OCSVM defence results, fourth version

Kernel	Features	Score
poly	-1	0.0584009407297605
rbf	-1	0.0562560842109899
sigmoid	-1	0.0562273666194763
linear	-1	0.0556411247685206

Table 5.27 presents the results of the fourth version. The top configuration had false alarms in every flight.

Table 5.28 OCSVM defence results, fifth version

Kernel	Multiplier	Features	Score
linear	5.0	9	0.6236015186534577
linear	3.0	7	0.6178592343206055
linear	5.0	8	0.6145744950450919
linear	10.0	9	0.6122094205413668
linear	5.0	7	0.611182186928034

The results of the OCSVM defence's fifth version are displayed in Table 5.28. The top configuration had 19 false alarms, all within the benign flights and under both heavily and lightly altered flight conditions, as well as missing half of the obvious attacks, 16 of the non-obvious ones and was late for two more. Approximately half of the speed attacks and half of the deviation attacks were missed. These were also approximately half under heavy flight conditions and half under lighter ones.

Table 5.29 OCSVM defence results, sixth version

Kernel	Multiplier	Features	Score
sigmoid	3.0	2	0.6575864641494656
poly	50.0	7	0.6570875689378216
sigmoid	2.0	-1	0.6543584638633397
sigmoid	1.5	-1	0.649843109943249
poly	50.0	6	0.6466369846182977

Table 5.29 presents the results of the sixth version. There were three false alarms within the benign flights for the configuration with the top result, three missed obvious attacks, 28 subtler ones missed and another detected late. All of the combination attacks were missed, three quarters of the speed ones and more than half of the deviation ones, mostly, though not always, under heavily altered conditions.

Table 5.30 OCSVM defence results, seventh version

Kernel	Multiplier	Features	Score
linear	2.0	8	0.7039282071517807
poly	50.0	2	0.6900070521604494
linear	3.0	8	0.6897251359736952
linear	2.0	-1	0.687935917672961
linear	1.75	-1	0.6824452895329032

Table 5.30 presents the results of the seventh version. Its top configuration had 20 false alarms in benign flights and 2 in those with attacks, mostly under lightly altered conditions. It missed one obvious attack and nine subtler ones, with a further four that were detected late, most of these under heavily altered conditions. These missed attacks were in majority speed and combination attacks.

Table 5.31 OCSVM defence results, eighth version

Kernel	Multiplier	Features	Score
linear	5.0	8	0.697554561433187
linear	3.0	4	0.6944189018714658
linear	3.0	5	0.6908356994829624
linear	3.0	3	0.6884498014220745
linear	2.0	4	0.6802618374204278

Table 5.31 presents the results of the eighth version. The top configuration had 13 false alarms during normal flights and missed 21 subtle attacks, spread proportionally amongst the different attack types. These attacks were generally missed while under heavily altered flight conditions, something that was not as much the case for the false alarms.

Table 5.32 OCSVM defence results, ninth version

Kernel	Multiplier	Features	Score
linear	2.0	-1	0.6715052590213463
linear	1.75	-1	0.6574544134681775
sigmoid	1.5	-1	0.6517936668964184
linear	1.5	-1	0.6476884910243479
sigmoid	1.25	-1	0.6466704753262311

Table 5.32 presents the results of the ninth version. Its top configuration had 11 false alarms, 2 missed obvious attacks, 21 missed subtle attacks and 2 more detected late. These missed subtle attacks were spread proportionally amongst the different types. Though multiple of these missed attacks were in flights with heavily affected conditions, approximately as many were not, a ratio that also holds true for the false alarms.

Table 5.33 OCSVM defence results, tenth version

Kernel	Multiplier	Features	Score
linear	2.0	2	0.7119069862164293
linear	3.0	-1	0.7110469091056432
linear	2.0	-1	0.692821560305626
linear	3.0	2	0.68727546956499
linear	2.0	3	0.6762071393965483

The results of the tenth version are displayed in Table 5.33. The top configuration for this version had 8 false alarms on normal flights, 22 missed non-obvious attacks and one more detected quite late, with these missed attacks spread out evenly between the speed and deviation attacks. As for the previous versions, though many of the flights with either missed attacks or false alarms were under heavily altered flights conditions, as many were not.

Table 5.34 OCSVM defence results, eleventh version

Kernel	Multiplier	Features	Score
linear	1.5	5	0.7592631338347492
linear	1.5	7	0.7579387726202538
linear	1.5	6	0.7569385193667232
linear	1.5	4	0.747092490468373
linear	1.75	7	0.746742396964594

Table 5.34 presents the results of the eleventh version, whose top configuration had 14 false alarms and 13 missed attacks, all of them subtle. These missed attacks were not concentrated in particular types of attacks but were instead in flights with heavily affected conditions. The false alarms, however, were mostly in flights where the wind was the only affected condition.

Table 5.35 OCSVM defence results, twelfth version

Kernel	Multiplier	Features	Score
linear	2.0	2	0.7268656694443032
linear	2.0	5	0.7176655528523642
linear	2.0	4	0.715659020443045
sigmoid	1.75	-1	0.7104514191151966
linear	3.0	4	0.7083554664844829

Table 5.35 presents the results of the twelfth version. There were 15 false alarms for the top scoring configuration as well as one missed obvious attack and 15 missed subtler ones. These were for the majority in the same locations as for the previous version, though with a few errors added.

Table 5.36 OCSVM defence results, thirteenth version

Kernel	Multiplier	Features	Score
poly	20.0	-1	0.6640423976903104
poly	5.0	-1	0.6611523544842863
poly	50.0	-1	0.655628670973525
poly	15.0	-1	0.655226687183113
poly	50.0	-1	0.6540790866686951

The results of the thirteenth version are displayed in Table 5.36. The top configuration had 30 false alarms in the benign flights and one in a flight under attack. It also missed six subtle attacks and was late to react to three more. These missed attacks were mostly of the speed type though they were generally in heavily affected flights. The false alarms, however, were not in a specific type of flight and were instead in a wide variety of possible conditions, some under heavily altered conditions, some not at all.

5.8.5 Discussion

The first version does not seem to have done much, as, even though it had no false alarms amongst the flights, it seems to only have noticed the two attacks that teleported the UAV directly to the destination.

The second version, in contrast, does something, though it can hardly be called an improvement. Out of the benign flights, only one was not flagged as containing an attack and, out of the 50 flights

with attacks, all but one had false alarms where attacks were detected too early, while the lone flight where that was not the case was instead not flagged at all, indicating a complete failure at the current task.

The third version managed to detect some attacks, though it also had quite a lot of false alarms while the fourth returned to the problems of the second version, with all flights having attacks falsely detected in them too early. The results of these four versions, when taken into account together, indicate that the built-in threshold does not seem suited to the current task.

The fifth version improved on all previous ones, finally beating the baseline though not by a wide margin, missing multiple obvious attacks. This improvement is due, in all likelihood, to the custom threshold that affords far more flexibility in setting the rejection boundaries than the built-in one. The sixth version improved even more by using a parameterization dataset to obtain a better value for the threshold, however it still missed obvious attacks.

The seventh version continued this trend of improvement, having the first result above 0.7 and continuing to reduce the overall number of missed attacks and false alarms, undoubtedly because it used the changes between successive datapoints which are not supposed to change as much as those datapoints themselves (during a flight the latitude and longitude progressively change from the departure to the destination while the changes in latitude and longitude between successive datapoints should remain relatively the same). The eighth version had some slightly worse results than the seventh version, indicating that the parameterization's usage of some of the training set for a better threshold did not compensate for the corresponding loss of training data.

The ninth version, though it did improve on the results of the fourth version, which used the same data, failed to reach the levels of the seventh or eighth. This may be due to the usage of a restricted dataset composed only of the latitude and longitude changes as well as the time differences. This conclusion, however, is hard to make as the lack of principal component analysis may have hurt the results obtained. The tenth version furthers this latter interpretation as its results outperform the ninth version (to which it is identical except for the added parameterization step) by a greater amount than the versions in a similar situation such as versions 5 and 6, where the difference is of around 0.03 and versions 7 and 8 where the difference is less than 0.01. Furthermore, as the tenth version outperforms versions 7 and 8, the conclusion that using the restricted dataset lowers the result should be rejected.

Reducing the size of each data entry by using only the GNSS log enhanced the performance of the defence, with the eleventh version obtaining better results than all others to date. Adding a parameterization step did not improve the results obtained by the twelfth version, on the contrary these are worse than those obtained by the eleventh version, upon which it is based.

The thirteenth version was conceptualized as a verification that the polynomial kernel was not disadvantaged by not setting explicitly the degree of the polynomial. This seems to have been the case as its results are slightly worse than those of the eighth version upon which it is directly based.

The best setup seems to have been the eleventh version with a linear kernel, a multiplier value of 1.5 and 5 features used in the PCA, as this setup had the highest score out of all those tested. Some more general observations can be made here such as the fact that amongst the best-performing versions, the linear kernel seems to obtain the best results. Another important observation is the fact that the modified algorithm heavily enhanced the results obtained. An interesting observation is also the fact that using only the GNSS logs and not the IMU and GNSS logs fused leads to better results. The usefulness of the parameterization step is unclear, as it improved the results in some cases but in others it did the opposite. All in all, this defence obtained some good results and enabled some interesting observations.

5.9 Distance Defence

5.9.1 Description

The following defences, this one included, served as simpler defences for the purpose of comparison. They were inspired by questions such as “How would a defence perform if it just looked at how far the UAV went or how fast it was going?” and were thus created to provide answers to these inquiries.

This defence consists in comparing the distance between the two most recent positions to a previously calculated reference value and, should the difference be greater than a threshold, the latest point would be considered under attack. This distance is calculated using the GNSS log’s information. This general idea was refined into multiple versions that varied depending on how and with what data the reference distances and thresholds are calculated.

5.9.2 Hyperparameters

There are two hyperparameters used in this defence, the multiplier and the lookback. The multiplier is applied to the threshold used for the difference between the reference value and the actual obtained one, to provide a greater flexibility in the possible results. The possible values were 0.01, 0.1, 1, 2, 3, 5, 7.5, 10, 25 and 50. The lookback is used when the reference value is calculated based upon a series of previous data entries. It serves to specify the length of said data entry series. The values tested were 5, 10, 20, 25, 50, 100, 200 and 400.

5.9.3 Implementation

Five different versions were implemented, the first of which calculated the reference value as the mean of the distances between a series of successive points leading to the present one, with the length of the series depending on the lookback hyperparameter. The threshold was calculated by multiplying the standard deviation of the previous series with the multiplier. The second version did the same, but instead of calculating the standard deviation for the threshold, it instead calculated the maximum difference between any distance in the series and the average.

The third version did as the first, but the series used for reference value and threshold estimation instead constituted of the training dataset (with no usage of the parameterization dataset). The fourth version did the same but used the maximum difference as the threshold instead (like the second). The fifth version was a variation of the fourth (and thus the second). It calculated the reference value as the mean over the training dataset while the threshold was the maximum difference between said reference value and any value of the parameterization dataset.

5.9.4 Results

Table 5.37 Distance defence results, first version

Multiplier	Lookback	Score
5.0	100	0.6900682268206901
10.0	20	0.6868755462189949
10.0	25	0.6817436660559535
5.0	400	0.6797373207648542
10.0	50	0.6794737487746787

Table 5.37 presents the results of the first version of the distance defence. The top result had 19 misclassifications during the normal flights, while detecting correctly all obviously spoofed ones and either being significantly late to detect attacks or even completely failing to do so for 20 non-obvious ones. These missed attacks were all the deviation attacks, most of the combination attacks, and a few speed attacks. The majority of the false alarms were windy flights, with no other flight condition affected.

Table 5.38 Distance defence results, second version

Multiplier	Lookback	Score
3.0	100	0.6940245278961171
3.0	50	0.6868487532310238
5.0	50	0.6815566887439364
5.0	25	0.6779100579894779
3.0	200	0.6774221411998055

Table 5.38 presents the results of the second version. The top result emitted seven false alarms during non-spoofed flights. It missed no obviously spoofed flights but 25 of the 40 non-obvious flights and was somewhat late in spotting the attack on three of the remaining fifteen. It missed all of the same attacks as the previous version, as well as a few more speed attacks, though these were under heavily affected conditions.

Table 5.39 Distance defence results, third version

Multiplier	Score
5.0	0.6980375711266437
7.5	0.6739044390327945
10.0	0.6591662785981198
25.0	0.6352230361197081
50.0	0.600204149248036

The best results of the third version are displayed in Table 5.39. The best configuration for this version had only 18 misclassifications of benign flights, no obvious attacks missed and only 18 of the 40 subtler ones missed. These missed attacks were composed of all the deviation type attacks

as well as a few speed and combined attacks, though under heavily altered flight conditions. The false alarms on the benign flights were, as often, mostly on flights with wind and little else.

Table 5.40 Distance defence results, fourth version

Multiplier	Score
2.0	0.6851604861816305
3.0	0.6697286028149929
5.0	0.6510697370607323
10.0	0.628948125303421
7.5	0.6243159911970284

Table 5.40 presents the results of the fourth version. The top configuration emitted false alarms quite early for 15 normal flights and, even though it correctly detected all obvious attacks, it missed 21 of the 40 subtler ones. The attacks missed are similar to the previous versions, being the deviation attacks as well as some combined ones and a few speed attacks. The false alarms were also similar to the previous versions, as they occurred mostly on flights with wind but little else.

Table 5.41 Distance defence results, fifth version

Multiplier	Score
3.0	0.6891039668980404
5.0	0.6586740261878257
2.0	0.650366459065781
10.0	0.6284005944153663
7.5	0.6230525525946092

Table 5.41 displays the best results of the fifth version. Its top configuration misclassified 14 of the 50 normal flights and correctly detected all obvious attacks but either took a long time to detect or completely missed 21 of the 40 less obvious ones. The results of the top configuration are almost identical to the fourth version, with almost exactly the same false alarms and missed attacks, except for one or two flights.

5.9.5 Discussion

The results obtained for the first version were interesting, with a better performance than the baseline. The attacks it failed to detect were mostly those incorporating a change in heading, something it was not designed to handle.

The second version obtained similar overall results, with a slightly higher top score, but quite different flight-by-flight ones with less false alarms but more missed attacks. This is probably due to its more lenient threshold calculation, using the maximum difference observed instead of the standard deviation.

Unlike the linear regression defence, for this one the lookback seems to have somewhat more of an impact as the multiplier parameter does also change within the top 5 of the first two versions. This greater impact is probably due to the fact the lookback controls the size of the series used for the rolling average and threshold which gives it more impact than what it has in the linear regression (Section 5.4). That being said, in both of these versions of the current defence, the multiplier switches between two values, which might simply indicate that its optimal value was not amongst those tested.

The third version's top result was slightly better than the previous two which might indicate that using a training dataset helps obtain better results than using a rolling average and threshold. This interpretation is however undercut by the fact the fourth and fifth versions underperformed compared to the second version (and compared to the first, however they are not as closely related to it as they are to the second) and it is thus unclear if one of these two options is better. The fact that all results are so close (less than 0.02 between the best and worst-performing version) furthers the idea that it is for now impossible to clearly crown one of the approaches.

The other observation that is potentially of greater interest is the fact that the fifth version outperformed the fourth. This indicates that its use of a parameterization dataset to calculate the threshold and produce one better-suited to unseen data is worth the reduced training set.

Apart from these two points, the main things to note are that this defence seems to improve by a non-negligible amount over the baseline's score of around 0.6, as such, it could be an interesting avenue to potentially explore further, and that, as could be expected from a theoretical point of

view, it has difficulty dealing with cases where the distance is not modified, something that is admittedly easy to deduce from its description.

5.10 Speed Defence

5.10.1 Description

This defence is a variation of the distance one explained in Section 5.9. The speed of the UAV between two successive positions is compared to a reference value and, if the difference is greater than a certain threshold, the latest point is considered an attack. As for the distance defence, there are multiple versions depending on how the reference values and threshold are calculated.

5.10.2 Hyperparameters

As with the distance defence, there are two hyperparameters, the lookback and the multiplier, with the lookback used to determine the length of the series of datapoints used to calculate the reference values and thresholds, while the multiplier is used to give a greater degree of leeway to calculate the thresholds. The possible values are the same as with the distance defence, 0.01, 0.1, 1, 2, 3, 5, 7.5, 10, 25 and 50 for the multiplier and 5, 10, 20, 25, 50, 100, 200 and 400 for the lookback.

5.10.3 Implementation

Here again, five different versions were implemented. The first version calculated the reference value as the mean distance of a series of the latest points, with the length of the series depending on the lookback hyperparameter. The threshold was calculated by multiplying the standard deviation of the previous series with the multiplier. The second version did the same but used the maximum difference between any distance in the series and the average as a threshold instead of the standard deviation.

The third version did as the first, but the series used for reference value and threshold estimation was instead made of the training dataset. The fourth version did the same but used the maximum difference instead of the deviation and the fifth added a parameterization step on top of this threshold change.

5.10.4 Results

Table 5.42 Speed defence results, first version

Multiplier	Lookback	Score
10.0	20	0.702908767646646
10.0	25	0.689881117412149
10.0	50	0.6837606354389666
7.5	20	0.6832067558790322
5.0	100	0.6805312133021869

Table 5.42 displays the results of the first version. The top configuration only incorrectly detects attacks during seven benign flights and correctly detects all obvious attacks, though it fails to detect 25 of the 40 non-obvious ones. These missed attacks were a few speed attacks under heavily degraded conditions as well as all deviation and combined attacks. The missed flights were mostly those with only wind.

Table 5.43 Speed defence results, second version

Multiplier	Lookback	Score
5.0	20	0.702381618693575
5.0	25	0.693655957063814
3.0	50	0.6874021649065774
3.0	100	0.6842698064422109
5.0	50	0.6759528228045629

The results of the second version are shown in Table 5.43. The best configuration only incorrectly signals attacks two times out of the fifty normal flights, while being able to detect all obvious attacks. It however fails to detect 28 of the 40 subtler ones and detects quite late a twenty-ninth one, missing more speed attacks that were not under heavily degraded conditions and missing all deviation and combined attacks.

Table 5.44 Speed defence results, third version

Multiplier	Score
5.0	0.6867446969253027
7.5	0.6625363093966246
10.0	0.6509876022920272
25.0	0.6251614037621213
50.0	0.5921463405710279

Table 5.44 displays the results of the third version. Its best-scoring configuration incorrectly signalled attacks on 16 of the 50 normal flights and, even though it correctly detected all obvious attacks, it missed or detected quite late half of the 40 non-obvious ones. The false alarms were half in flights with only wind and half in flights with heavily degraded conditions while the missed attacks were concentrated on deviation and combined attacks which were almost all missed.

Table 5.45 Speed defence results, fourth version

Multiplier	Score
2.0	0.6769500629878419
3.0	0.6515285827793811
5.0	0.632839290735946
10.0	0.6243364340447447
7.5	0.6115297429021591

Table 5.45 presents the results of the fourth version. The top configuration here caused false alarms on 15 benign flights and detected quite late, if at all, 22 of the non-obvious attacks, though it did detect the 10 obvious ones. Those 15 false alarms were mostly in flights with wind and nothing else. The missed attacks were all the deviation attacks as well as half of the combined attacks and a few speed attacks under heavily degraded conditions.

Table 5.46 Speed defence results, fifth version

Multiplier	Score
3.0	0.6611322227044354
5.0	0.6488358443459379
2.0	0.639281700203562

Table 5.47 Speed defence results, fifth version (cont'd)

10.0	0.6235160473622896
7.5	0.6104604305872728

Tables 5.46 and 5.47 list the results of the fifth version, whose top configuration incorrectly detected attacks in 14 of the benign flights, mostly windy flights with no other flight conditions affected. It also missed or reacted quite late to 24 of the subtle spoofing attacks, missing all deviation and most of the combination attacks as well as a few speed attacks under heavily affected conditions.

5.10.5 Discussion

The first version has some interesting results, clearly beating the baseline in score. Its main failures were in detecting deviation and combination attacks, something it was ill-suited for as it only looks at the speed and not the direction. The second version is in a similar situation, obtaining marginally slightly inferior results to the first one, indicating that the choice of using the standard deviation or the maximum difference does not change much. Both of these versions beat all the results obtained by the distance defence of Section 5.9, indicating that the choice of speed over distance is a good one. This is in all likelihood because of the ability of speed to take into account a variation in the time elapsed between two datapoints, something that a defence based upon distance cannot do.

The next three versions, however, had worse results than the first two, indicating that the usage of a training dataset is suboptimal compared to rolling averages and thresholds. This is quite probably because of the greater flexibility afforded by looking at the most recent points instead of having to use as a reference data from the start of the flight.

5.11 Heading Defence

5.11.1 Description

This version is also a variation of the distance defence but with a focus on the UAV's heading instead of the distance or the speed. The average heading of a series of datapoints is used as a reference value while a threshold based upon the differences between the reference value and a series of datapoints. Then, for every point in the flight, if the difference between that point and the reference value is greater than the threshold, the point is considered faked and an attack.

5.11.2 Hyperparameters

There are, as in the previous case, two different hyperparameters. The first of these is the lookback, which establishes the number of successive datapoints used to establish the reference value and the threshold. The possible values are 5, 10, 20, 25, 50, 100, 200 and 400. The second of these is the multiplier used in the threshold calculation, with the possible values for this being 0.01, 0.1, 1, 2, 3, 5, 7.5, 10, 25 and 50.

5.11.3 Implementation

As with the distance and speed defences (Sections 5.9 and 5.10), five versions were implemented. The first used a rolling average and a standard deviation multiplied by the multiplier as threshold while the second replaced the standard deviation by the maximum difference between a value of the series used to calculate the average and said average. The third used the training set at the beginning of the flight to calculate the average and standard deviation. The fourth did like the third but replaced the standard deviation by the maximum difference and the fifth did it like that as well but added a parameterization step to calculate the maximum difference.

5.11.4 Results

Table 5.48 Heading defence results, first version

Lookback	Multiplier	Score
50	7.5	0.6197883407576237
25	7.5	0.6193503385105821
25	10.0	0.6182178047529387
400	5.0	0.6169122004772117
400	10.0	0.6109576535934665

The results of the heading defence's first version are displayed in Table 5.48. The top configuration incorrectly signalled attacks in three normal flights and missed the four obvious attacks teleporting the UAV forward as well as 32 out of 40 subtler ones, all of the speed attacks as well as most of the deviation and combined attacks.

Table 5.49 Heading defence results, second version

Lookback	Multiplier	Score
5	0.1	0.5605923746740572
400	2.0	0.5400128394233018
100	2.0	0.5400121952801925
200	2.0	0.5400121952801925
50	2.0	0.5400121952801925

Table 5.49 shows the results of the second version. Its best configuration incorrectly detected six attacks on flights with none and six too early on flights where some happen later. It also missed four obvious attacks and 29 subtler ones. The attacks missed are similar to the previous version.

Table 5.50 Heading defence results, third version

Multiplier	Score
5.0	0.6416551837252643
7.5	0.6379045829726399
10.0	0.6115940966333007
25.0	0.5720801980684201
50.0	0.551985069863333

Table 5.50 displays the results of the third version. The best configuration had mistakenly detected attacks in eight of the normal flights, almost all of which only had wind, while in the flights with attacks, four obvious ones were not detected and 26 non-obvious ones were not either. These missed attacks were the four obvious attacks teleporting the UAV forward, all speed attacks as well as half of the deviation ones, though it detected all the combination attacks.

Table 5.51 Heading defence results, fourth version

Multiplier	Score
2.0	0.6295777481230158
3.0	0.6182341625766259
5.0	0.6020603085114187
7.5	0.5809325387629314
10.0	0.5720769066854215

Table 5.52 shows the results of the fourth version. For the best configuration, nine of the normal flights were incorrectly classed as under attack while four of the obvious attacks, all the forward teleportation ones, and 26 of the non-obvious ones were missed. These missed subtler attacks were almost all speed attacks as well as half of the deviation attacks and one combination one.

Table 5.52 Heading defence results, fifth version

Multiplier	Score
3.0	0.6383443629683875
2.0	0.6295512309386312
5.0	0.6098312415852932
7.5	0.5909253672363178
10.0	0.5818888867295429

Table 5.52 presents the results of the fifth version. The top configuration only had five early mistaken detections in the 50 benign flights. However, it missed the same 4 obvious attacks as the other versions (forward teleportation) and 28 of the subtler ones, all the speed attacks as well as most of the deviation ones and one combination attack.

5.11.5 Discussion

All five versions had similar results, varying mainly in how many false alarms they had. They mostly missed the same attacks, mainly those affecting the UAV's distance but not its heading, such as the four obvious attacks that teleported the UAV forward in its path as well as the 20 speed attacks. Though this was predictable given the fact this defence tests only the heading and not the distance traveled or the speed, it is nonetheless a major problem as it renders this defence unable to detect some obvious attacks. The fact that all versions missed multiple deviation attacks is a warning that this defence is not even good at stopping the attacks it is designed to.

5.12 Combination Defence

5.12.1 Description

From the previous three defences (Sections 5.9 to 5.11), an idea naturally springs to mind: to simply combine them and see the results obtained. As the speed defence had better results than the distance defence and considering they both aimed to cover the same types of attacks, the choice was made

to use the speed defence and combine it with the heading one. This combination works as both did, except at the same time. A reference value is established for the speed, and another one for the heading. Two different thresholds are then created, one for each, and they are then used to test the differences between the data entries and the reference values. If either threshold is surpassed, then the defence signals an attack.

5.12.2 Hyperparameters

There are three hyperparameters, the two multipliers and the lookback. There are two different multipliers used, one to calculate the speed threshold and one to calculate the heading threshold. Both have the same possible values, these being 0.01, 0.1, 1, 2, 3, 5, 7.5, 10, 25 and 50. The lookback is the number of previous points to use when calculating the reference values and the thresholds. The possible values are 5, 10, 20, 25, 50, 100, 200 and 400.

5.12.3 Implementation

Five versions were implemented, combinations of the corresponding versions of both speed and heading defences.

The first version was, as stated previously, the combination of the first versions of both speed and heading. It calculated, for each entry, the mean values of speed and heading as well as both standard deviations over the preceding entries with the number of previous entries considered being the lookback hyperparameter. The standard deviations were then multiplied with their corresponding multipliers to obtain the actual thresholds. If the difference between an entry's value and the reference one is greater than the threshold for any of the speed or heading, then the flight is considered under attack starting with that entry.

Then, the versions worked as they did in Sections 5.9, 5.10, and 5.11. The second version replaced the standard deviations by the maximum differences between any value in the series used to calculate the averages and said averages. The third version used the training set to calculate the averages and standard deviations while the fourth did the same but used maximum differences instead of standard deviations and the fifth also used maximum differences but calculated them using the parameterization dataset instead.

5.12.4 Results

Table 5.53 Combination defence results, first version

Lookback	Heading Multiplier	Speed Multiplier	Score
20	7.5	10.0	0.7536067350433643
20	10.0	10.0	0.7524743501108587
25	7.5	10.0	0.7492138767823616
100	5.0	5.0	0.7490343350157389
25	10.0	10.0	0.7480813430247182

Table 5.53 shows the results of the first version. The top configuration had 11 false alarms in normal flights, mostly windy ones though a few with more degraded conditions. It also missed 17 subtle attacks, relatively spread out between speed, deviation, and combined attacks, all of these under heavily degraded flight conditions.

Table 5.54 Combination defence results, second version

Lookback	Heading Multiplier	Speed Multiplier	Score
20	50.0	5.0	0.702381618693575
20	7.5	5.0	0.702381618693575
20	10.0	5.0	0.702381618693575
20	25.0	5.0	0.702381618693575
20	5.0	5.0	0.7023814744837464

The results of the five best configurations of the combination defence's second version are presented in Table 5.54. Two false alarms took place within the 50 normal flights of the top configuration and 28 subtle attacks were completely missed, with another one detected late. These missed attacks were mostly all of the deviation and combination attacks. Of note is the fact that all these results are essentially the same.

Table 5.55 Combination defence results, third version

Heading Multiplier	Speed Multiplier	Score
5.0	5.0	0.7692012922391416
5.0	7.5	0.7635174674333824
5.0	10.0	0.7625357536664822
7.5	5.0	0.7584453871287993
7.5	7.5	0.752253649987502

Table 5.55 displays the results of the third version. For its top configuration, there were false alarms in 17 normal flights and 11 missed attacks, where the UAV either completely failed to notice an attack or did so quite late. The missed attacks were mostly the deviation attacks while the false alarms were mostly on windy flights.

Table 5.56 Combination defence results, fourth version

Heading Multiplier	Speed Multiplier	Score
50.0	2.0	0.6769500629878419
2.0	2.0	0.6769500629878419
25.0	2.0	0.6769500629878419
10.0	2.0	0.6769500629878419
7.5	2.0	0.6769500629878419

Table 5.56 lists the fourth version's results. Its top configuration had 15 false alarms in normal flights and 20 missed subtle attacks with a further two detected late. These missed attacks were the majority of the deviation and combination attacks while the false alarms were mostly composed of windy flights. An interesting point is that all 5 configurations obtained the same results.

Table 5.57 Combination defence results, fifth version

Heading Multiplier	Speed Multiplier	Score
3.0	3.0	0.7365707714308404
3.0	5.0	0.7246586437446482
2.0	3.0	0.7243416443315295
5.0	3.0	0.7108825552513203
2.0	5.0	0.7081425297142365

Table 5.57 lists the fifth version's results. Its top configuration had 17 false alarms in the normal flights as well as 13 missed subtle attacks and a further one detected late. The false alarms were on flights with only wind for most and the false alarms were mainly composed of the deviation attacks.

5.12.5 Discussion

The results provided some interesting observations, notably that overall, using the standard deviation (as in the first and third versions) made it possible to obtain better results than using the maximum difference (as in the second, fourth and fifth versions). This is in all likelihood due to the fact that the standard deviation is less sensitive to outliers than the maximum difference, as the latter only takes into account the most extreme value. Another observation, though relatively minor, is that using a parameterization dataset improved the results obtained by a far more significant margin than what was observed in other defences' versions. The reason for this is probably that using a parameterization step helps in establishing a better threshold for both speed and heading, leading to greater improvements than what could be observed when using only one of the two.

More generally, these defences obtained results that are an improvement on those of the speed and heading defences, bearing out the idea that combining them would produce better results as said combination would be able to handle more types of attacks. However, some of the results led to questions regarding the efficacy of this defence. This is particularly true for the second and fourth versions where the only hyperparameter value that changed was the heading multiplier which used the most permissive values it could, leading to the belief that the heading was taken into account as little as possible, something that seems confirmed by the observation that the corresponding versions of the speed defence obtained extremely similar if not identical scores. The cause of this is probably the usage of the maximum difference without parameterization in these two versions, which makes them unable to be properly calibrated.

5.13 Mean Defence

5.13.1 Description

Another simple defence idea is to do as in the previous cases, but to use both the latitude and longitude instead of the distance, the heading, the speed, or a combination. Instead of creating a reference value for the distance the UAV should have moved between two successive positions, this defence creates two reference values, one for the latitude change and one for the longitude

change, as well as a threshold for each. If the change between the two latest points is too different from a reference value, say, the difference between the change and the reference value is greater than the threshold for the latitude or longitude, then the flight is considered under attack.

5.13.2 Hyperparameters

There are two hyperparameters, the lookback and the multiplier. The lookback is used to decide the length of the series used to establish the reference values and the thresholds when said length varies. Its possible values are 5, 10, 20, 25, 50, 100, 200 and 400. The multiplier is used in the threshold calculations to modify the leeway afforded to the point being tested (the same value is used for both latitude and longitude). Possible values are 0.01, 0.1, 1, 2, 3, 5, 7.5, 10, 25 and 50.

5.13.3 Implementation

As with the defences in Sections 5.9 through 5.12, five versions were implemented, following the same organization. The first version calculated the reference values as the means of the latitude and longitude changes of a series composed of the most recent datapoints whose length was defined by the lookback hyperparameter. The thresholds were calculated by multiplying the standard deviations of these series with the multipliers. The second version did the same, though it replaced the standard deviations with the maximum differences between the values in the series and the average. The third version used the training set instead of a rolling series to calculate the averages and the standard deviations. The fourth did it as well, though it used the maximum differences instead of the standard deviations. The fifth used the maximum differences as well, though it calculated them using the parameterization set instead.

5.13.4 Results

Table 5.58 Mean defence results, first version

Lookback	Multiplier	Score
50	7.5	0.7410714853666188
400	5.0	0.7397299970084549
100	5.0	0.7337200051958276
50	10.0	0.7223242342069214
200	5.0	0.7208499243817326

Table 5.58 presents the results of the first version. The top configuration incorrectly detected attacks somewhat early only 12 times out of the 50 normal flights, these being almost exclusively wind-only flights. It detected all 10 obvious attacks but missed 17 of the 40 subtler ones, almost only deviation and combined attacks.

Table 5.59 Mean defence results, second version

Lookback	Multiplier	Score
100	3.0	0.7513858103460029
25	5.0	0.7389938139768277
50	3.0	0.7384654767168728
200	3.0	0.7216450842601929
50	5.0	0.7073567512383663

Table 5.59 displays the second version's results. The best configuration only incorrectly detected attacks somewhat early during 9 of the 50 benign flights. It correctly detected all 10 obvious attacks but missed 18 of the 40 more discreet ones, with a nineteenth that took some time to be detected. These detailed results are similar to the previous version, with false alarms in wind-only flights and missed deviation and composite attacks.

Table 5.60 Mean defence results, third version

Multiplier	Score
5.0	0.7656601866388013
7.5	0.7278321404046153
10.0	0.7210955095122569
25.0	0.6445363452103672
50.0	0.5794848325812625

Table 5.60 shows the results of the third version. There were 18 mistaken detections in the 50 benign flights for the top configuration, while only 10 failed detections, with none of the missed attacks being obvious ones. These false alarms were both wind-only flights and ones under heavily affected conditions while the missed attacks were a few speed ones under heavily degraded conditions and the deviation attacks.

Table 5.61 Mean defence results, fourth version

Multiplier	Score
2.0	0.7490628987174253
3.0	0.7273440039081598
5.0	0.6736101338138273
7.5	0.6469238897336311
10.0	0.636452380778422

The fourth version's results are displayed in Table 5.61. Here, 17 benign flights were mistakenly labeled as under attack by the best configuration while only 12 attacks either took a long time to be detected or were not at all. These results were almost the same as those of the previous version, false alarms on wind-only and heavily affected flights as well as a few missed speed attacks in heavily degraded flight conditions and almost all deviation attacks missed.

Table 5.62 Mean defence results, fifth version

Multiplier	Score
3.0	0.7229063434092101
2.0	0.6941884255861066
5.0	0.6856631764833035
10.0	0.6499151811289814
7.5	0.6373346466184286

The results of the fifth version are listed in Table 5.62. For the top configuration, amongst the benign flights, 16 had attacks incorrectly detected while 16 attacks were missed, none of them obvious ones. Its results are also quite similar to the two previous versions, with false alarms in wind-only and in heavily affected flights as well as a few missed speed and composite attacks in bad conditions and almost all deviation attacks missed.

5.13.5 Discussion

This defence obtained better results than the distance, speed and heading defences, indicating that the idea of using both longitude and latitude indeed improved upon these previous defences' results as changes in deviation or speed automatically affect latitude and longitude. That being said, this

defence was still deficient in handling the deviation attacks which were mostly if not completely missed by all versions. More work is thus required to further refine it.

Another observation is the fact that using the training set seems to often enable better results as the third and fourth versions attest compared to the first and second. This is in all likelihood due to the ability to use a bigger series that enables better reference values and thresholds estimation. The lower results of the fifth version reinforce this interpretation as it used the parameterization dataset for the thresholds, thus had less data for both the reference values and the thresholds.

As first noted in Subsection 5.9.5 of the distance defence, the lookback parameter seems to have a lesser impact on the defences' scores than the multiplier in the first two versions as over the top 5 results in both versions, the multipliers varied far less than the lookback, indicating that the top results are more concentrated around specific multiplier values than specific lookback values. This is probably due to the fact the multiplier has a far more direct impact on the leniency of the thresholds than the lookback, whose sole impact is indirect through the size of the series used to calculate them. That being said, as noted in Subsection 5.9.5, it is not without impact, unlike for the linear regression discussed in Section 5.4.

CHAPTER 6 DISCUSSION

Now that the results for each defence have been discussed, a more comprehensive comparison is in order. This chapter will tackle exactly that, beginning with the general comparison of the different versions of each defence and the observations that can be made, with the second section covering some important precisions regarding these results. Afterwards, it will cover the different implementation choices, such as the use of PCA, of the parameterization step, and others, to discuss if a pattern regarding their usefulness appears. The next section will then discuss the limitations of this work, whether regarding the datasets, the implementations, or others while the final one will deal with some possible avenues for future research.

6.1 Comparison

The following table displays the best result of each defence, indicating for each its score. The baseline is also displayed as an added marker for comparison. A more detailed table (Table B.1) can be seen in appendix B displaying the results for each version.

Table 6.1 Overall Results

Name	Score
Autoencoder	0.8162065503688939
LOF	0.8087331786192506
Combination	0.7692012922391414
Mean	0.7656601866388012
OCSVM	0.7592631338347487
Linear Regression	0.7583316200872077
EKF	0.7514238808137792
Speed	0.7029087676466456
Distance	0.6980375711266434
Heading	0.6458865480504737
Baseline	0.5984912541439668
LSTM	0.582892955734885
Linear Acceleration	0.5314006509821438

The results obtained can be separated into three parts, these being those that severely failed at the task with results worse than the baseline, those that improved upon the baseline but did not beat the best modifications possible for the EKF, and those that clearly improved upon the UAV's built-in defences.

The first group is made up of the LSTM and linear acceleration defences which could not improve upon the baseline. They are defences that are quite clearly inferior to what is currently implemented on the UAV and should thus not be considered for future work.

The second category is composed of the speed, distance and heading defences. While these improved upon the baseline, they obtained worse results than the best version of EKF. This means that a small modification to the EKF, a defence already implemented onboard the UAV, would outperform them thus making them of lesser interest for future work. These defences are, by their very nature, not meant to detect many of the attacks, something that explains why an improvement upon the built-in EKF could obtain better results than them.

The third group corresponds to the defences that surpassed the EKF's best result. These are composed of the linear regression, OCSVM, mean, combination, LOF and autoencoder defences. These managed to detect far more attacks than the baseline while limiting the number of false alarms, making them quite interesting for further study. They are defences that are meant to be able to handle the whole range of possible attacks, whether that be because they use multiple parts of the data entry to handle both distance and deviation attacks or instead because they can take the whole data entry into account.

Amongst this third group, the best defence was the autoencoder, specifically using only the GNSS log, with 7 features for the PCA, an internal layer size of 2, a multiplier value of 3, 1000 epochs and a learning rate of 0.001. It achieved the best results out of all the flights, detecting almost every attack and managing to keep its number of false alarms relatively low.

An interesting observation is that some rather simple defences such as the combination and mean defences managed to somewhat outperform more complex ones like the OCSVM, linear regression and, by far, the LSTM. This is probably because the combination and mean defences are more suited to the current task than them. The fact these two outperformed the linear regression defence is very intriguing as they are extremely similar in principle to it, particularly the mean defence. The reasons behind this might very well warrant further study.

6.2 Important Nuances

This section discusses some important nuances regarding the results summarized in the previous section, highlighting why, even though the second version of the autoencoder defence obtained the best results with the hyperparameter values described hereabove, some other values or even some other defences might be preferable.

6.2.1 Sensitivity of Defences

An important point to note regarding the ranking established in the previous section is that the best results displayed provide an idea of the maximum score a specific version can obtain and that some slight changes in the values of some hyperparameters can still provide good results.

This is especially important regarding the inevitable trade-off between enhancing the detection of attacks and keeping the number of false alarms as low as possible. As such, should some applications require a lower rate of false alarms and would be willing to sacrifice some of the detection accuracy for it, the usage of different values for the hyperparameters, or even entirely different defences, might be preferable.

6.2.2 Time and Power Costs Concerns

Another point that is also important to consider when choosing which defence to use is the costs in computation time and power to use the defence during a flight. Though these were not measured in the current project, they are nonetheless important factors that should be taken into account when selecting a defence to implement on an actual drone. These factors might very well justify using the mean defence instead of the autoencoder defence as the slightly lower score of the mean defence might be offset by probably lesser power and computing needs.

6.2.3 Flight Plan

Another important nuance is that the projected flight plan can potentially highly affect the performance of many of the defences. For example, suppose a flight plan consists in flying to multiple successive destinations, one after the other. Such a flight would not be a straight line, unlike those tested here. While some defences could be adapted by considering each leg of the flight in isolation, this would require for many of them to train their defences once more during 100 seconds at the beginning of each of these sub-flights. However, this 100-second-long training

phase was predicated upon the assumption that there would be no attack during this period. While this assumption was made for the beginning of the flight in this work, it would be harder to justify for every stop along the flight plan as well.

Some flight plans could also require turns during flight, something that only a few of the defences, namely the EKF-based ones, the linear acceleration, as well as the speed and distance ones might theoretically be able to handle. However, none of these have been tested in such a scenario here. Some other flight plans might need to have changes in UAV speed during the flight, something that multiple defences are also in all likelihood not able to handle. All in all, these possible variations might require using a defence other than the one with the best score, as it might not fit the flight plan.

6.2.4 Training Phase

A further point that the previous subsection leads to is also quite important, namely whether training at the beginning of the flight is doable or not in the context considered.

Indeed, there are some cases such as the ones mentioned hereabove where the flight plan implies that the UAV will not be in the same situation for the whole flight. This means that defences that use a training phase to have an unchanging reference to compare with the UAV's current state for the whole flight must be modified to be able to update their reference at certain stages in the flight, as the expected state of the UAV also changes. In such cases, the defences that use running reference values, such as the first two versions of the distance, speed, heading, combination, and mean defences, generally have the advantage that these reference values require less time to be created than what would be needed for a whole training phase.

In a similar vein, there are times when the flight conditions change during the flight, whether these be because of the weather or other factors. In these situations, especially if these flight condition changes are progressive, it is much harder to determine a precise point at which a new training phase must take place. Detection techniques that use changing reference values such as the defences noted in the previous paragraph may very well be better suited to the task than the top scoring one as they can adapt to these changes.

6.2.5 Flight Dataset Considerations

Another important nuance to consider is the fact that the results obtained here are skewed by the makeup of the flight dataset. Notably, this dataset contains as many spoofed flights as benign ones, a ratio that is not necessarily representative of reality. This question of ratio also applies to the types of attacks, especially considering the fact that for an external attacker spoofing the UAV to imply a change of speed or heading without modifying the other is difficult.

An interesting point to study is the parts of the flights where the defences most often failed, whether that be because of false alarms or because attacks missed. These parts are mostly the flights 75 to 80, flights 89 to 92 and flights 43, 44, 45, 47, 49 and 50 as well as a few other more isolated ones such as flights 98 and 100. All of these flights are ones with heavily degraded conditions, which explains why the defences had difficulties flying without false alarms and detecting attacks. However, while there is a simple explanation for these flights, another series was also challenging to handle but more difficult to explain. These are flights 6 through 15 as well as flight 34. They often had more false alarms than other benign flights even though they only had wind in their flights, and half of them were without turbulences. The most probable explanation is that these were almost all long flights and thus with the added wind, the likelihood of encountering an outlier that the defence could not handle at some point during the flight was quite high.

6.3 Implementation Choices

This section deals with some of the choices that need to be made when implementing most of these defences, such as whether to have a parameterization step or not, whether to use PCA, and others. It will analyze these steps' respective usefulness, though focusing on their usefulness with the best performing versions.

6.3.1 Parameterization Step

One of the choices to make is whether to use a parameterization step. Doing so means that the data used to create thresholds is different from the one used to establish reference values. This also implies that less data is available to calculate the reference values as some of it is necessary to create the thresholds.

The use of this parameterization step has somewhat of a mixed track record. It improved the results in some cases, for example adding that step improved the ninth version of LOF into the tenth, as

well as the fourth versions of the heading and distance defences enhanced as the fifth versions. However, it also degraded the performance of other versions, notably the eleventh LOF version into the twelfth, or the fourth speed version into the fifth one.

In any case, these changes in performance are somewhat minor, around two to three points of percentage at most. As there is no clear answer emerging from the results, the best advice that can be given is to test both and, if that is not possible, refer to the results obtained here to see if it helped for the specific version implemented.

6.3.2 Datapoints or Difference

Another question regarding implementation is whether to use the successive data entries or the changes between said successive data entries instead. Out of the defences implemented which used both, their versions using the changes significantly outperformed those that used the data entries themselves. The seventh and eighth versions of LOF outperformed the fifth and sixth by a wide margin, much like the seventh and eighth versions of OCSVM also outperformed the corresponding fifth and sixth versions.

Amongst the other defences, the vast majority used only the differences and not the data entries themselves. This further highlights the point that using these differences permits a greater performance than would be possible with the datapoints themselves.

6.3.3 Principal Component Analysis

PCA was used in multiple defences, notably the autoencoder, LOF and OCSVM. In all of these cases, it generally did not seem to particularly help the performance of the defences. This can be observed from noting how the top results from these defences often do not use it (they use -1 features), notably the LOF's eleventh version, almost the top performing one.

A further marker of this is the fact that within the top 5 of versions using PCA, its number of features is often the hyperparameter which varies the most, if not the only one, indicating its lack of relative impact compared to the other hyperparameters. After all, if it had a major impact then a change in its value, while keeping the other hyperparameter values identical, would have a greater impact on the score. All of these points thus clearly highlight how the PCA's impact on the defences' performance is quite limited. This is probably due to the limited dimensionality of the input logs which makes PCA's dimensionality reduction unnecessary.

6.3.4 Multiplier-Based Thresholds

Multiplier-based thresholds are meant here as those used in the OCSVM and LOF defence versions where they replace the default classification algorithm. They proved quite useful in highly enhancing the results of their defences, as can be seen by comparing the results obtained by the fifth and ninth LOF versions to the corresponding third and fourth ones. The only difference between these pairs is the replacement of the default classification algorithm by a threshold calculated by multiplying the maximum error observed in the training set with a multiplier hyperparameter. This made it possible to obtain far better results, moving from below 0.1 to above the baseline.

A similar observation can be made with the OCSVM defence and the same versions, going from less than 0.25 to above 0.6. All in all, the use of a multiplier-based threshold highly improved the results obtained. This is in all likelihood because these multiplier-based thresholds afford more flexibility and options than the built-in one as modifying the value of the multiplier directly affects the defence's tolerance of outliers.

6.4 Limitations

While all these observations provide useful insights some care must be taken before applying them in the real world as there are limitations to the current work that must be taken into account.

6.4.1 Simulator Limitations

There were some important limitations with the simulator, notably that the GNSS signals were not fully simulated and that they were always sent with the same frequency.

The GNSS signals were not fully simulated, as some of their values were fixed no matter the flight conditions. For example, the number of satellites was a fixed value that did not change over the course of the flight, which made some defences impossible to implement in the current project. This was, however, not as great of a problem as it might seem, as the focus of this work was more on the GNSS coordinates than on the signal itself.

The second problem is that a new GNSS signal arrived roughly every 200 milliseconds, with some extremely rare exceptions. This means that defences could not really be tested on their ability to

handle missed GNSS signals which might unduly advantage defences such as the distance, heading or mean ones.

6.4.2 Dataset Limitations

There are also some important limitations regarding the dataset and its contents, notably the fact that all flights are composed of a single straight line, that the flight conditions do not change over time (with the exception of wind when turbulences are added) and that the custom logs cannot capture enough precision regarding GNSS position.

The choice to make all flights a straight line from departure to destination was based on multiple factors. The first reason was that flights within the simulator were done by providing a target location to reach which made flights with turns impossible to implement in a satisfying way. The second was that the great majority of the defences were not designed to handle flights with turns. The goal of this project was to see which defences were most able to detect attacks and thus the focus was specifically on the attacks, their specificities, and subtleties, and not on whether the defences could handle different trajectories.

The fact that the flight conditions did not change over time was a choice made as implementing such changes without making them abrupt was not truly possible with the simulator and the changes would thus have been jumps from one value to another. Having flight conditions change during a flight would have made the analysis more complex and would not have particularly improved the results as an attack can only begin once in a flight and the goal of the project is to see how good the defences are at detecting said attacks. Two shorter flights with unchanging conditions would have been more interesting to that effect than one with changing conditions. Thus, the dataset does not include flights with changing conditions, except for winds when turbulences are added.

Another problem is that the GNSS positions were spoofed directly when and where they were received by the UAV autopilot. This was done to make sure no subsystem would use the non-spoofed values. Since these received values are not precise enough and often repeat themselves for successive points, they are automatically refined further within their lifecycle in the autopilot. However, the custom log for the spoofing was created at the specific point of interception, it thus could not benefit from this refining which consequently made this log unwieldy for directly testing defences with it. Even though the defences were able to use the UAV's standard GNSS log instead of the custom one, this still represents a limitation of the current work.

A final limitation of the dataset is the fact that the data comes from a simulator and not from real flights. While there were multiple reasons for this decision detailed in Chapter 4, the dataset would have obviously been more realistic if the data had come from real flights.

6.4.3 Implementation Limitations

A problem with the comparison resides with the metric used, more precisely in that it is not truly able to correctly handle cases where the UAV is teleported to the destination which is what happens in flights 59 and 60. The problem is that defences often detect an attack when they believe they have reached their destination as the UAV stops moving. It thus becomes impossible to differentiate between a defence that detected an attack because the UAV was teleported and a defence that detected an attack because the UAV stopped moving. Though this is a problem, it only applies to the two aforementioned flights out of a hundred.

Another limitation is more specific to the OCSVM defence. It is the fact that the PCA could not be applied on some of its versions, making it impossible to assess their full potential. Even though this only took place on a few implementations which were not expected to be the most performing ones when considering the corresponding LOF defence versions, this is still a limitation of the current work.

6.4.4 Comparison Limitations

Though the different results of the defences were compared, there are other factors that would need to be considered before implementing these defences in real life, factors that were not quantified here.

Amongst these factors are the costs in computing time and in power of the defences. These can have an important impact on the viability of a defence. After all, what good is a defence if it is not able to come to a decision regarding whether the new signal is an attack before the next one arrives?

Another point to take into consideration is the fact that spoofing was done in the simulator by directly modifying the values when they were received by the autopilot. This provides a hypothetical attacker with a degree of control and precision over the attack that may be unlikely in real life when one considers factors such as the movement of the attacker if spoofing the UAV from a distance, say, when in a car following the UAV, for example. Should the attacker manage

to replicate the process followed here to spoof the UAV, then the limited computing capabilities of the UAV constitute a different limitation that must be taken into consideration.

6.5 Future Research

There are multiple possible approaches to further the research detailed here, whether that be by enhancing the defences tested here, testing those that can predict future positions of the UAV, or handling some of the limitations listed previously.

6.5.1 Defence Enhancements

Though the defences were almost always implemented in multiple versions, there are still some possible areas of improvement to create better ones.

An obvious idea to do so is to use bagging techniques to combine the existing defences and try to create a better one. This was done with the combination defence that, as its name indicated, combined two defences, the speed and heading ones, to try and achieve better results, something it succeeded in doing. This could however be furthered by continuing such work, for example combining the mean and combination defences and signalling an attack only if at least two of the four elements tested (the latitude change, the longitude change, the heading, and the speed) obtained abnormal results.

It would even be possible to enhance this further by testing less closely related defences such as the linear regression and autoencoder defences together. While these combinations might very well not produce interesting results, perhaps they will and are a possible option to test.

6.5.2 Position Prediction

One of the possible future works would be to test some of the defences' ability to substitute themselves to GNSS. Indeed, the linear regression and LSTM defences were conceived to be able to also predict the UAV's position in the paper that proposed them [12] and some other defences tested here, such as the mean and combination defences, could be modified to attempt it as well.

This could be done with another comparison similar to the current one except for the fact that the defences, instead of trying to detect attacks, attempt to predict the successive UAV positions based upon their models. The metric would be the distance from the actual final position. This could be

done with the current dataset, offering an improvement on the paper that proposed the linear regression as it only tested it with a single flight [12].

6.5.3 Fixing the Limitations

A third and somewhat obvious avenue for future research is to focus on handling the limitations of the current work. This could be done in multiple ways, from testing the energy consumption and computing costs of the proposed defences, to verifying the flight dataset correctly mirrors real-life scenarios, or handling any of the different issues noted in the previous section.

CHAPTER 7 CONCLUSION

UAVs are an important emerging technology with wide-ranging applications. However, they are quite vulnerable to multiple types of attacks. Amongst these, GNSS spoofing is a type of attack that aims at making the UAV deviate from its trajectory by leading it to believe falsified GNSS messages. Multiple defences have been created to handle these attacks; however, they have rarely been compared to one another and there is no dataset that can be used for that purpose. This project fills this gap.

A dataset of one hundred flights was created through computer simulation in order to be used as a common basis for the assessment and comparison of defences. It is composed of flights with diverse departures and destinations as well as varied lengths. Multiple types of attacks are represented, from the obvious teleportation to the subtler deviation. The flight conditions were also modified to simulate flights in degraded conditions, whether that be from wind or noise in GNSS sensors as well as in any of the accelerometer, barometer, gyroscope, or others. All of this was done to obtain a dataset both large and realistic enough for defence evaluation.

This dataset was used to assess the defences used in this project. These were an extended Kalman filter (EKF), an autoencoder, a linear regression, a long short-term memory network (LSTM), a linear acceleration, a local outlier factor (LOF), a one-class support vector machine (OCSVM), as well as the distance, speed, heading, combination, and mean defences. Generally, each had at least three if not more different versions that modified the basic functioning implemented. They were all assessed by applying them onto the flight dataset to see how each would perform and find out the reasons for these performances.

The results obtained were compared, finding that the autoencoder defence performed best. The nuances, implementation choices, and limitations of the results were then analyzed to better understand them.

Finally, some interesting research avenues, such as enhancing the defences or working on position prediction when GNSS data is unreliable, were proposed.

REFERENCES

- [1] U. Ukaegbu, L. Tartibu, and M. Okwu, "Unmanned Aerial Vehicles for the Future: Classification, Challenges, and Opportunities," presented at the 2021 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), Durban, South Africa, 2021.
- [2] R. Altawy and A. M. Youssef, "Security, Privacy, and Safety Aspects of Civilian Drones: A Survey," (in English), *Acm Transactions on Cyber-Physical Systems*, vol. 1, no. 2, pp. 1-25, Feb 2017.
- [3] J. P. Yaacoub, H. Noura, O. Salman, and A. Chehab, "Security analysis of drones systems: Attacks, limitations, and recommendations," in *Internet of Things*, 2020, vol. 11.
- [4] S. Iqbal, "A Study on UAV Operating System Security and Future Research Challenges," presented at the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), NV, USA, 2021.
- [5] A. Oruc, "Potential cyber threats, vulnerabilities, and protections of unmanned vehicles," (in English), *Drone Systems and Applications*, vol. 10, no. 1, pp. 51-58, Jan 4 2022.
- [6] C. G. L. Krishna and R. R. Murphy, "A Review on Cybersecurity Vulnerabilities for Unmanned Aerial Vehicles," presented at the 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Shanghai, China, 2017.
- [7] A. Y. Javaid, W. Sun, V. K. Devabhaktuni, and M. Alam, "Cyber Security Threat Analysis and Modeling of an Unmanned Aerial Vehicle System," presented at the 2012 IEEE Conference on Technologies for Homeland Security (HST), Waltham, MA, USA, 2012.
- [8] M. V. Mamchenko, "Analysis of Control Channel Cybersecurity of the Consumer-Grade UAV by the Example of DJI Tello," *Journal of Physics: Conference Series*, vol. 1864, no. 1, 2021.
- [9] D. J. He *et al.*, "A Friendly and Low-Cost Technique for Capturing Non-Cooperative Civilian Unmanned Aerial Vehicles," (in English), *Ieee Network*, vol. 33, no. 2, pp. 146-151, Mar-Apr 2019.
- [10] N. Shijith, P. Poornachandran, V. G. Sujadevi, and M. M. Dharmana, "Spoofing technique to counterfeit the GPS receiver on a drone," presented at the 2017 International Conference on Technological Advancements in Power and Energy (TAP Energy), Kollam, India, 2017.
- [11] D. J. He, S. Chan, and M. Guizani, "Communication Security of Unmanned Aerial Vehicles," (in English), *Ieee Wireless Communications*, vol. 24, no. 4, pp. 134-139, Aug 2017.
- [12] L. Meng *et al.*, "An Approach of Linear Regression-Based UAV GPS Spoofing Detection," *Wireless Communications and Mobile Computing*, vol. 2021, p. 5517500, 2021/05/07 2021.
- [13] G. Aissou, H. O. Slimane, S. Benouadah, and N. Kaabouch, "Tree-based Supervised Machine Learning Models For Detecting GPS Spoofing Attacks on UAS," presented at the 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2021.

- [14] S. Panzieri, F. Pascucci, and G. Ulivi, "An outdoor navigation system using GPS and inertial platform," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 2, pp. 134-142, 2002.
- [15] G. Panice *et al.*, "A SVM-based detection approach for GPS spoofing attacks to UAV," presented at the 2017 23rd International Conference on Automation and Computing (ICAC), Huddersfield, UK, 2017.
- [16] M. Ahmad, M. A. Farid, S. Ahmed, K. Saeed, M. Asharf, and U. Akhtar, "Impact and Detection of GPS Spoofing and Countermeasures against Spoofing," presented at the 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2019.
- [17] S. P. Arteaga, L. A. M. Hernandez, G. S. Perez, A. L. S. Orozco, and L. J. G. Villalba, "Analysis of the GPS Spoofing Vulnerability in the Drone 3DR Solo," (in English), *Ieee Access*, vol. 7, pp. 51782-51789, 2019.
- [18] S. Z. Khan, M. Mohsin, and W. Iqbal, "On GPS spoofing of aerial platforms: a review of threats, challenges, methodologies, and future research directions," *PeerJ Comput Sci*, vol. 7, p. e507, 2021.
- [19] M. R. Manesh, J. Kenney, W. C. Hu, V. K. Devabhaktuni, and N. Kaabouch, "Detection of GPS Spoofing Attacks on Unmanned Aerial Systems," presented at the 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2019.
- [20] M. L. Psiaki and T. E. Humphreys, "GNSS Spoofing and Detection," (in English), *Proceedings of the Ieee*, vol. 104, no. 6, pp. 1258-1270, Jun 2016.
- [21] H. Wen, P. Y.-R. Huang, J. Dyer, A. Archinal, and J. Fagan, "Countermeasures for GPS Signal Spoofing," presented at the Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2005), Long Beach, CA, 2005.
- [22] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned Aircraft Capture and Control Via GPS Spoofing," (in English), *Journal of Field Robotics*, vol. 31, no. 4, pp. 617-636, Jul-Aug 2014.
- [23] D. Wang and D. Feng, "Research on the Strategy of Drones Intrusion Detection Based on Game Theory," presented at the 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2018.
- [24] Q. W. Wang, T. Nguyen, K. Pham, and H. Kwon, "Mitigating Jamming Attack: A Game-Theoretic Perspective," (in English), *Ieee Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6063-6074, Jul 2018.
- [25] L. Chaari, S. Chahbani, and J. Rezgui, "MAV-DTLS toward Security Enhancement of the UAV-GCS Communication," presented at the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), Victoria, BC, Canada, 2020.

- [26] R. D. Pietro, G. Oligeri, and P. Tedeschi, "JAM-ME: Exploiting Jamming to Accomplish Drone Mission," presented at the 2019 IEEE Conference on Communications and Network Security (CNS), Washington, DC, USA, 2019.
- [27] Z. Feng *et al.*, "Efficient drone hijacking detection using onboard motion sensors," presented at the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 2017.
- [28] Z. Feng *et al.*, "An Efficient UAV Hijacking Detection Method Using Onboard Inertial Measurement Unit," *ACM Transactions on Embedded Computing Systems*, vol. 17, no. 6, pp. 1-19, 2018.
- [29] E. Basan, A. Basan, A. Nekrasov, C. Fidge, N. Sushkin, and O. Peskova, "GPS-Spoofing Attack Detection Technology for UAVs Based on Kullback-Leibler Divergence," (in English), *Drones*, vol. 6, no. 1, p. 8, Jan 2022.
- [30] P. Jiang, H. Wu, and C. Xin, "DeepPOSE: Detecting GPS spoofing attack via deep recurrent neural network," *Digital Communications and Networks*, 2021.
- [31] J. Whelan, A. Almeahmadi, and K. El-Khatib, "Artificial intelligence for intrusion detection systems in Unmanned Aerial Vehicles," (in English), *Computers & Electrical Engineering*, vol. 99, no. 107784, Apr 2022.
- [32] Y. Qiao, Y. Zhang, and X. Du, "A Vision-Based GPS-Spoofing Detection Method for Small UAVs," in *2017 13th International Conference on Computational Intelligence and Security (CIS)*, 2017, pp. 312-316.
- [33] A. Eldosouky, A. Ferdowsi, and W. Saad, "Drones in Distress: A Game-Theoretic Countermeasure for Protecting UAVs Against GPS Spoofing," (in English), *Ieee Internet of Things Journal*, vol. 7, no. 4, pp. 2840-2854, Apr 2020.
- [34] N. T. Chibi, H. E. Ghazi, and W. F. Fihri, "Drone Cyber-Attack: An Intrusion Detection Technique Based on RSSI and Trilateration," presented at the 2021 Third International Conference on Transportation and Smart Technologies (TST), Tangier, Morocco, 2021.
- [35] R. Hartley, B. Kamgar-Parsi, and C. Narber, "Using Roads for Autonomous Air Vehicle Guidance," (in English), *Ieee Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3840-3849, Dec 2018.
- [36] A. R. Vetrella and G. Fasano, "Cooperative UAV navigation under nominal GPS coverage and in GPS-challenging environments," presented at the 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), Bologna, Italy, 2016.
- [37] Z. Z. M. Kassas, J. Khalife, K. Shamaei, and J. Morales, "I Hear, Therefore I Know Where I Am: Compensating for GNSS Limitations with Cellular Signals," *IEEE Signal Processing Magazine*, vol. 34, no. 5, pp. 111-124, 2017.
- [38] J. J. Khalife, S. Bhattacharya, and Z. M. Kassas, "Centimeter-Accurate UAV Navigation with Cellular Signals," in *Proceedings of the 31st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2018)*, Miami, Florida, USA, 2018, pp. 2321 - 2331.

- [39] A. R. Vetrella and G. Fasano, "Cooperative UAV navigation under nominal GPS coverage and in GPS-challenging environments," presented at the 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), Bologna, Italy, 2016.
- [40] J. Whelan, T. Sangarapillai, O. Minawi, A. Almeahmadi, and K. El-Khatib, "UAV Attack Dataset," ed. IEEE Dataport, 2020.
- [41] X. Wei, Y. Wang, and C. Sun, "PerDet: Machine-Learning-Based UAV GPS Spoofing Detection Using Perception Data," vol. 14, no. 19, p. 4925, 2022.
- [42] T. E. Humphreys, J. A. Bhatti, D. P. Shepard, and K. Wesson, "The Texas Spoofing Test Battery: Toward a Standard for Evaluating GPS Signal Authentication Techniques," in *ION GNSS 2012*, Nashville, TN, 2012, pp. 3569-3583.
- [43] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6235-6240.
- [44] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2004, vol. 3, pp. 2149-2154 vol.3.
- [45] "ArduPilot," ed. Online, 2009.
- [46] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007.
- [47] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, p. 5, 2011.
- [48] A. Paszke *et al.*, "PyTorch: an imperative style, high-performance deep learning library," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*: Curran Associates Inc., 2019, p. Article 721.

APPENDIX A VERSION GRAPHS

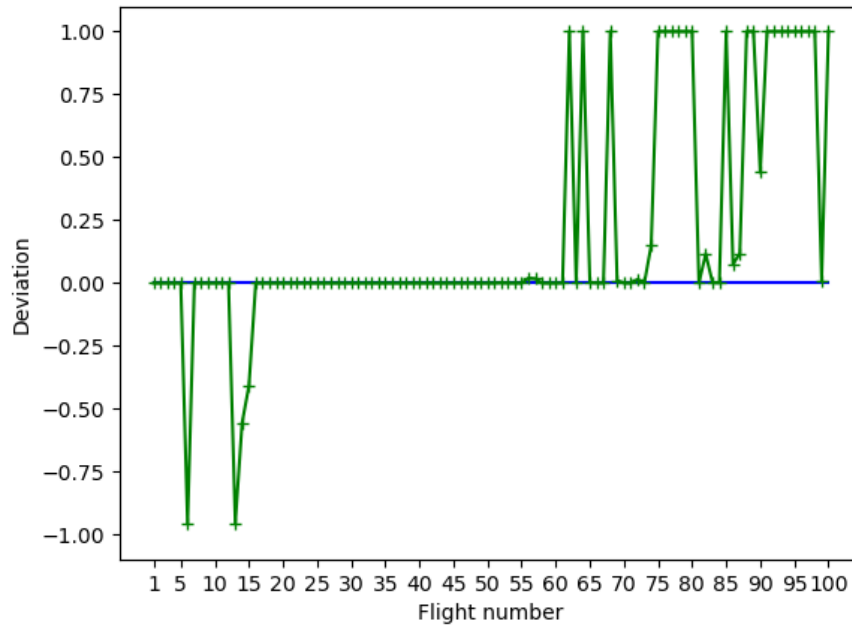


Figure A.1 EKF defence results, second version

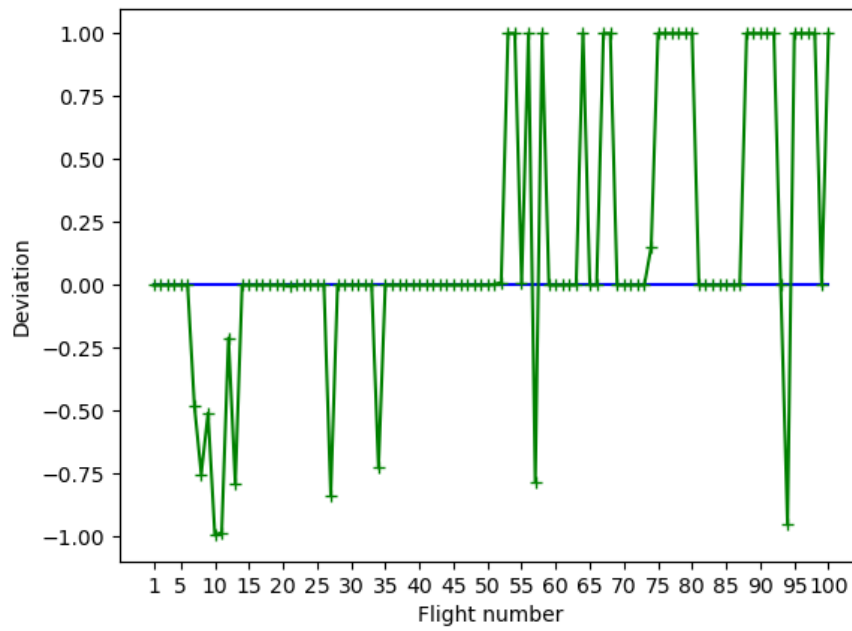


Figure A.2 EKF defence results, third version

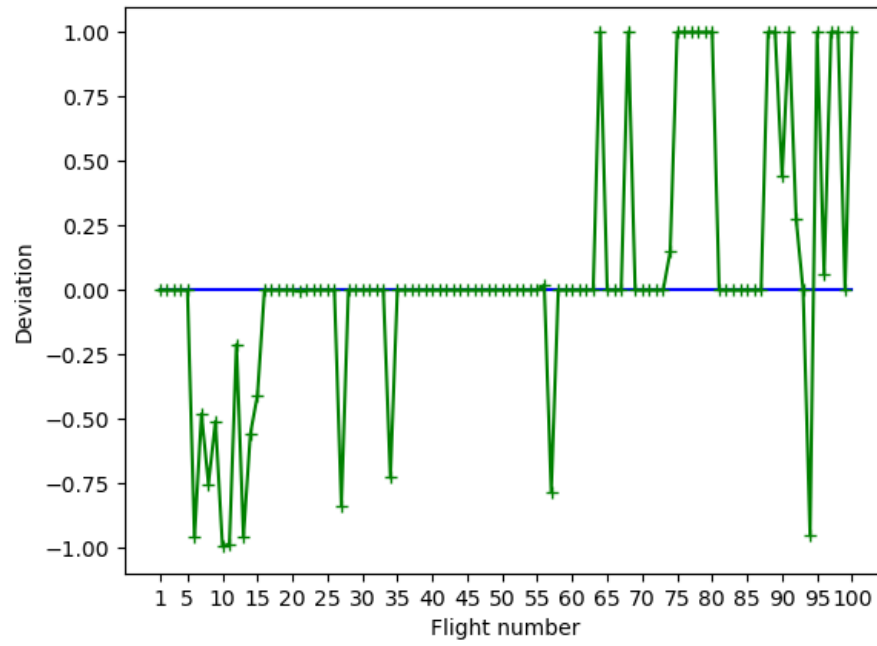


Figure A.3 EKF defence results, fourth version

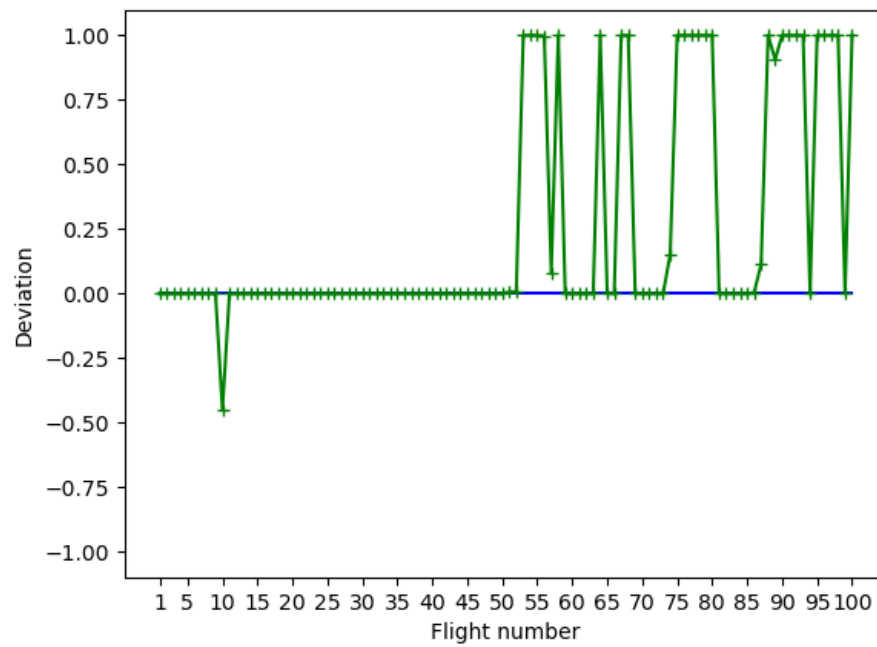


Figure A.4 EKF defence results, fifth version

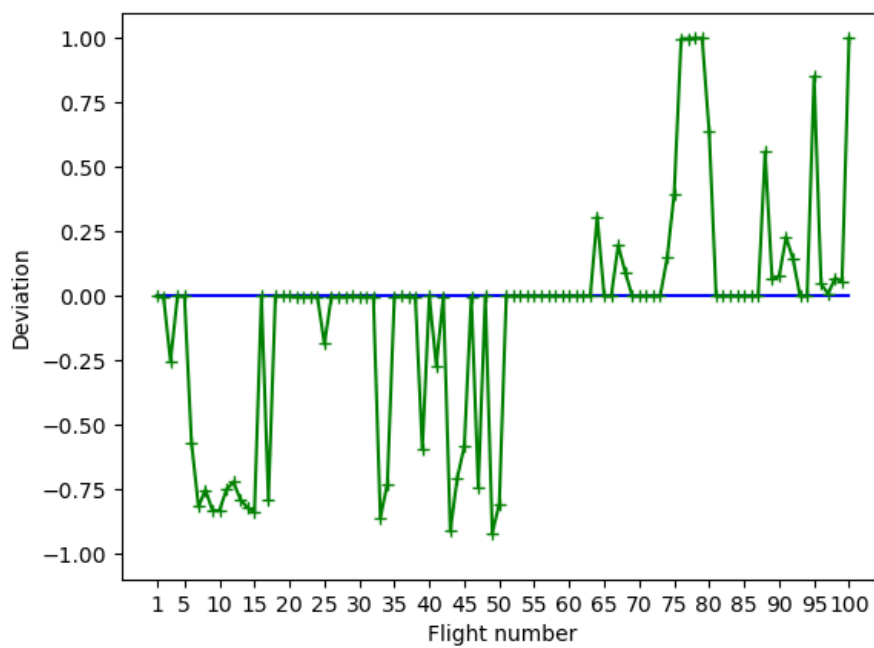


Figure A.5 Autoencoder defence results, first version

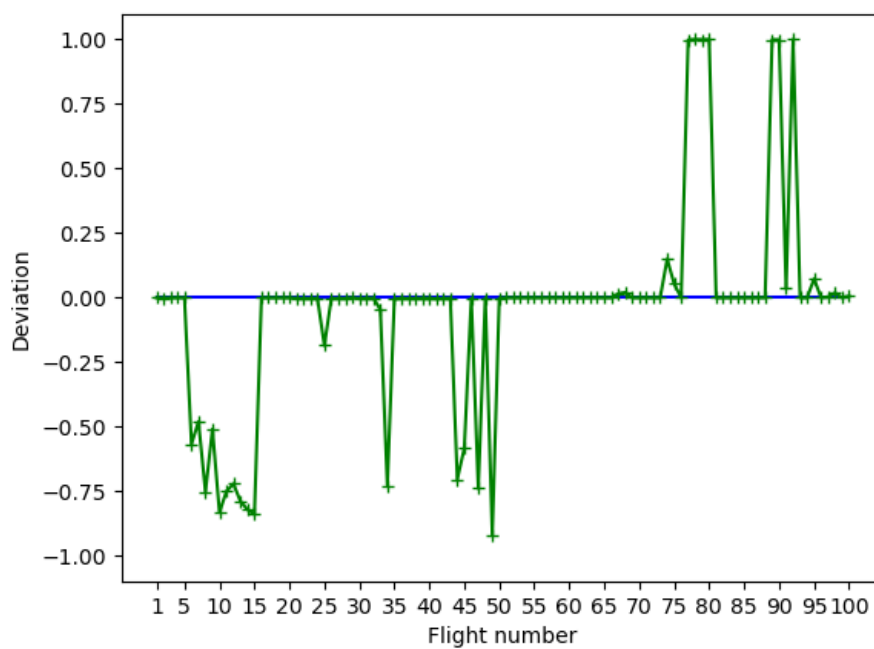


Figure A.6 Autoencoder defence results, second version

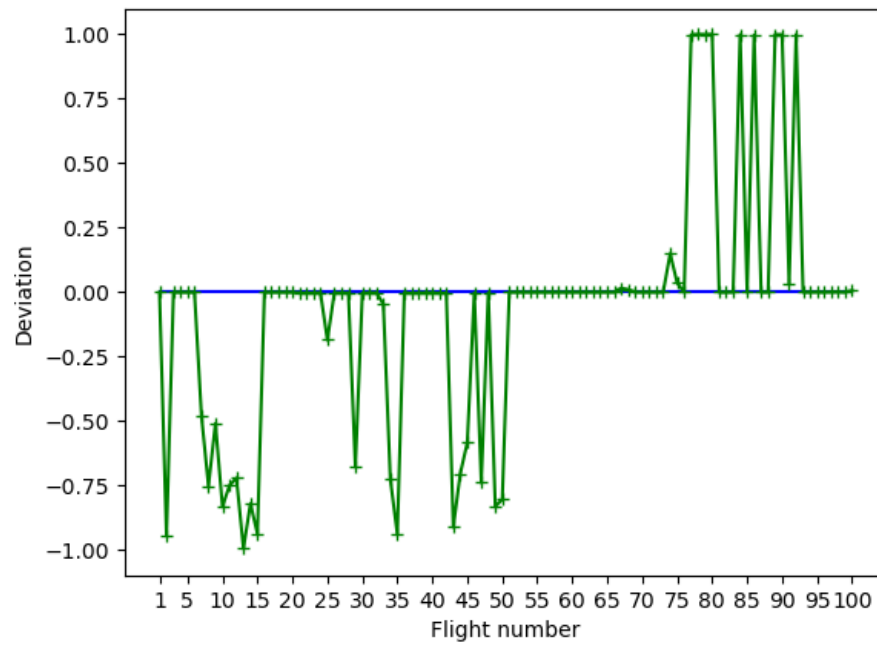


Figure A.7 Linear regression defence results, first version

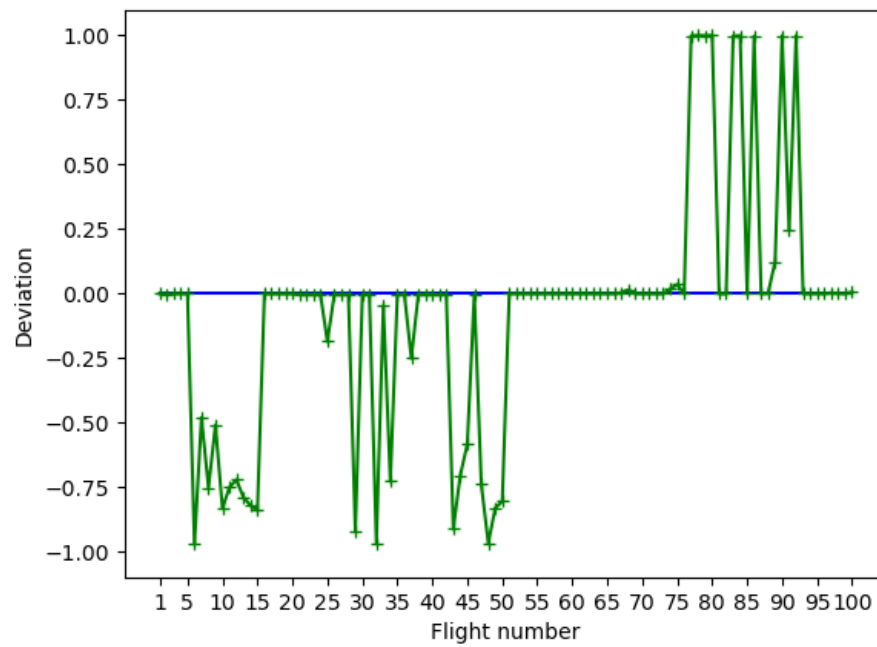


Figure A.8 Linear regression defence results, second version

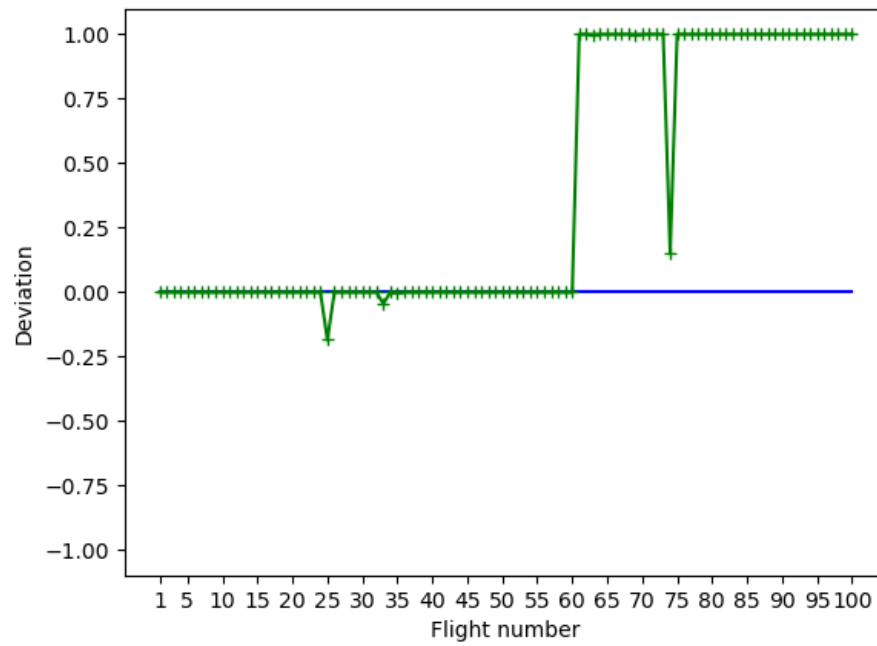


Figure A.9 Linear regression defence results, third version

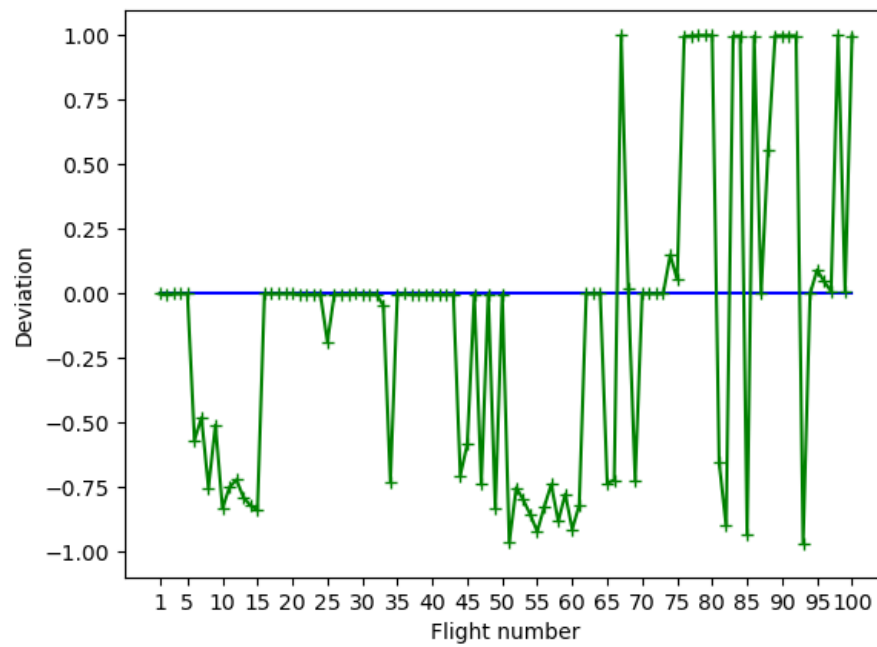


Figure A.10 LSTM defence results

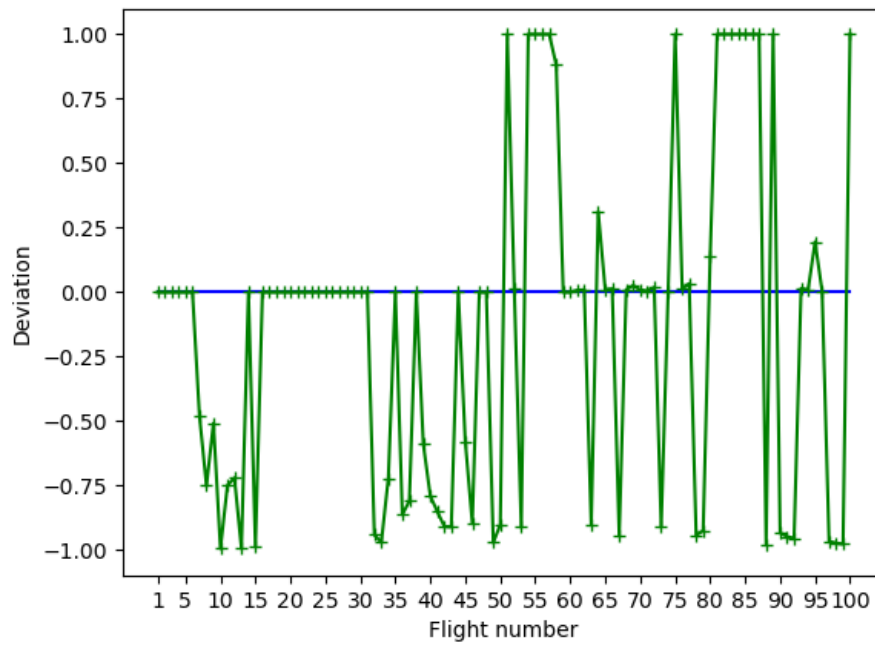


Figure A.11 Linear acceleration defence results

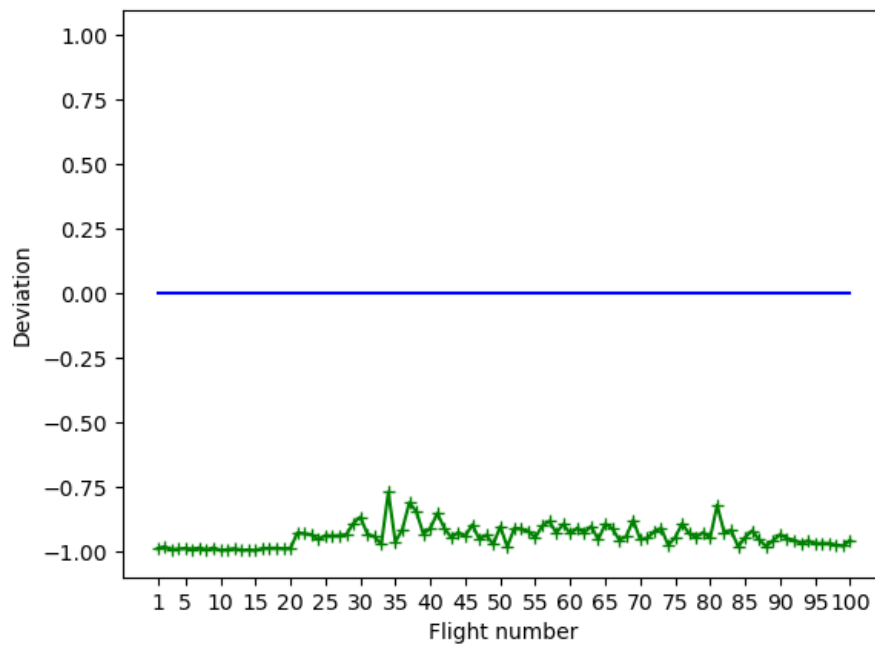


Figure A.12 LOF defence results, first version

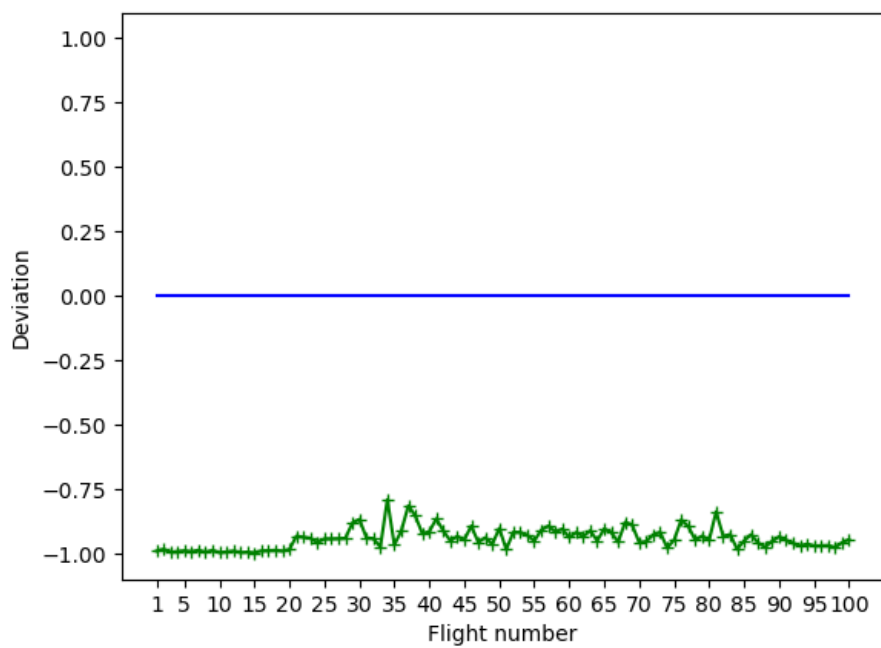


Figure A.13 LOF defence results, second version

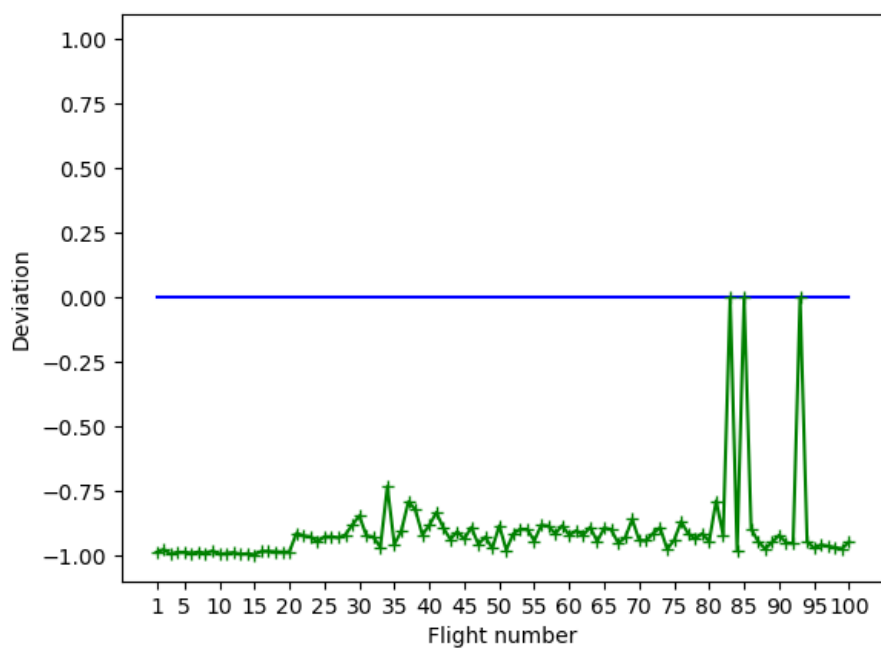


Figure A.14 LOF defence results, third version

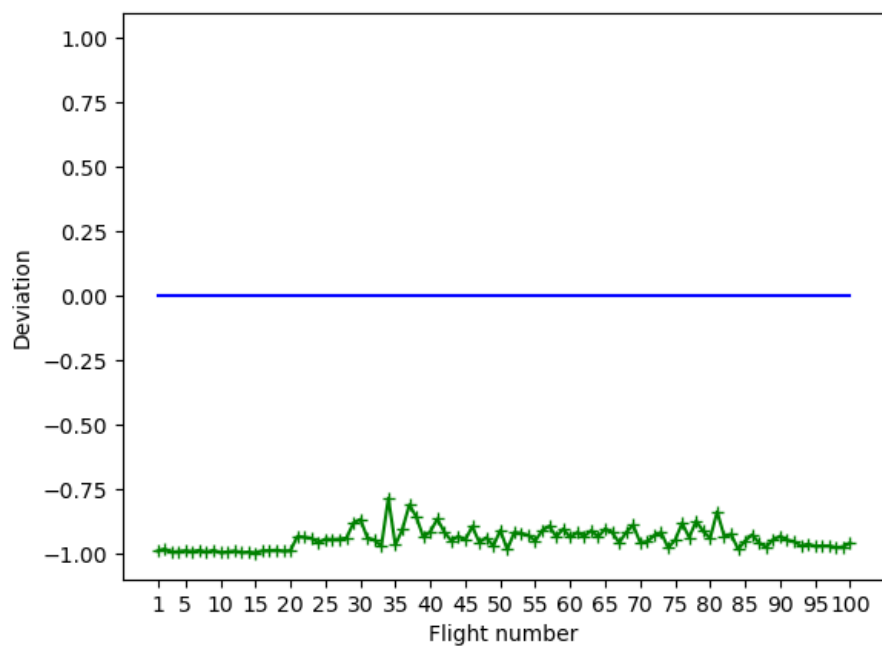


Figure A.15 LOF defence results, fourth version

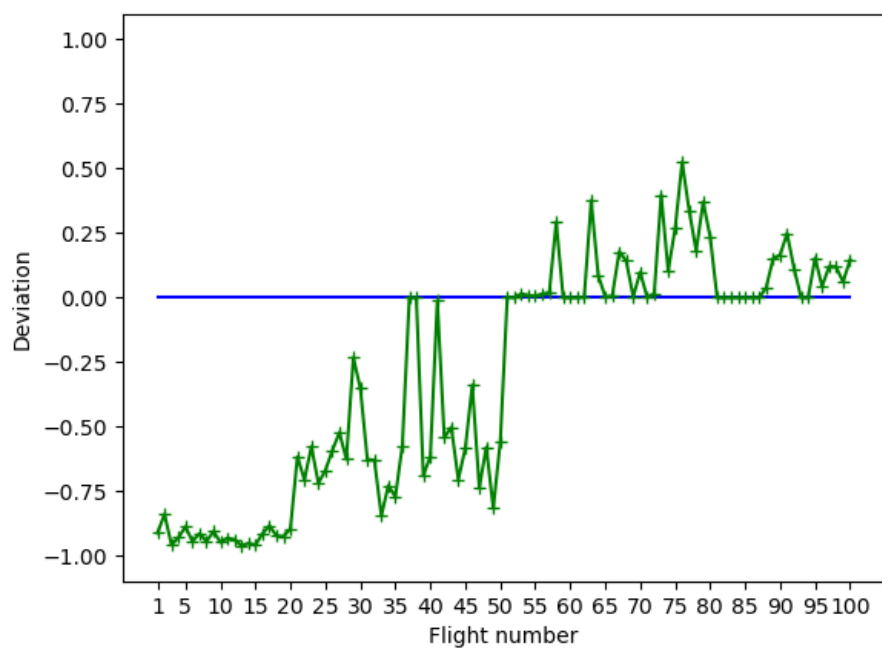


Figure A.16 LOF defence results, fifth version

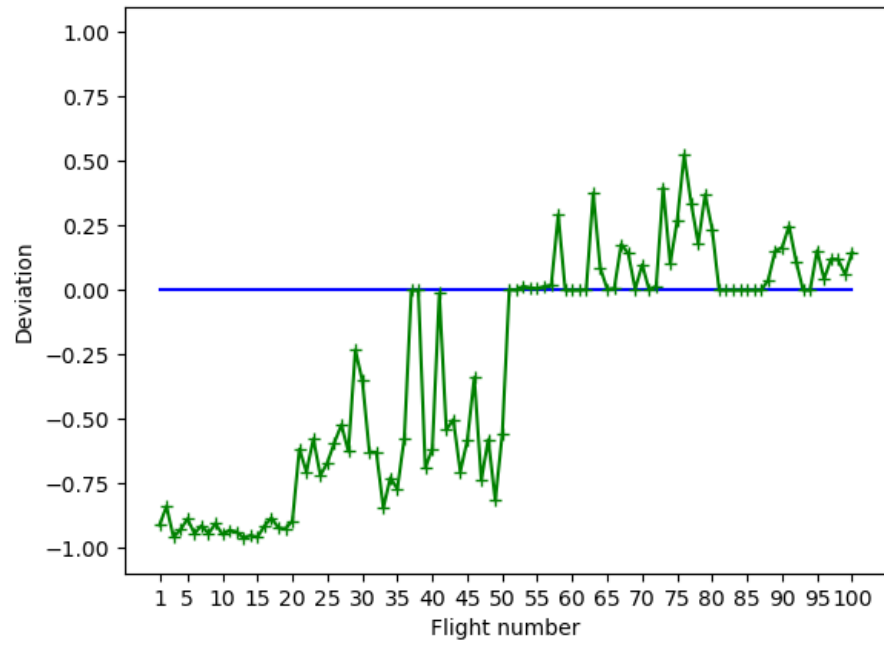


Figure A.17 LOF defence results, sixth version

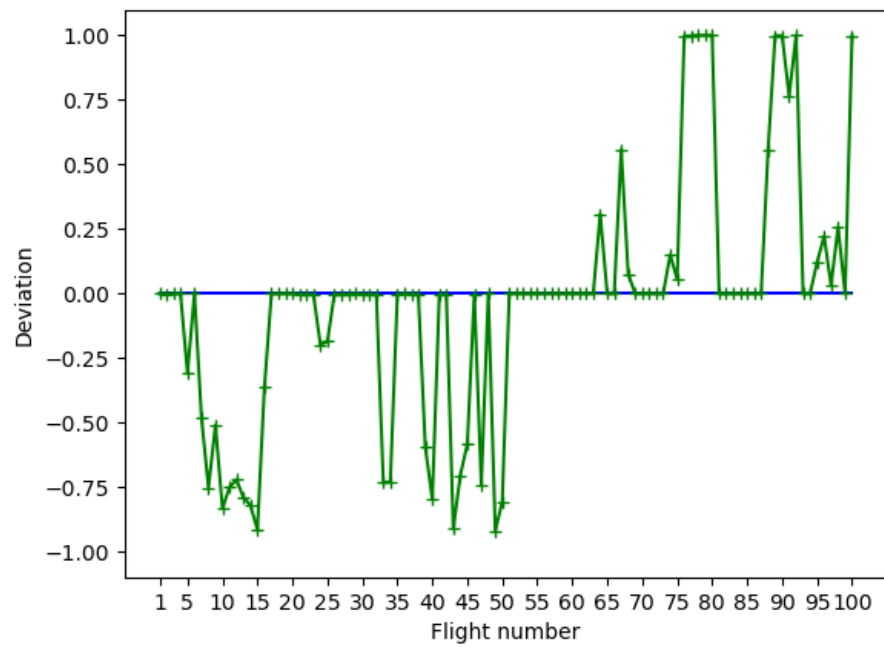


Figure A.18 LOF defence results, seventh version

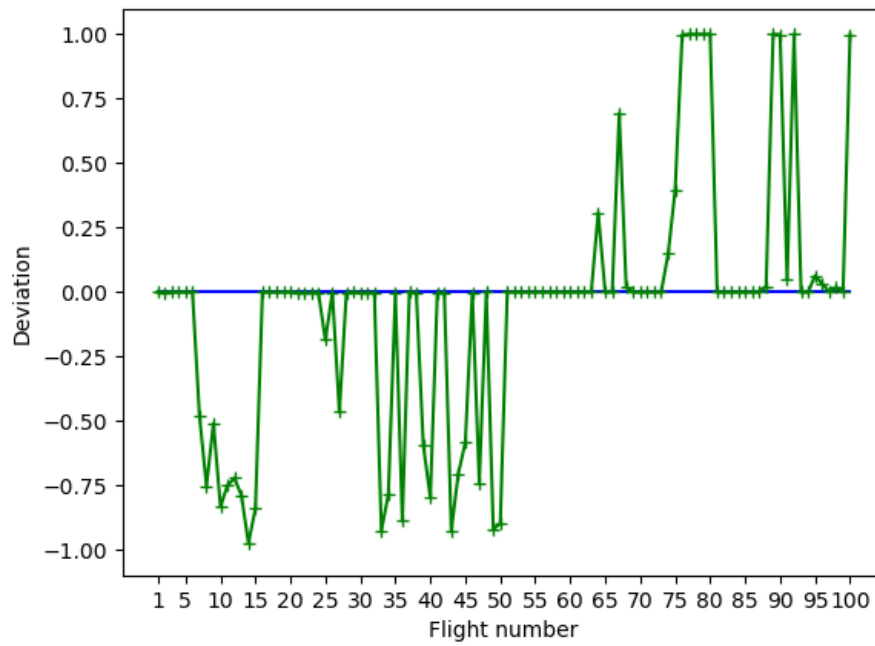


Figure A.19 LOF defence results, eighth version

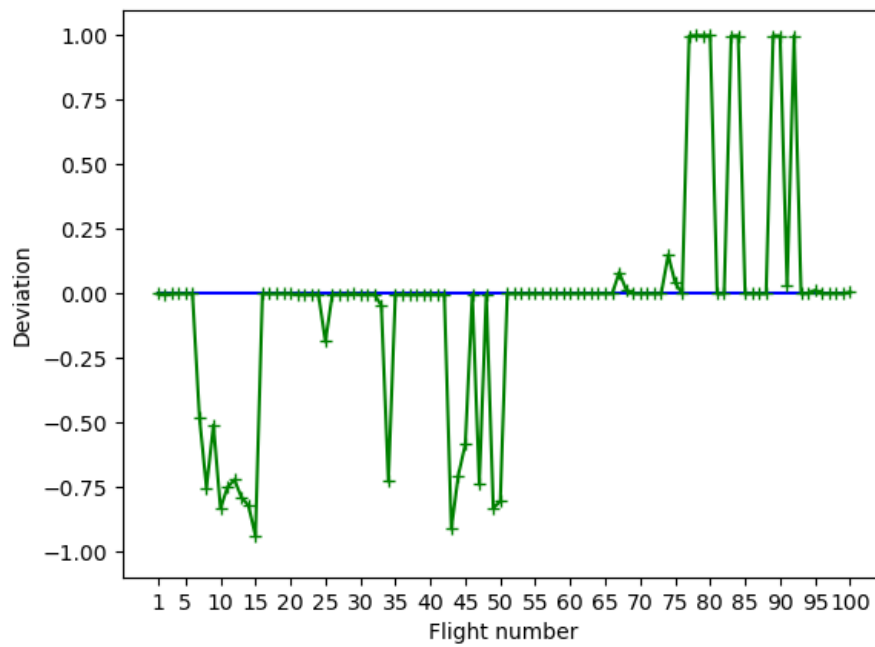


Figure A.20 LOF defence results, ninth version

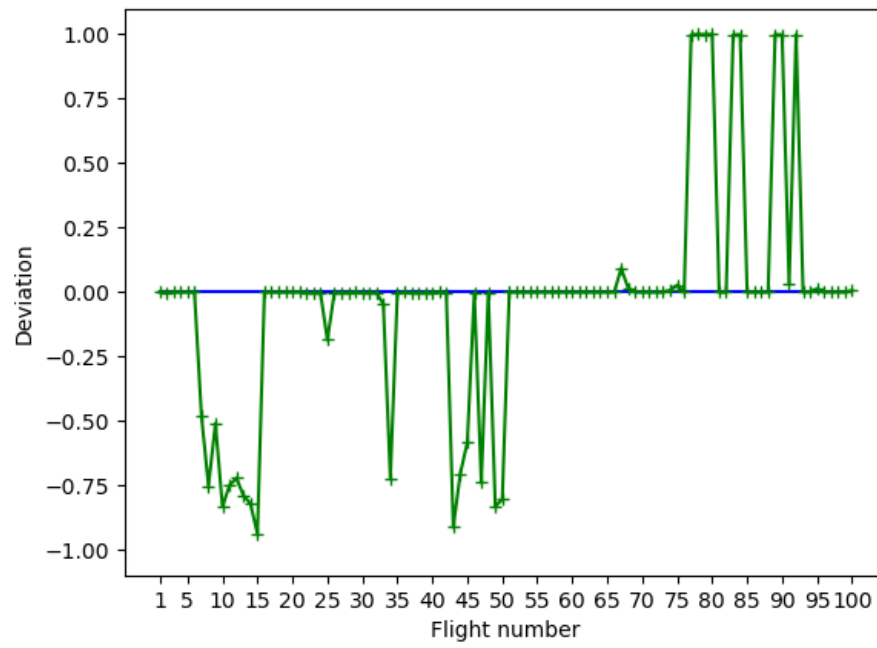


Figure A.21 LOF defence results, tenth version

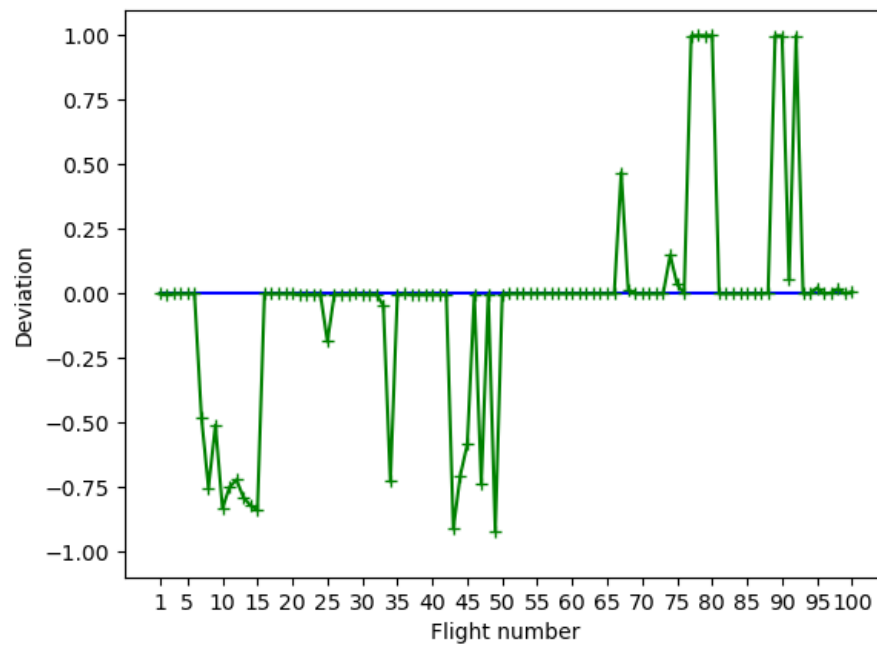


Figure A.22 LOF defence results, eleventh version

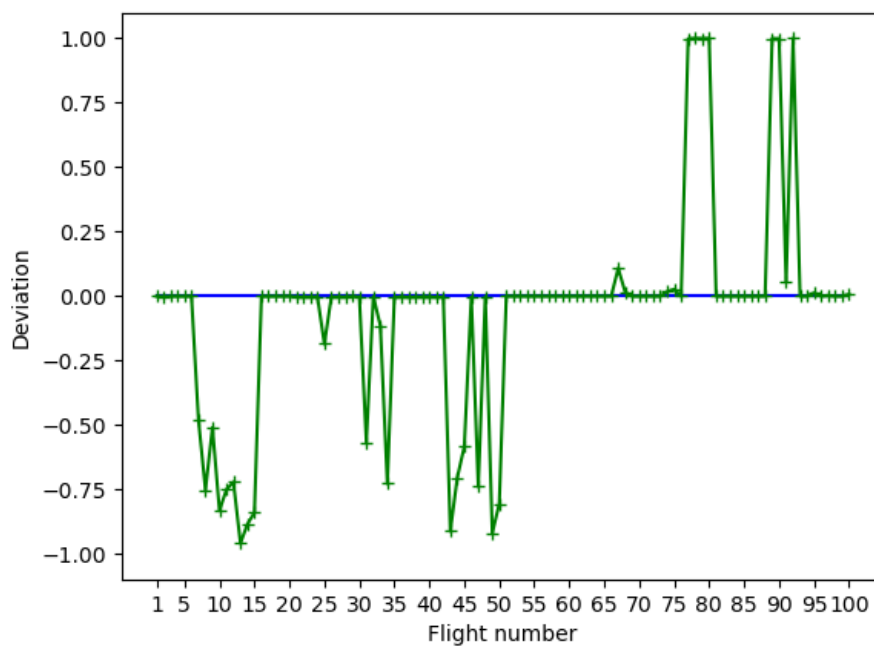


Figure A.23 LOF defence results, twelfth version

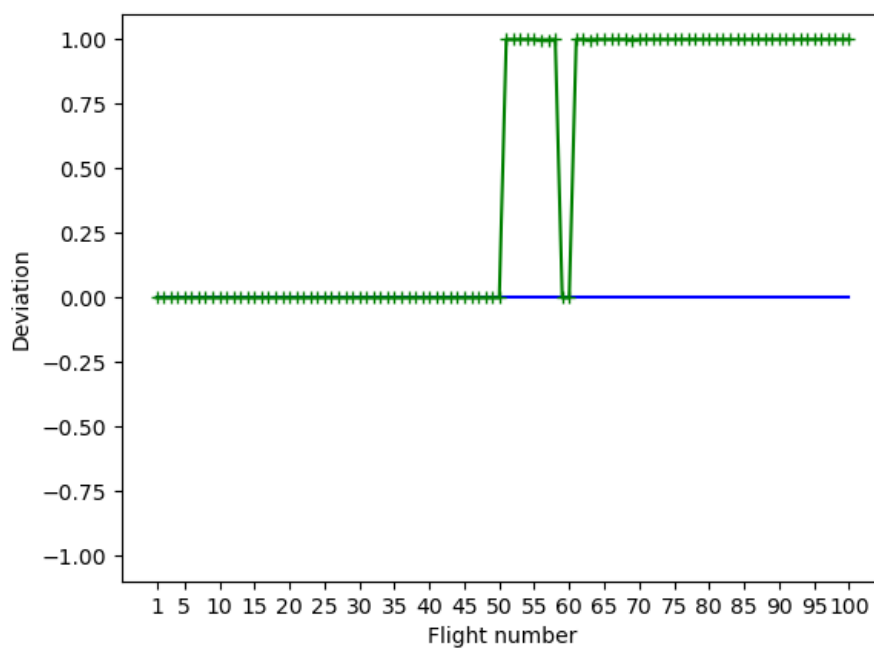


Figure A.24 OCSVM defence results, first version

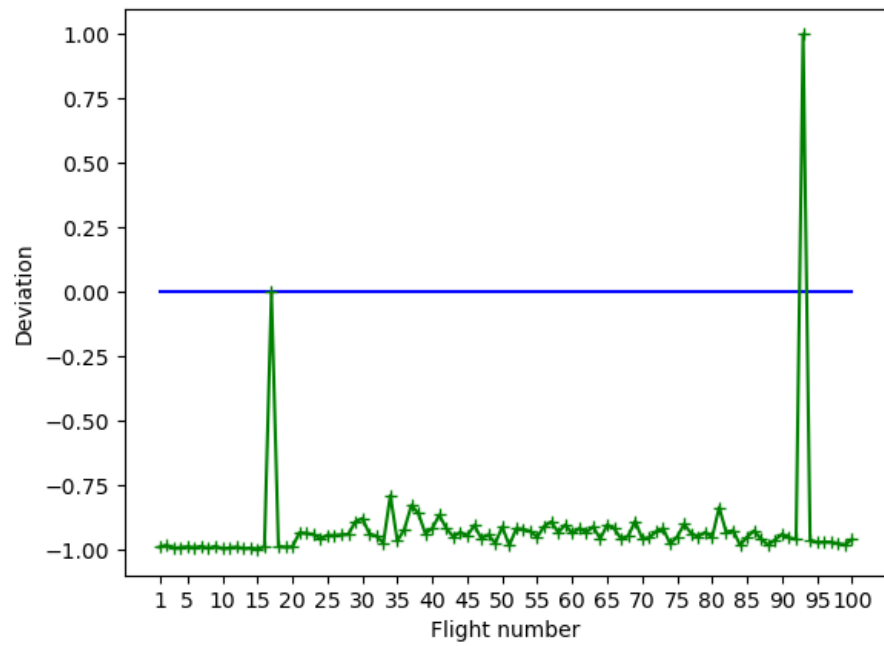


Figure A.25 OCSVM defence results, second version

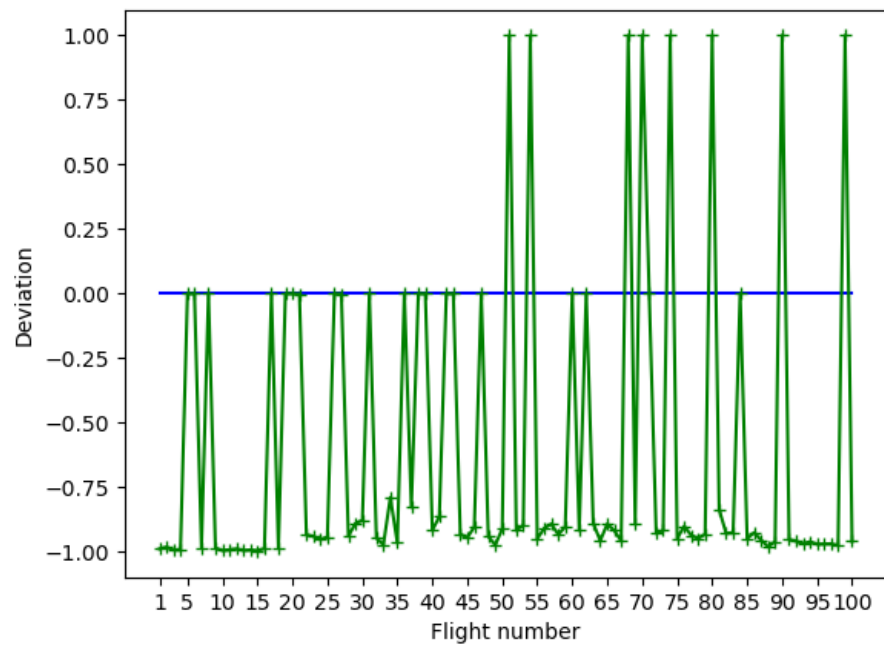


Figure A.26 OCSVM defence results, third version

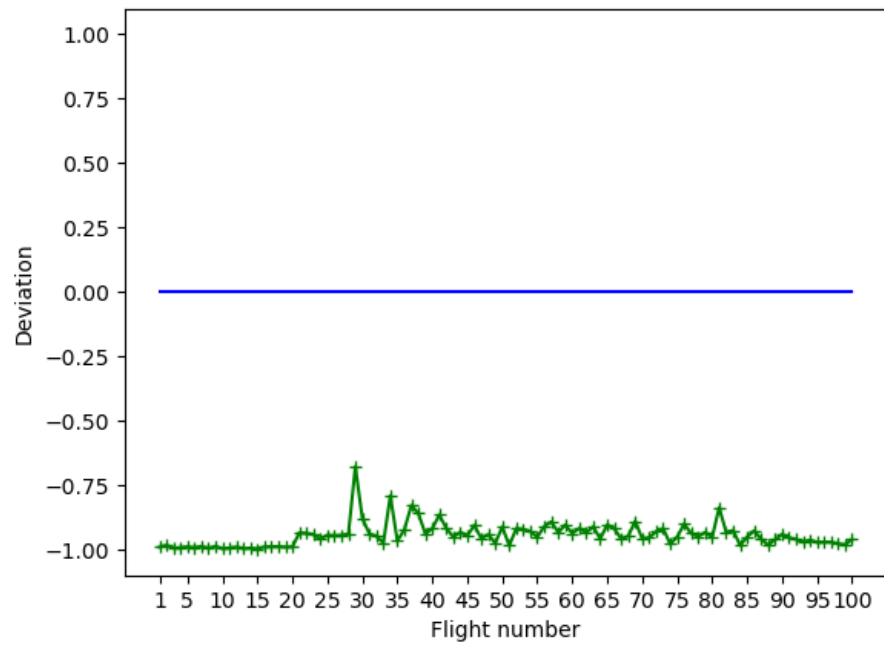


Figure A.27 OCSVM defence results, fourth version

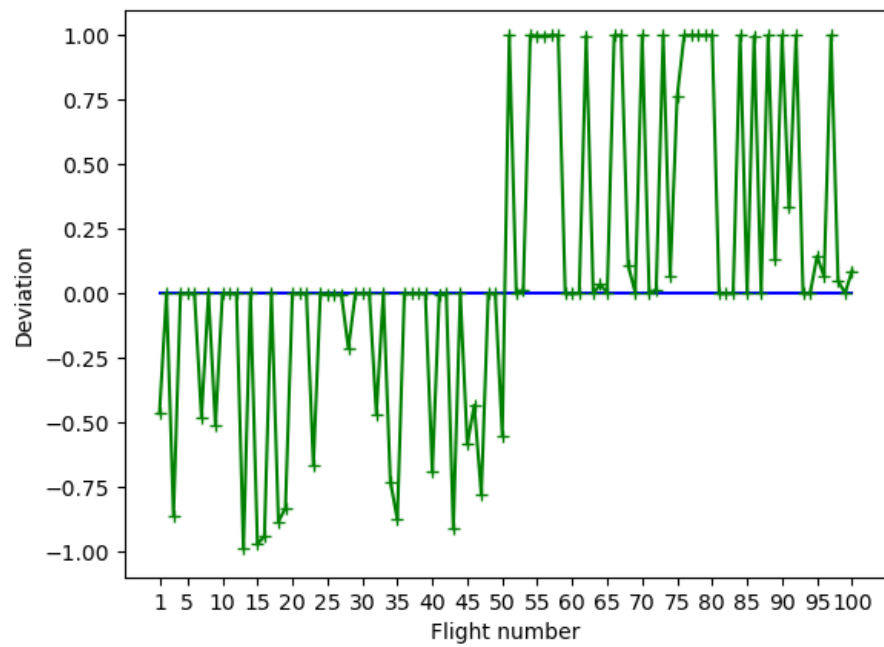


Figure A.28 OCSVM defence results, fifth version

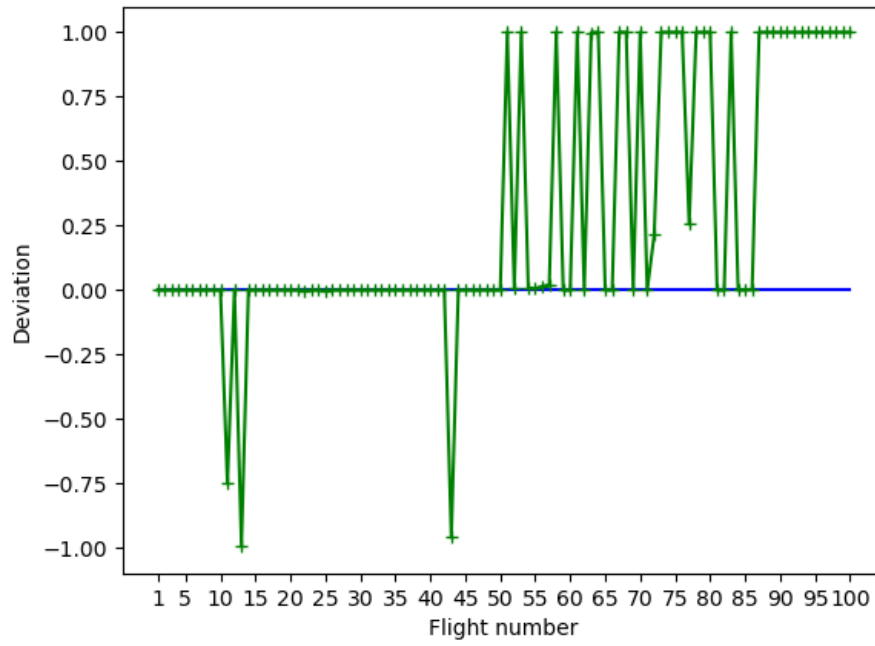


Figure A.29 OCSVM defence results, sixth version

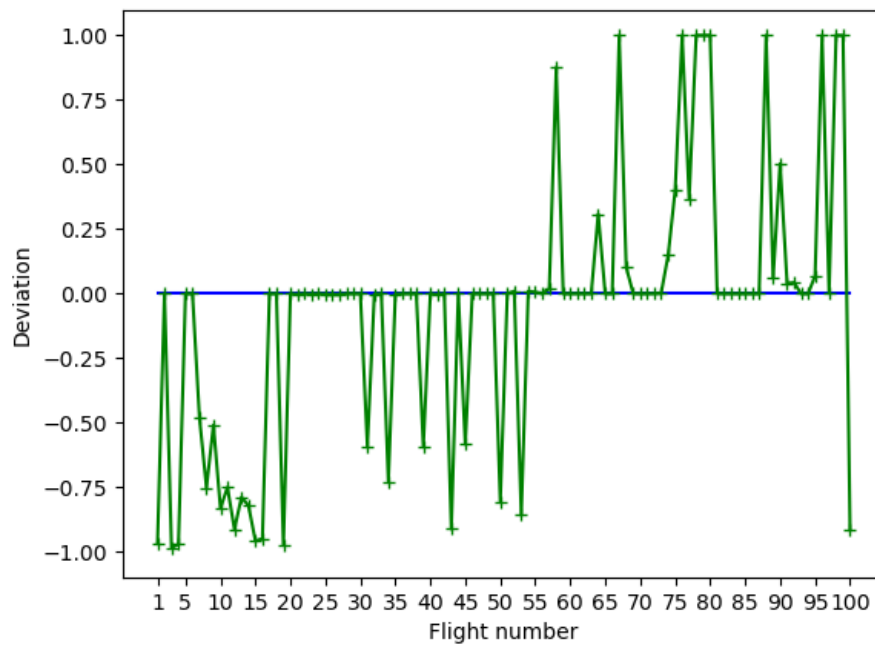


Figure A.30 OCSVM defence results, seventh version

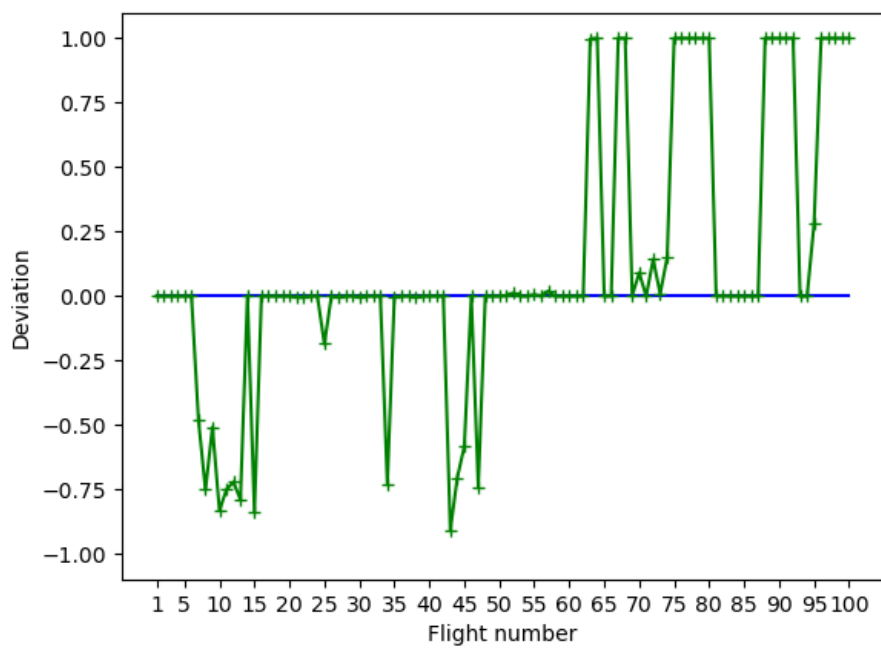


Figure A.31 OCSVM defence results, eighth version

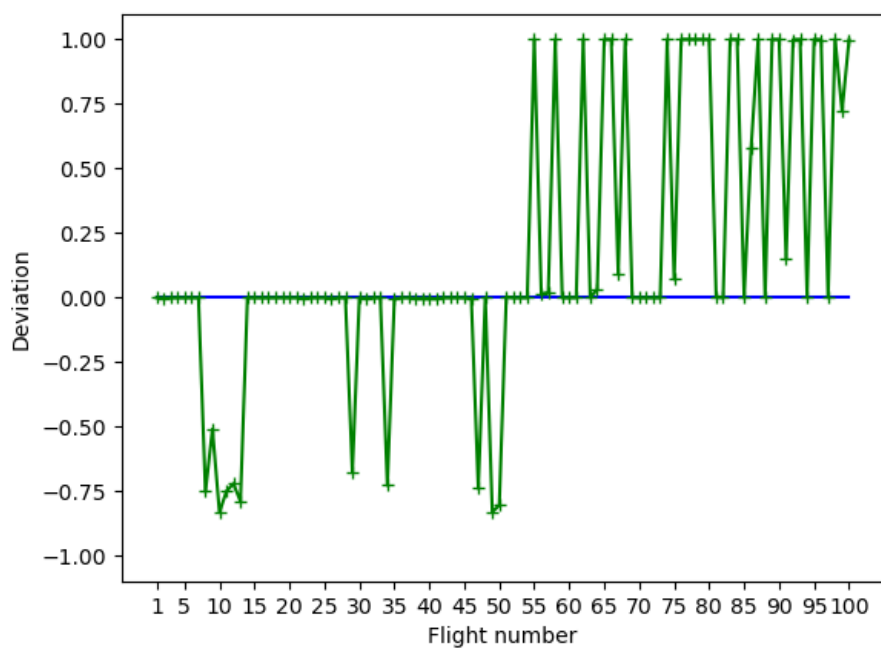


Figure A.32 OCSVM defence results, ninth version

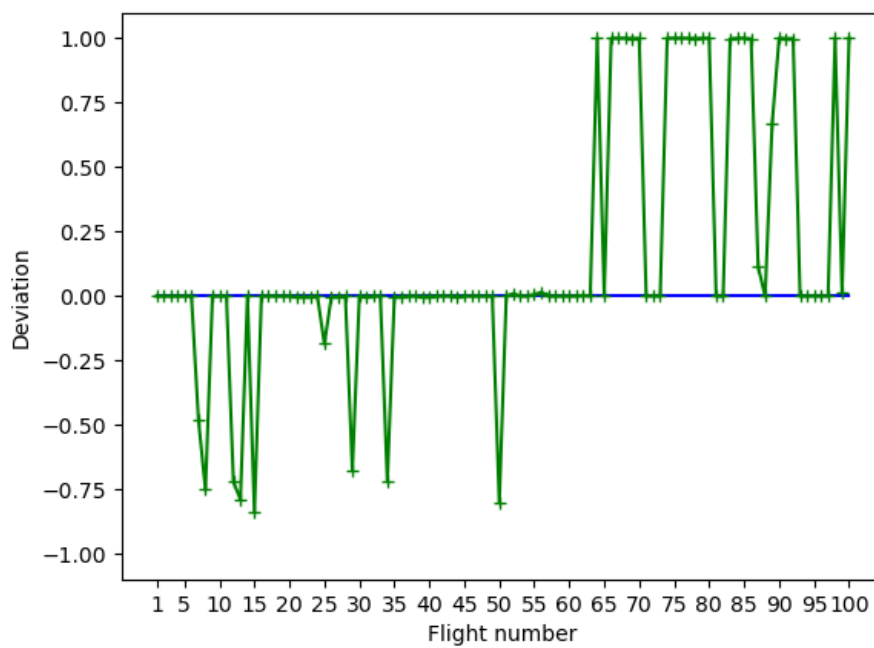


Figure A.33 OCSVM defence results, tenth version

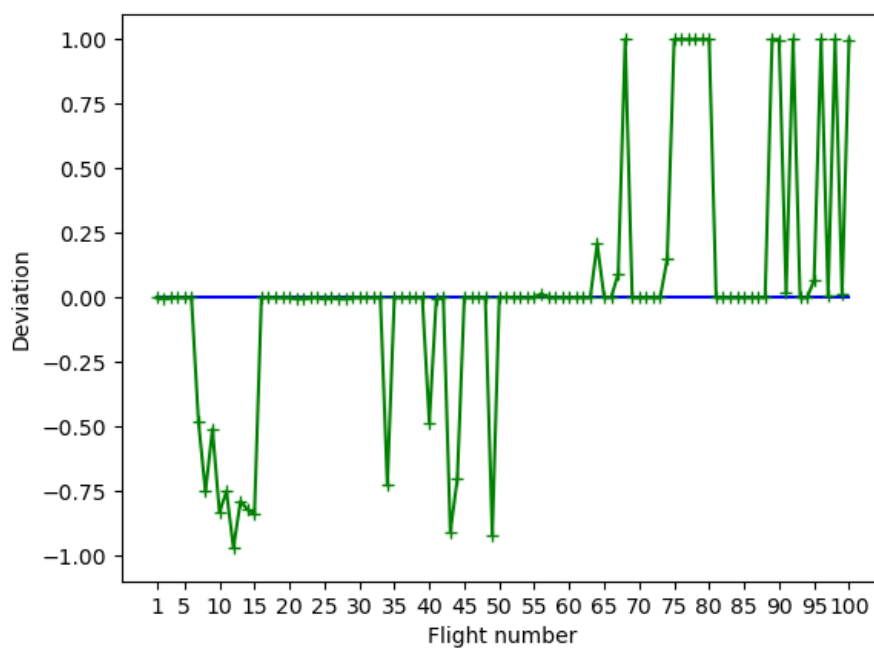


Figure A.34 OCSVM defence results, eleventh version

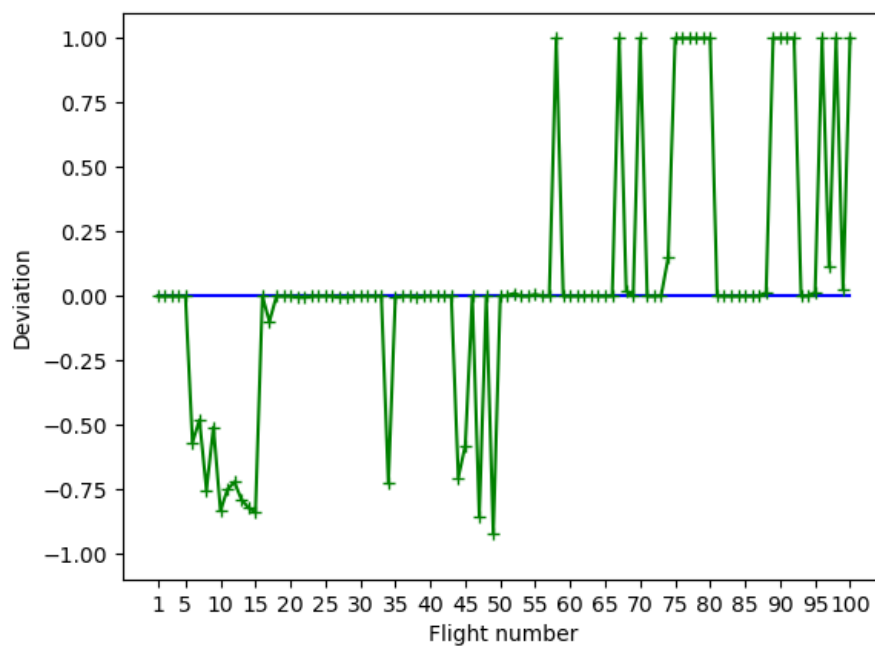


Figure A.35 OCSVM defence results, twelfth version

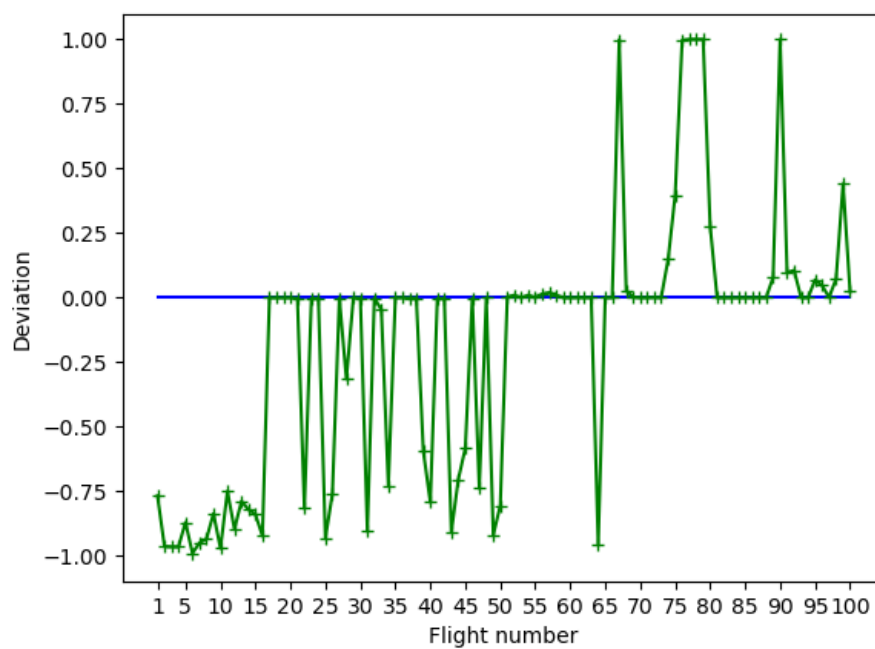


Figure A.36 OCSVM defence results, thirteenth version

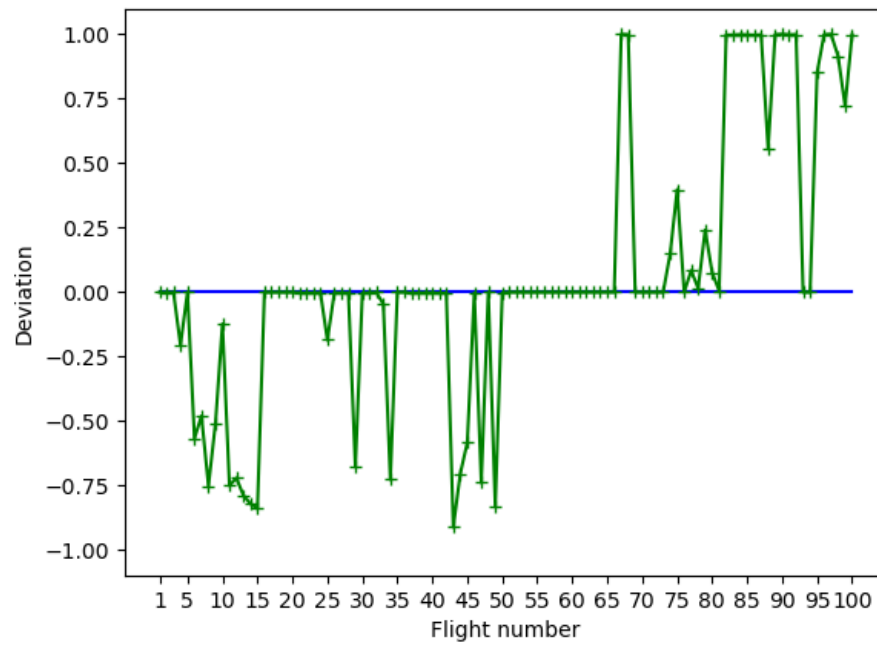


Figure A.37 Distance defence results, first version

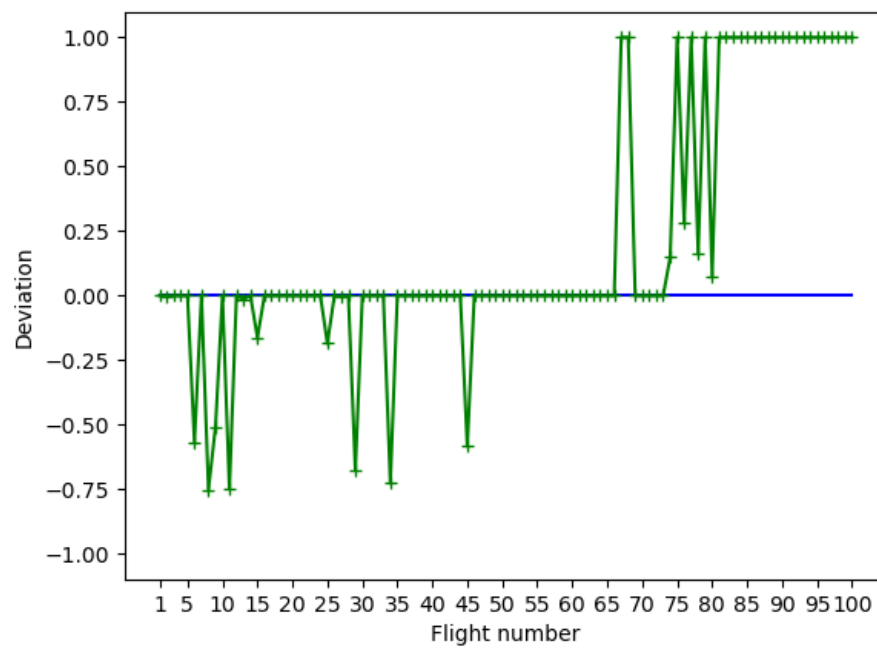


Figure A.38 Distance defence results, second version

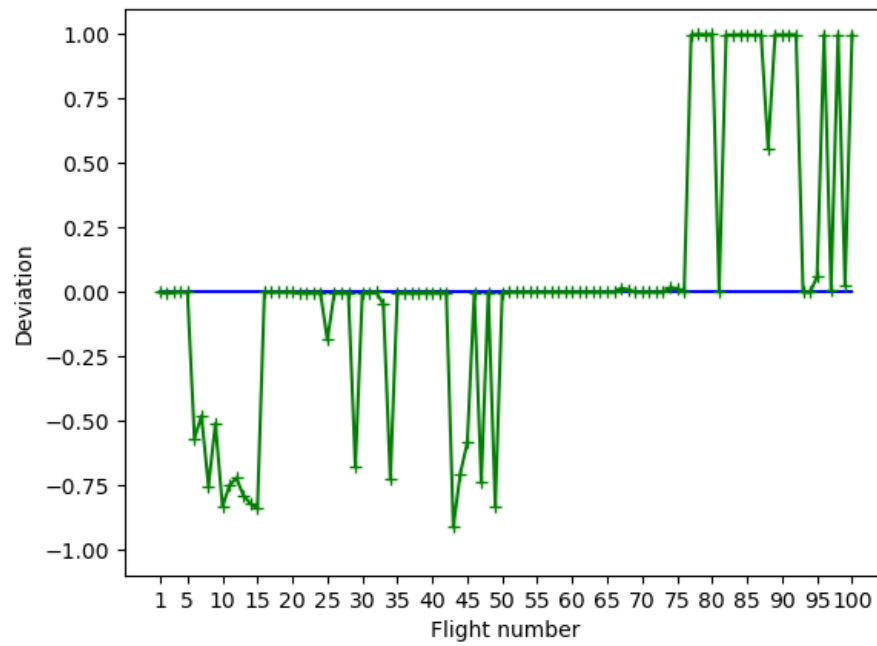


Figure A.39 Distance defence results, third version

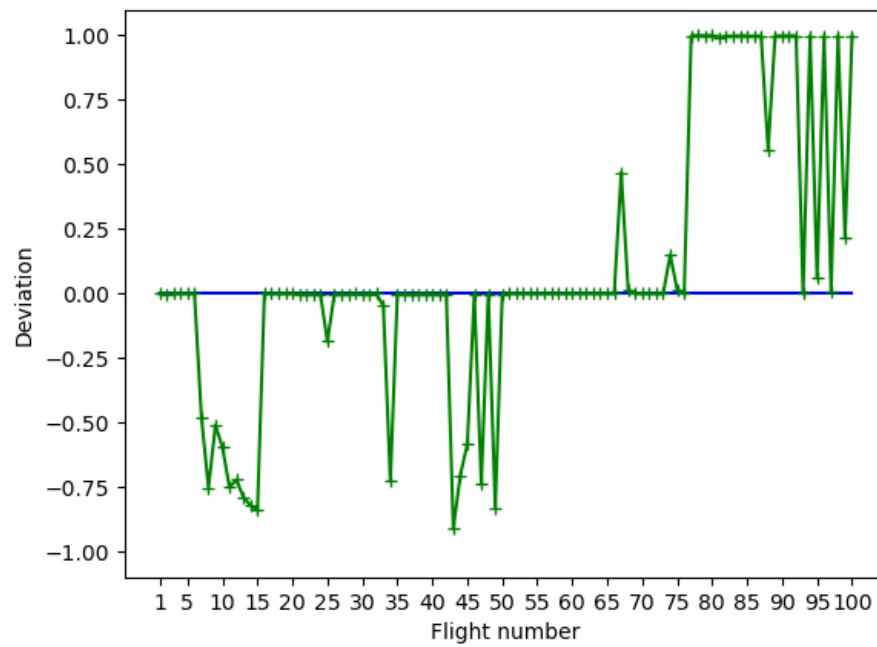


Figure A.40 Distance defence results, fourth version

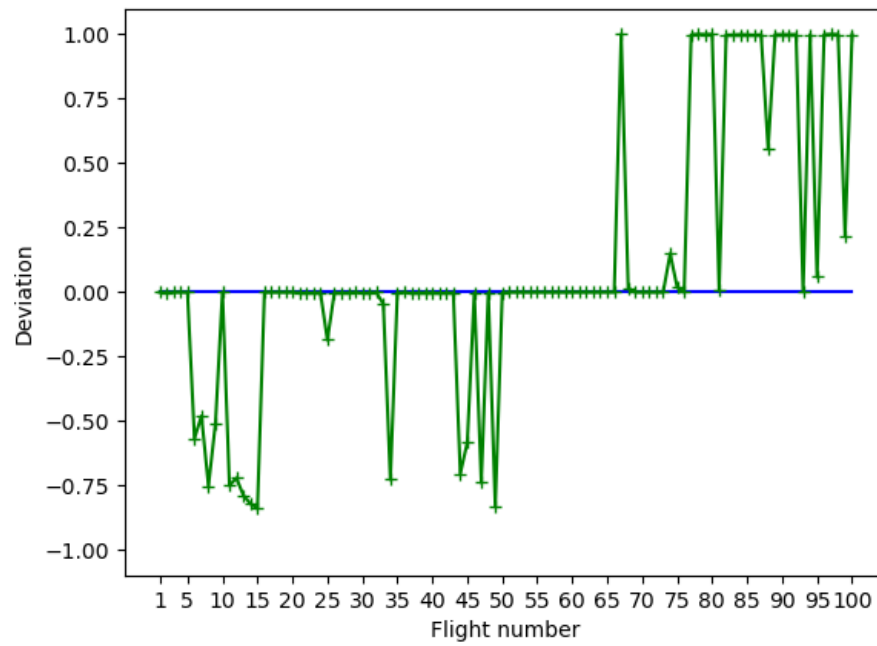


Figure A.41 Distance defence results, fifth version

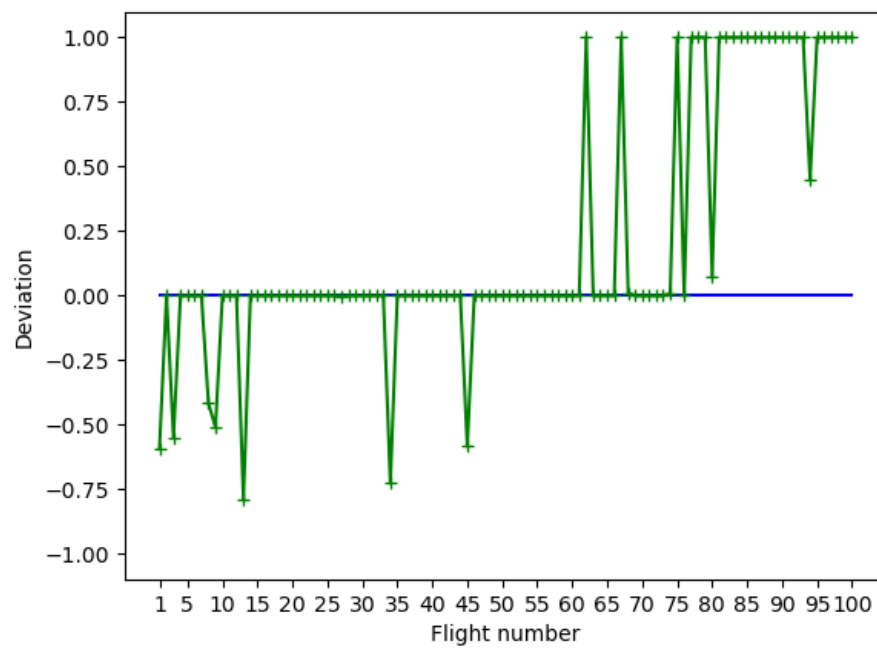


Figure A.42 Speed defence results, first version

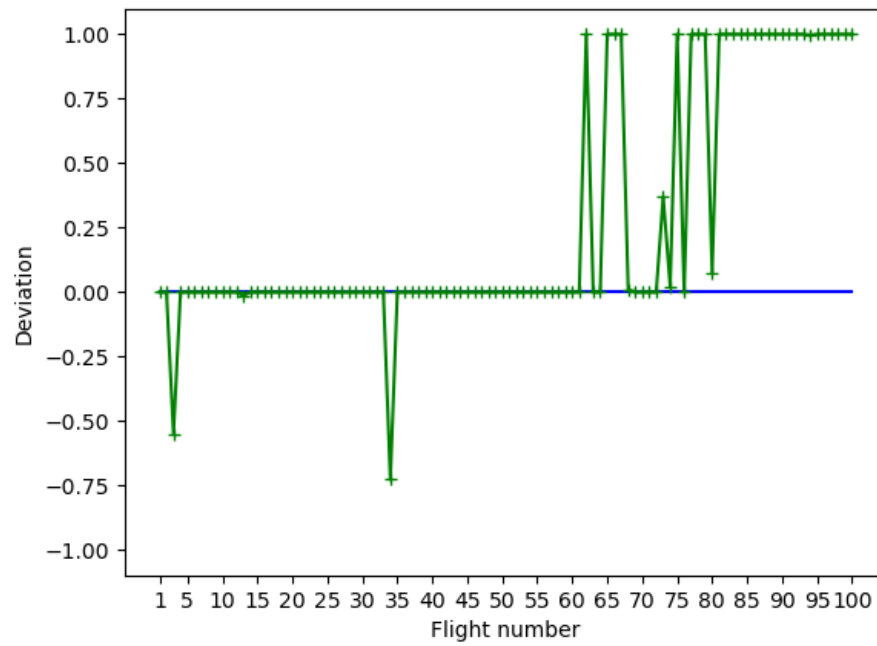


Figure A.43 Speed defence results, second version

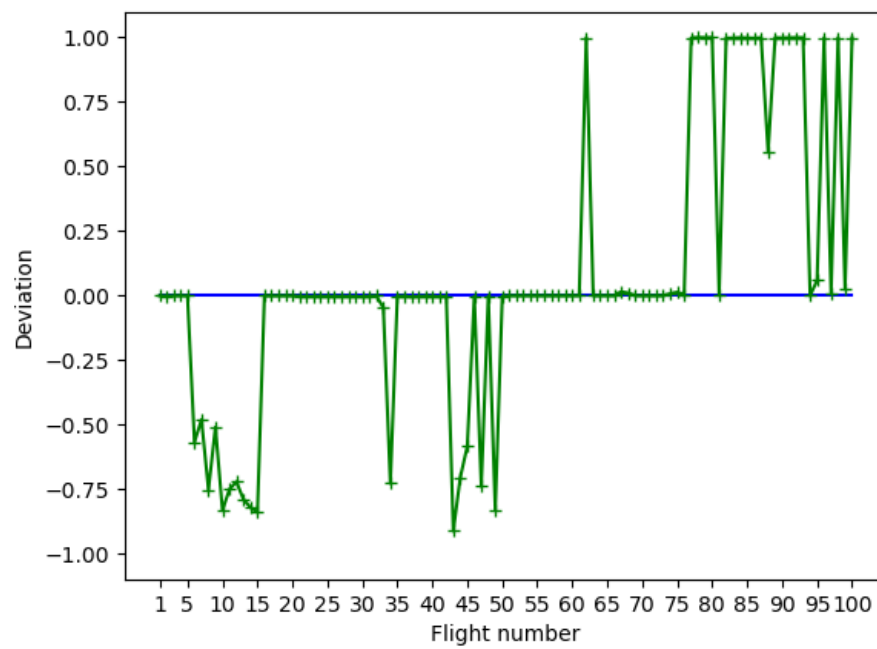


Figure A.44 Speed defence results, third version

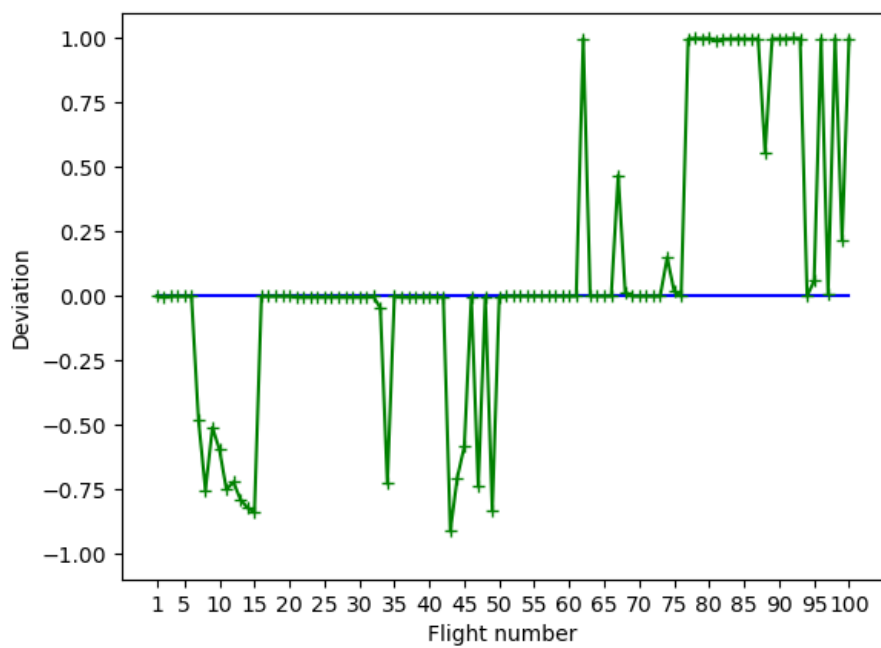


Figure A.45 Speed defence results, fourth version

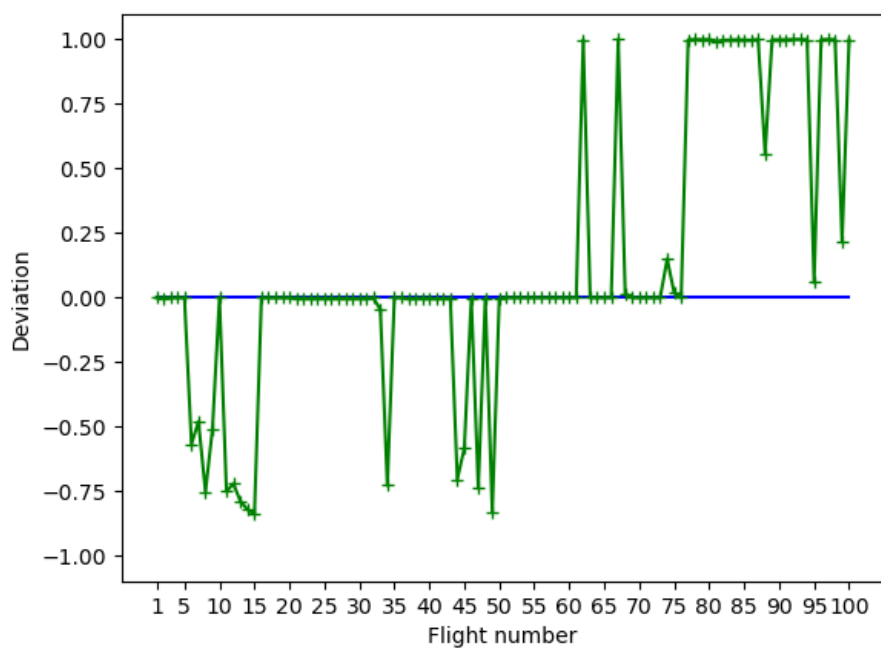


Figure A.46 Speed defence results, fifth version

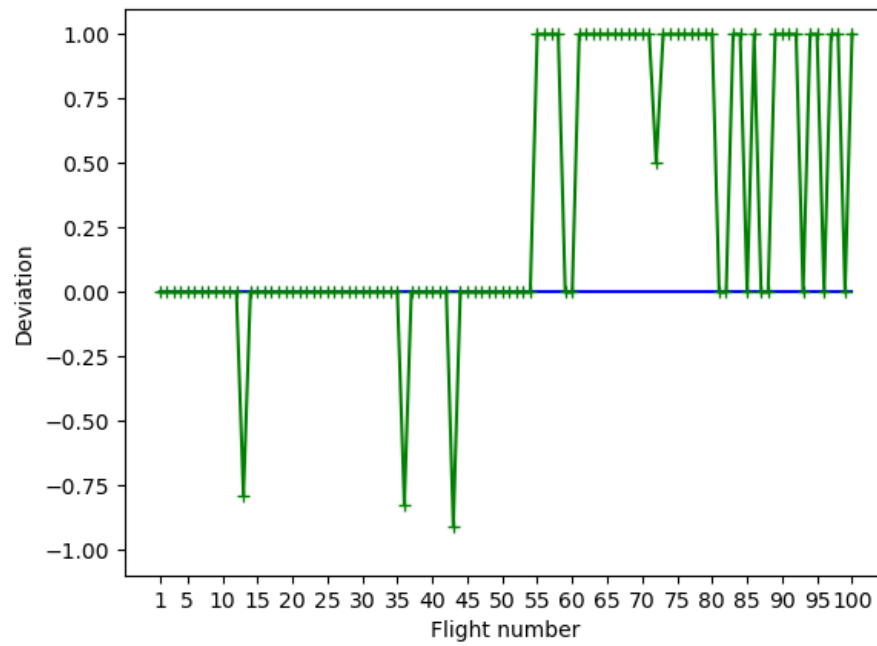


Figure A.47 Heading defence results, first version

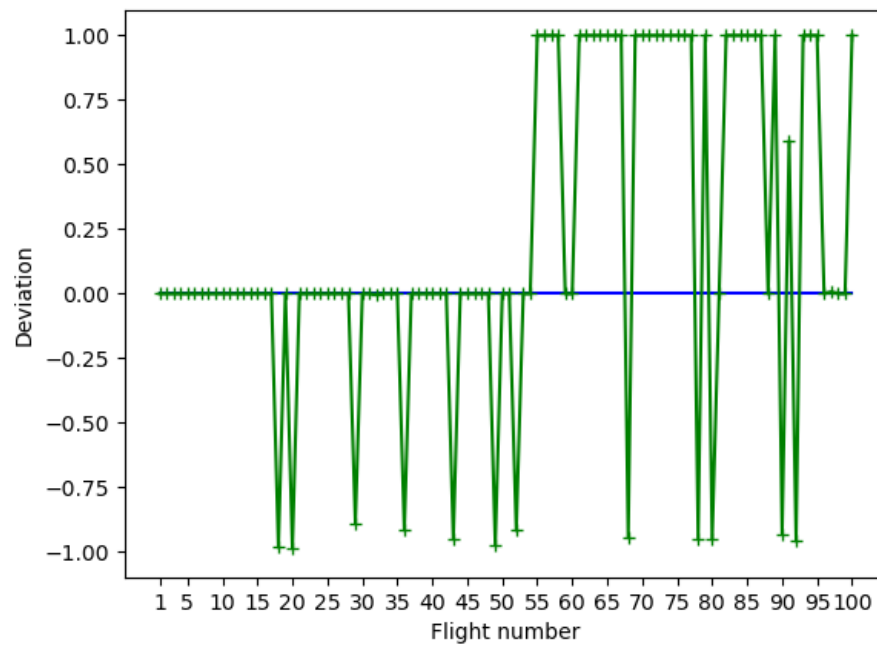


Figure A.48 Heading defence results, second version

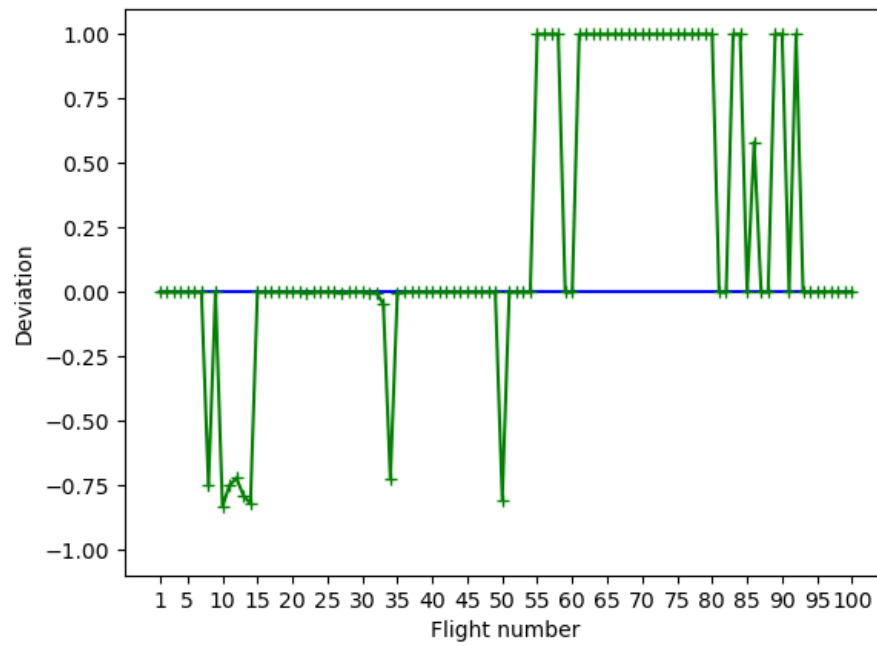


Figure A.49 Heading defence results, third version

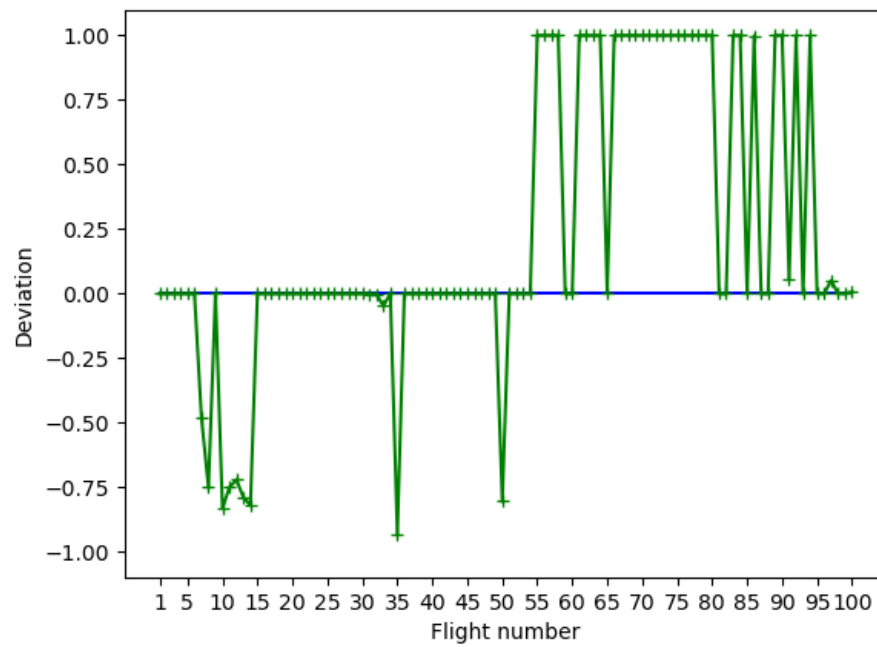


Figure A.50 Heading defence results, fourth version

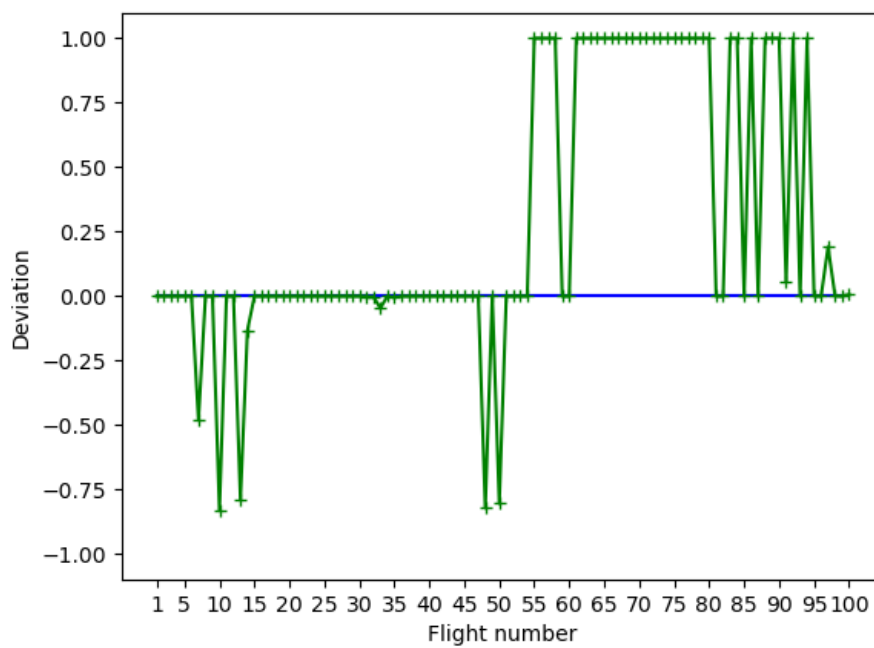


Figure A.51 Heading defence results, fifth version

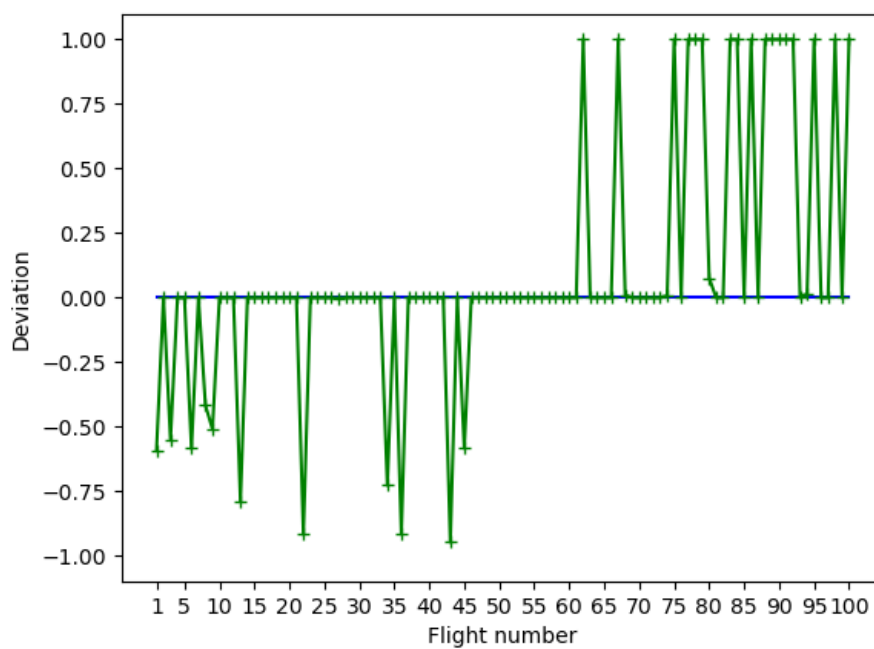


Figure A.52 Combination defence results, first version

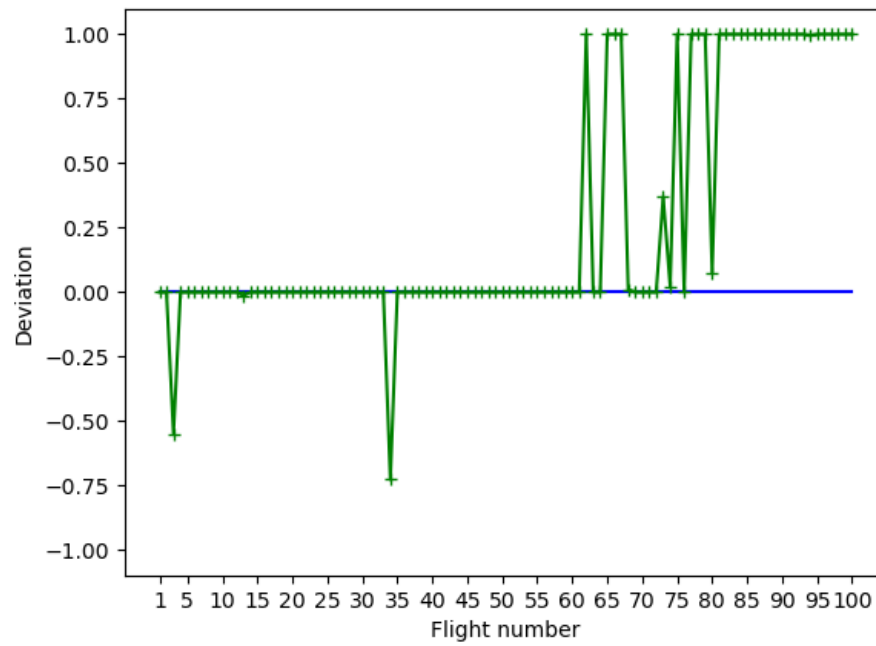


Figure A.53 Combination defence results, second version

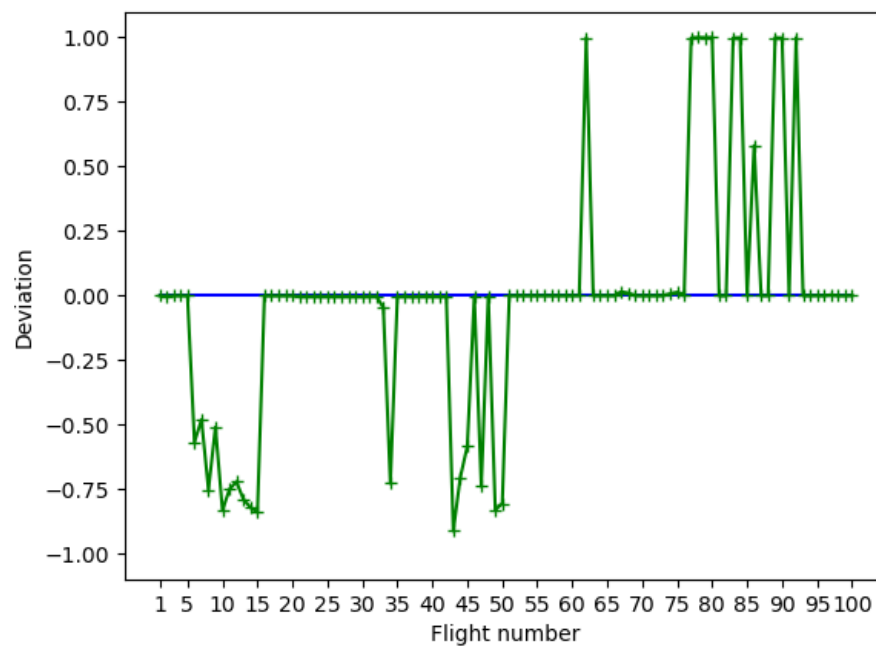


Figure A.54 Combination defence results, third version

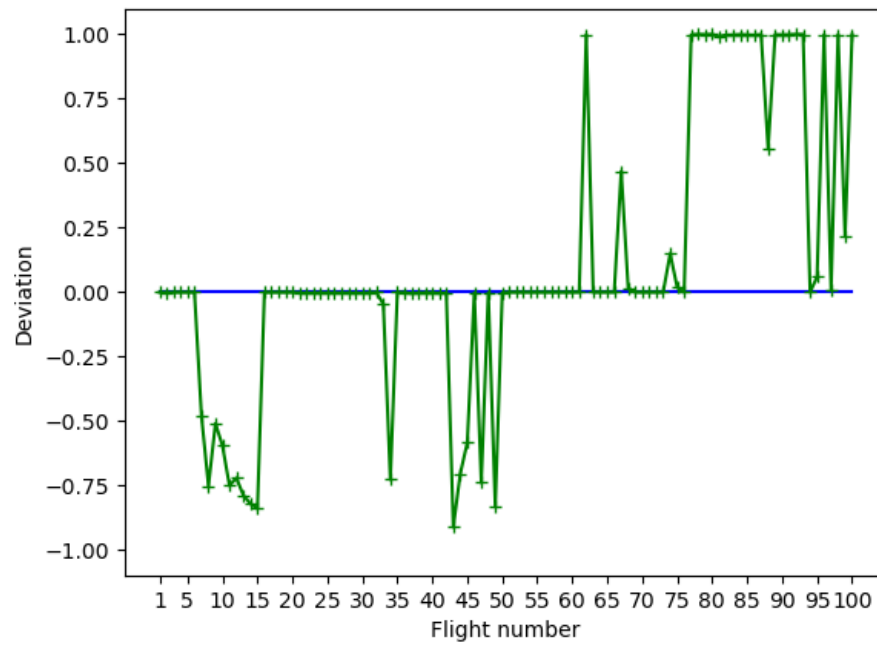


Figure A.55 Combination defence results, fourth version

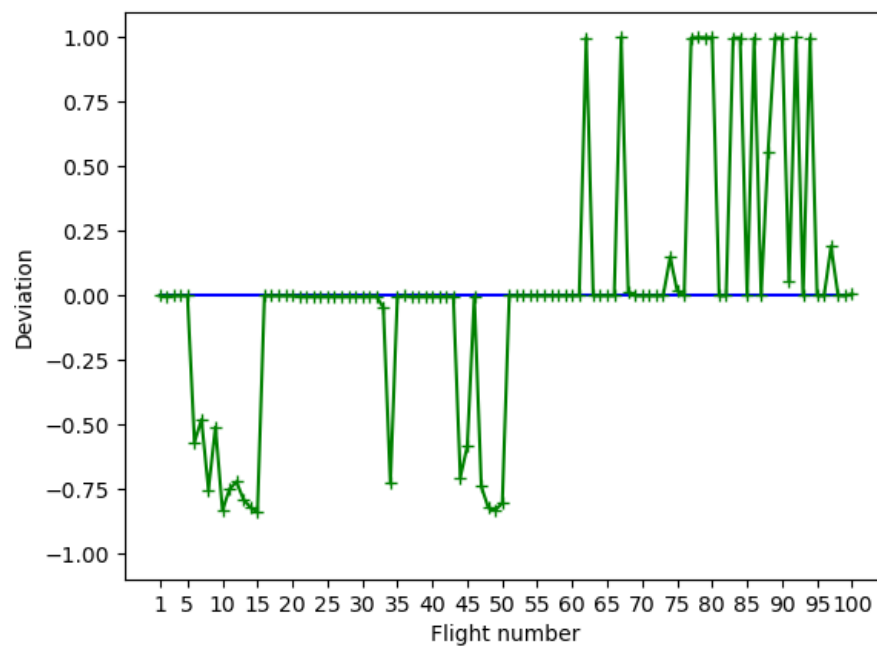


Figure A.56 Combination defence results, fifth version

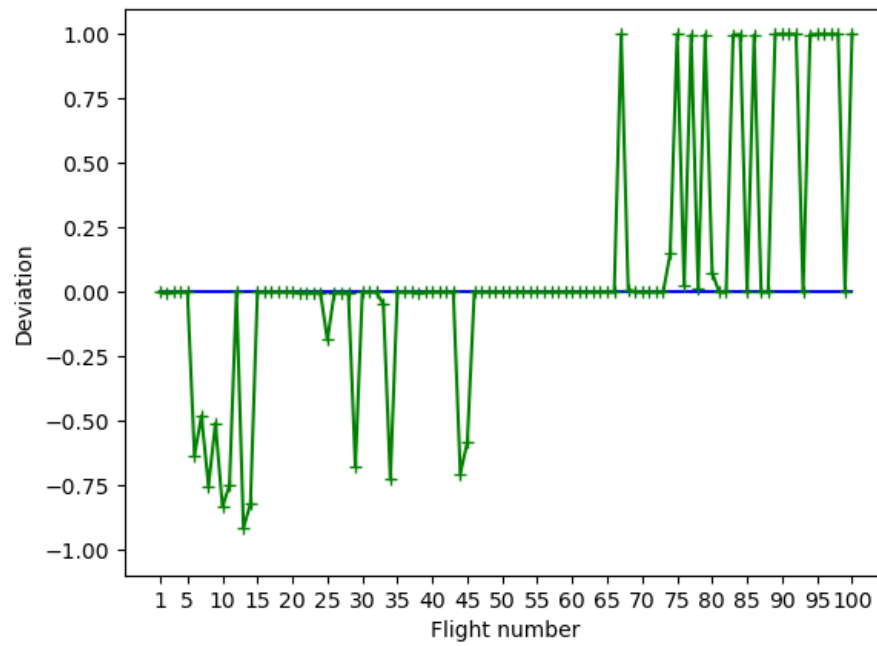


Figure A.57 Mean defence results, first version

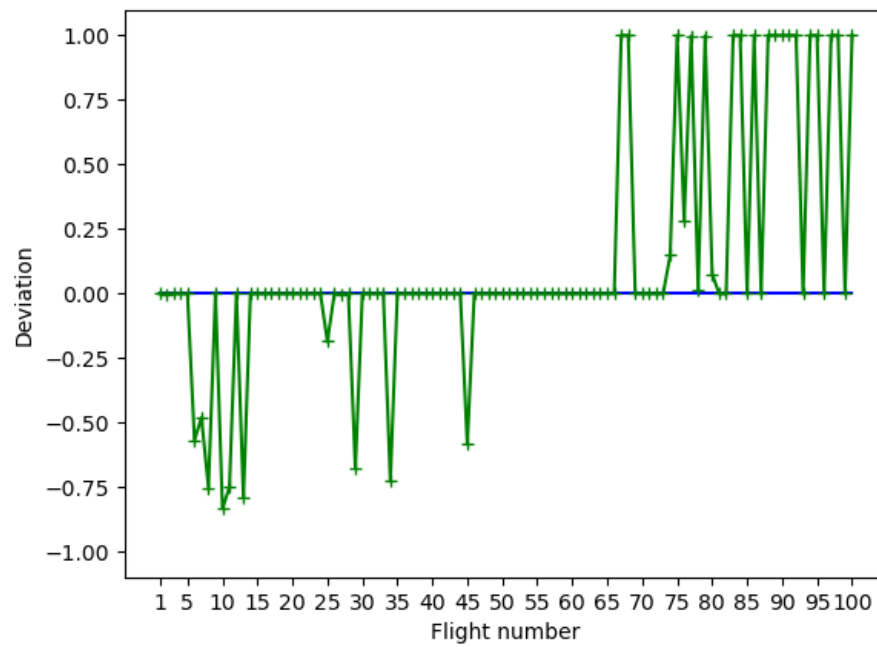


Figure A.58 Mean defence results, second version

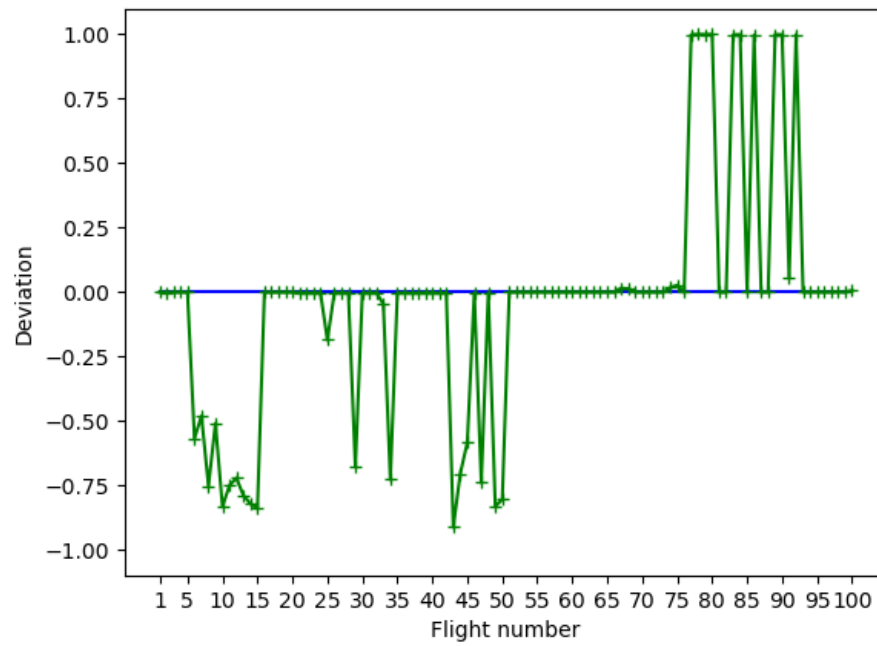


Figure A.59 Mean defence results, third version

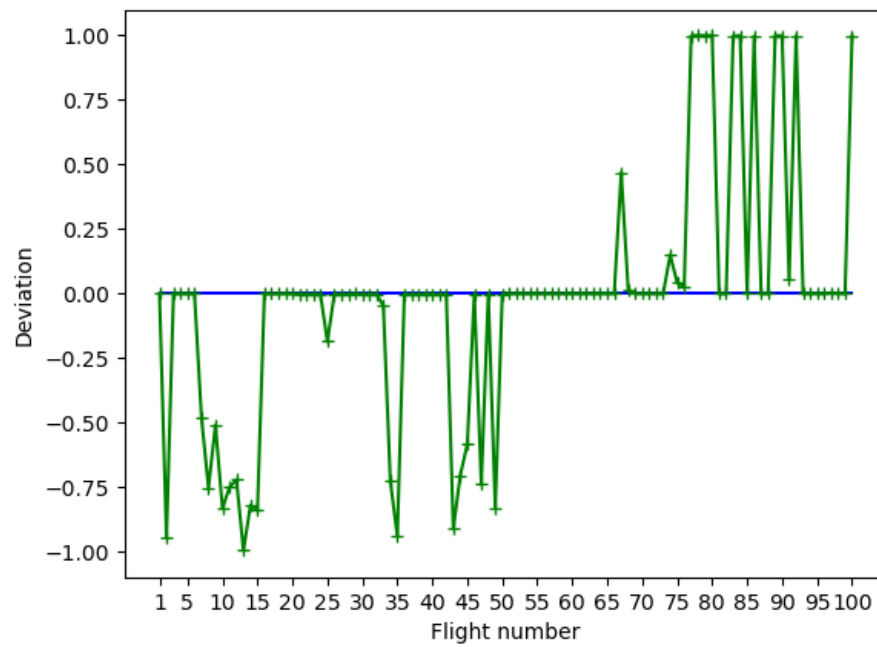


Figure A.60 Mean defence results, fourth version

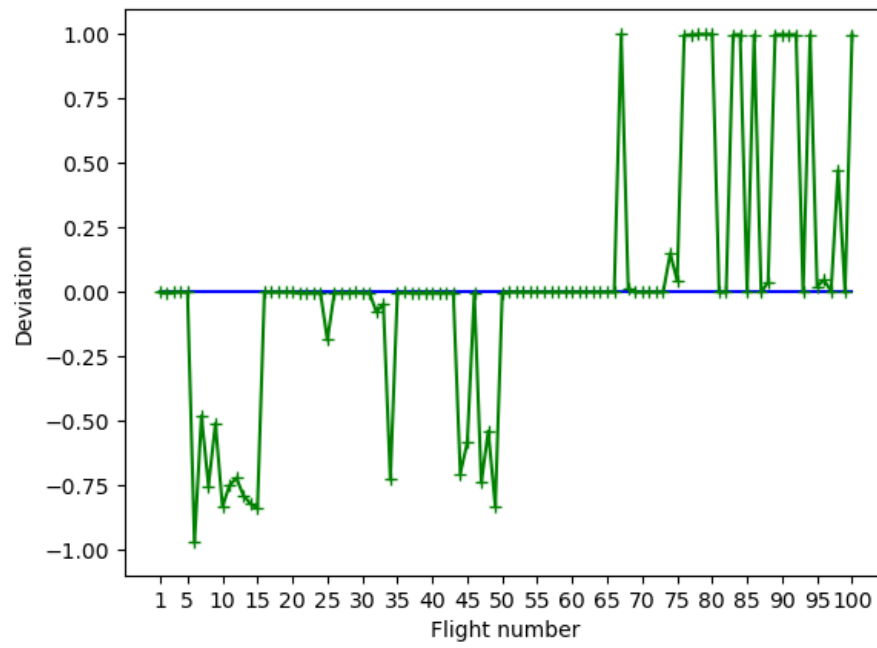


Figure A.61 Mean defence results, fifth version

APPENDIX B COMPLETE RESULTS TABLE

Table B.1 Complete Results

Name	Version	Score
Autoencoder	Second	0.8162065503688939
LOF	Eleventh	0.8087331786192506
LOF	Twelfth	0.7973556994270835
LOF	Tenth	0.7865818553033712
LOF	Ninth	0.78499231356428
Combination	Third	0.7692012922391414
Mean	Third	0.7656601866388012
Combination	Fifth	0.7610183974019012
OCSVM	Eleventh	0.7592631338347487
Linear Regression	First	0.7583316200872077
Combination	First	0.7536067350433643
EKF	Second	0.7514238808137792
Mean	Second	0.7513858103460029
Autoencoder	First	0.7502604887413689
Mean	Fourth	0.7490628987174253
Mean	Fifth	0.7442994150463014
Linear Regression	Second	0.7442071204573331
EKF	Sixth	0.7426796606823655
Mean	First	0.7410714853666187
EKF	Fifth	0.7394572667561295
Combination	Sixth	0.7365707714308404
LOF	Eighth	0.7316120031426654
LOF	Seventh	0.7273295101359879
OCSVM	Twelfth	0.7268656694443035
Mean	Sixth	0.7229063434092101
OCSVM	Tenth	0.7119069862164296
OCSVM	Seventh	0.7039282071517807
Speed	First	0.7029087676466456

Table B.2 Complete Results (cont'd)

Combination	Second	0.7023816186935744
Speed	Second	0.7023816186935744
Distance	Third	0.6980375711266434
OCSVM	Eighth	0.6975545614331868
Distance	Second	0.6940245278961167
Distance	First	0.6900682268206899
Distance	Sixth	0.6891039668980403
EKF	Third	0.6880657911513459
Speed	Third	0.6867446969253024
Distance	Fourth	0.6851604861816303
Combination	Fourth	0.6769500629878419
Speed	Fourth	0.6769500629878419
Distance	Fifth	0.6733709293476903
OCSVM	Ninth	0.6715052590213462
Speed	Fifth	0.6651974083976918
OCSVM	Thirteenth	0.6640423976903103
Speed	Sixth	0.6611322227044354
OCSVM	Sixth	0.6575864641494655
Heading	Fifth	0.6458865480504737
Heading	Third	0.6416551837252642
Heading	Sixth	0.6383443629683875
Heading	Fourth	0.6295777481230157
OCSVM	Fifth	0.6236015186534575
Heading	First	0.6197883407576237
Linear Regression	Third	0.606216763948459
LOF	Fifth	0.6036618939286186
LOF	Sixth	0.600528633483409
EKF	First	0.5984912541439668
LSTM	First	0.582892955734885
Heading	Second	0.5605923746740571

Table B.3 Complete Results (cont'd)

Linear Acceleration	First	0.5314006509821438
OCSVM	First	0.5200689049659085
EKF	Fourth	0.5200689049659085
OCSVM	Third	0.243560436880502
LOF	Third	0.0979408443639658
OCSVM	Second	0.0658885176591396
LOF	First	0.0616167331755774
LOF	Second	0.0604884421900144
LOF	Fourth	0.05971713600455
OCSVM	Fourth	0.0584009407297604

APPENDIX C FLIGHT INFO

Table C.1 Flight Coordinates

Flight Number	Departure Location	Departure Latitude	Departure Longitude	Arrival Latitude	Arrival Longitude
1	McMillan	35.718719	-120.769818	35.0013241	-120.5234211
2	Marcopter	44.806968	11.610389	45	11
3	UChicago	41.7886079	-87.59871329999999	42	-86
4	Karuizawa	36.323203	138.618215	36	139
5	Pyramid	29.9764	31.1339	29.5	31
6	Rotherham	53.275131	-1.19404	54	-2
7	Skyrocket	33.982069	-118.426372	34	-117.7
8	SKT	44.850687	-0.684029	44	-1
9	KawaguchiLake	35.4712023	138.7450261	35.2	138.5
10	OSPN	30.53202	-97.62892	30.95	-98.85
11	Unalga	51.578	-179.0483	52	-178
12	GrupoSicoss	19.316518	-99.220255	19.5	-100
13	Snarbyeidet	69.775839	19.548929	69	20
14	Dalby	-27.274439	151.290064	-27.9902	151.97232
15	LeeField	32.42553	-84.79109	32.9872	-85.7875
16	KalaupapaCliffs	21.16564233	-156.88680160	21.35432	-156.163154
17	LakeGeorgeLookout	-35.10122092	149.37517574	-34.99876	148.5743621
18	RF_AirStadium	36.893328	-2.720371	37.5	-3

Table C.2 Flight Coordinates (cont'd)

19	SFO_Bay	37.62561973	-122.33235387	37.984312	-122.8798213
20	TomCarpark	26.048950	-80.254219	26.4012423	-80.0023176
21	PolyMTL	45.504914187778965	-73.61320455988907	45.6001243	-73.4767769
22	Biosphere	45.5140261225577	-73.53144080221767	45.4000134	-73.6500453
23	PeaceTower	45.42530712716617	-75.69995070222058	45.5543251	-75.8120134
24	WhiteHouse	38.89779311297567	-77.03656200574972	39.102343	-77.0048534
25	Eiffel	48.858532430128626	2.294492026729748	49.001002	2.1992031
26	Buckingham	51.5015242671342	-0.14186854434146765	51.6345321	0
27	Holyrood	55.952291066645074	-3.1755453427632103	56.10023	-3.0875432
28	UN	40.749379238082575	-73.96747811875441	40.8381324	-74.12341
29	ForbiddenCity	39.917144803426325	116.39072025218793	39.9992314	116.4112341
30	Azuchi	35.15602873983646	136.13949946448804	35.10012	136.201031
31	Reichstag	52.51876709179007	13.376230115124983	52.6	13.5
32	Warsaw	52.2439987801896	21.016725417928683	52.3	21.1
33	Coliseum	41.890361928635215	12.492230897669138	41.8	12.4
34	TajMahal	27.17538334167778	78.0420778051293	27.2	78
35	Alhambra	37.17628338034626	-3.5881413196798637	37.2	-3.7
36	Lisbon	38.6937815042744	-9.205722231258827	38.6	-9.2
37	Stonehenge	51.17971722162697	-1.8265080767803141	51.2	-1.9
38	Vienna	48.20583653711134	16.364769569036294	48.2	16.3
39	Hollywood	34.13426998562924	-118.32150489117413	34.1	-118.4

Table C.3 Flight Coordinates (cont'd)

40	Sidney	-33.85660618893986	151.21528596884352	-33.8	151.3
41	ChichenItza	20.67942161385877	-88.57074489722596	20.735	-88.6
42	Mecca	21.434039710801805	39.82387621855246	21.352311	39.900127
43	Jerusalem	31.778291495591095	35.23598304636962	31.62138	35.231102
44	Athens	37.97169756908113	23.726716580519756	37.9007	23.801
45	GreatWall	40.94653144682091	116.56360941582882	40.9	116.5
46	ChristRedeemer	-22.951659070744704	-43.210487222255274	-23	-43.3
47	Petra	30.323811512902253	35.480065033032254	30.2	35.6
48	Halicarnassus	37.03806722700596	27.42417004170937	37.1	27.3
49	Rabat	34.001583269655896	-6.840877660216901	34.1	-6.8
50	Teotihuacan	19.81079331782552	-98.96713489694977	19.75435	-98.9
51	BuenosAires	-34.609714754858	-58.39260611982117	-34.7	-59
52	Wellington	-41.10981318152284	174.79730107754747	-41	174.8
53	Canberra	-35.30533142348122	149.12461383994042	-35.4	149.2
54	NewDelhi	28.61746611627941	77.20820427746534	28.7	77.1
55	Stockholm	59.33206027346738	18.069501962059977	59.5	17.95
56	Reykjavik	64.14685901342426	-21.940162890145515	64.05	-22
57	Oslo	59.91960245051236	10.752151201233188	60.001	10.8
58	Helsinki	60.182330556422265	24.939005072686047	60.05	25
59	Barcelona	41.40777303307369	2.173815964078182	41.5	2.2
60	Juno	49.33674991294253	-0.45282603084192935	49.2	-0.4

Table C.4 Flight Coordinates (cont'd)

61	HagiaSophia	41.00876109327952	28.98015353997194	41.1	28.9
62	Ankara	39.918181519139395	32.84942823130222	39.85	32.7
63	Madrid	40.42314987683604	-3.7128716702484383	40.5	-3.8
64	Belfast	54.60941531819268	-5.9282257587349365	54.5	-6
65	Dublin	53.34687538947021	-6.259914546075192	53.26	-6.2
66	Cardiff	51.464679988086644	-3.1615598038625468	51.55	-3.05
67	Alger	38.47449355923938	2.876658676537208	38.55	3
68	Tunis	36.83698618743565	10.141843187799427	37	10.1
69	Tripoli	32.88171032846136	13.190052305083336	32.8	13.2
70	Cairo	30.065196973227696	31.238605331503976	29.9	31.1
71	Everglades	25.379902674245603	-80.92267188348453	25.47543	-81.1011
72	Yellowstone	44.47085691099975	-110.61032348053628	44.371121	-110.50054
73	Seattle	47.60940618698757	-122.3301686350862	47.51008	-122.400654
74	LosAngeles	34.05152055119422	-118.25679126021564	34.15	-118.15
75	Harvard	42.37715346672454	-71.11667083115171	42.25	-71.0021
76	Austin	30.274822639048175	-97.74037197892132	30.34231	-97.8131
77	SaltLakeCity	40.768033831201464	-111.88890903595201	40.654532	-112
78	Winnipeg	49.89988969543453	-97.13927897018432	50	-97.03251
79	Edmonton	53.545763402641036	-113.49028460193757	53.45432	-113.4
80	Halifax	44.651785368053766	-63.57865464133996	44.7211	-63.715436
81	Manilla	14.58967628364151	120.9816137186421	14.64351	121

Table C.5 Flight Coordinates (cont'd)

82	Singapore	1.305750758756111	103.851104243537	1.421411	103.78921
83	Jakarta	-6.209105057258184	106.79991983689449	-6.300231	106.72131
84	Bangkok	13.795585546970337	100.51707696321009	13.3712014	100.7542112
85	AngkorWat	17.817353168441684	103.6607796360959	17.7	103.8
86	Kathmandu	27.691620563499487	85.33660344720998	27.8	85.4
87	PortMoresby	-9.427506423202695	147.1917840971872	-9.5	147
88	Kabul	34.462519252542	69.1202939858207	34.3	69.3
89	Luanda	-5.704122788937915	13.330679561782047	-5.8	13.5
90	Ouagadougou	12.382415544087744	-1.5135958791083626	12.5	-1.6
91	SanJose	11.33566017823603	-84.36394681450975	11.2	-84.5
92	Tallinn	59.443653258633155	24.7355622295007	59.6	24.8
93	Islamabad	33.730893161666636	73.09708233451488	33.5	73.3
94	Kyiv	50.44745695505903	30.536887726785412	50.6	30.7
95	Harare	-17.828042524249586	31.052052523997084	-18	30.95
96	CapeTown	-33.47019101484542	18.024482117743553	-33.2	17.9
97	Antananarivo	-18.864068171574107	47.50830032906943	-19	47.7
98	Riyad	24.773193454274416	46.68920617046243	25	47
99	Seoul	37.53901914607376	126.91381137031487	37.685421	126.7131
100	Oulan-Bator	47.92978322154172	106.918283689398	47.7671123	106.699812

Table C.6 Flight Plans

Flight Number	Distance (km)	Heading (deg)	Speed (m/s)
1	82.84	164	10
2	52.65	294	10
3	134.4	79	10.4
4	49.66	136	5.3
5	54.53	193	7.1
6	96.54	326	10
7	67	88	10
8	97.86	194	7.6
9	37.46	216	8.2
10	125.61	291	6.9
11	86.03	56	7.8
12	84.28	222	10
13	88.06	168	5.3
14	104.17	139	10
15	112.21	299	10
16	77.87	74	10
17	73.78	278	10
18	46.2	339	10
19	62.48	309	7.4
20	46.54	32	5.8

Table C.7 Flight Plans (cont'd)

21	14.998	45	10
22	15.694	216	10
23	16.796	328	10
24	22.91	6	10
25	17.304	336	10
26	17.745	33	10
27	17.335	18	10
28	16.423	306	10
29	9.294	11	10
30	8.364	138	10
31	12.312	42	10
32	8.419	42	7.7
33	12.622	217	10
34	4.981	303	10
35	10.254	284	10
36	10.44	177	8.1
37	5.596	293	10
38	4.844	262	6.7
39	8.17	242	5.1
40	10.042	51	8.2
41	6.889	333	9.1

Table C.8 Flight Plans (cont'd)

42	12.038	139	10
43	17.454	181	8.8
44	10.235	140	10
45	7.439	225	7.47
46	10.624	239	10
47	17.95	140	10
48	12.992	302	7.6
49	11.573	18	6.7
50	9.42	131	8.3
51	56.46	259	10
52	12.213	1	10
53	12.552	146	10
54	13.99	310	10
55	19.86	340	10
56	11.156	195	10
57	5.995	16	10
58	15.098	167	10
59	10.485	12	10
60	15.68	165	10
61	12.17	326	10
62	14.834	239	8

Table C.9 Flight Plans (cont'd)

63	11.285	319	10
64	13.017	200	10
65	10.448	157	10
66	12.232	39	10
67	13.626	51	10
68	18.504	348	10
69	9.133	174	10
70	22.708	216	10
71	20.83	300	10
72	14.107	141	10
73	12.246	205	10
74	14.717	41	10
75	16.99	146	10
76	10.25	317	10
77	15.715	216	10
78	13.5	34	7
79	11.792	149	8
80	13.28	305	6.5
81	6.304	18	10
82	14.586	331	10
83	13.348	220	10

Table C.10 Flight Plans (cont'd)

84	53.7	151	10
85	19.69	131	10
86	13.57	27	10
87	22.527	249	10
88	24.465	137	10
89	21.554	119	10
90	16.093	324	10
91	21.16	224	10
92	17.76	11	9
93	31.816	143	10
94	20.51	34	8
95	21.75	209	10
96	32.19	338	10
97	25.2	126	10
98	40.235	51	10
99	24.033	312	10
100	24.35	222	10

Table C.11 Flight Conditions

Flight Number	WindDir	WindSpd	WindTurb	BaroRnd	GyrRnd	AccRnd	MagRnd	GPSVerr
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	301	1	0	0	0	0	0	0
7	241	2.1	0	0	0	0	0	0
8	41	4.3	0	0	0	0	0	0
9	201	5.7	0	0	0	0	0	0
10	37	3.4	0	0	0	0	0	0
11	260	5.2	0	0	0	0	0	0
12	145	2.9	0.6	0	0	0	0	0
13	225	6.9	0.4	0	0	0	0	0
14	90	3.6	1.2	0	0	0	0	0
15	300	4.6	2.3	0	0	0	0	0
16	0	0	0	1	0	0	0	0
17	0	0	0	1	1	0	0	0
18	0	0	0	0	1	0	1	1
19	0	0	0	0	1	0	0.5	0
20	0	0	0	0.5	0	1	0	1.2

Table C.12 Flight Conditions (cont'd)

21	0	0	0	0	0	1	0	0
22	0	0	0	0	0	2.3	0	0
23	0	0	0	0	1	0	0	0
24	0	0	0	0	1.7	0	0	0
25	0	0	0	1	0	0	0	0
26	0	0	0	2.5	0	0	0	0
27	0	0	0	0	0	0	1	0
28	0	0	0	0	0	0	3.1	0
29	0	0	0	0	0	0	0	1
30	0	0	0	0	0	0	0	3
31	25	3	1	1	0	0	0	0
32	47	6	2	0	0	0	0	0.3
33	253	8	3	0	0.8	1.2	0	0
34	136	6	0.5	0	0	0	0	0
35	278	7	0.2	0.5	0	0	1.7	0
36	331	4.5	1.2	0	0	0	1.1	2.1
37	15	5	0.7	0	0	1.2	0.8	1.6
38	75	6	1.2	0.9	0	1.1	0	0.2
39	45	2	0.3	0	0.5	0	2	0.4
40	195	2.3	0.2	0.5	0	0	0	1.2
41	100	8	1.1	2	1	0.5	0.5	0

Table C.13 Flight Conditions (cont'd)

42	234	7	0.8	0.7	0.4	1.2	0	0.6
43	15	8	0.5	0.9	0.6	0	0.9	0.7
44	176	4.5	0.7	2.1	0	1.1	0.7	0.3
45	234	6.6	0.3	0	0.8	0.9	0.5	0.2
46	23	7	0.1	1	1.4	2.1	0.7	0.8
47	95	3.5	0.6	1.9	0.2	0.1	1	0.6
48	13	0.5	0.1	2.3	1.1	0.8	0.7	1.1
49	45	8	2.2	3.1	1.2	0.5	2	0.7
50	45	6.4	0.5	0.9	0.1	0.7	0.9	0.5
51	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0
53	245	4	1.2	0	0	0	0	0
54	0	0	0	3	0	0	3	0
55	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0
57	0	0	0	0	0	1.3	2	0
58	1	3	0.7	0	1	0	0	0.5
59	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0
62	0	0	0	0	0	0	0	0

Table C.14 Flight Conditions (cont'd)

63	43	4	1	2	0	1	1	0
64	189	5	1.3	2	3.1	1.2	2	1.2
65	0	0	0	0	0	0	0	0
66	0	0	0	0	0	0	0	0
67	43	5	2	4	2	2	0	0
68	0	0	0	2	2	0	1	1
69	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0
71	0	0	0	0	0	0	0	0
72	0	0	0	0	0	0	0	0
73	71	5	1	2	1	0	0	0
74	37	7	2	0	2	2	1	0
75	165	2	0.5	1	0	3	2	1.2
76	228	1	0.2	1	1	1	0	1
77	19	2	0.1	4	2	2.2	3	1
78	321	5	1.1	2	1	3	1	1.7
79	232	4	1	1	1	1.5	2	1.8
80	154	4	1.3	1.2	1	1.2	2	1.5
81	0	0	0	0	0	0	0	0
82	0	0	0	0	0	0	0	0
83	0	0	0	0	0	0	0	0

Table C.15 Flight Conditions (cont'd)

84	0	0	0	0	0	0	0	0
85	0	0	0	0	0	0	0	0
86	0	0	0	0	0	0	0	0
87	312	2	0.2	2	1	0	1	0
88	153	7	2.1	1	3	1	0	2
89	172	3	0.2	2	1	0	1	1.5
90	47	4	1.4	1.2	3	0.5	1.3	2
91	111	6	2.1	1.4	2.3	0.7	1.1	1.4
92	319	4.7	2	1.3	2.1	1.1	2	1.7
93	0	0	0	0	0	0	0	0
94	0	0	0	0	0	0	0	0
95	215	5	2.1	1	1.2	0.7	1.3	0.7
96	131	6.2	1.1	0.8	1.1	0.3	1.2	0.5
97	59	7.1	2.3	2.1	3.2	1.7	2.4	1.5
98	180	6.2	2.4	1.1	2.1	2.4	1.1	0.8
99	317	5.3	2.1	0.6	0.7	0.6	0.9	0.3
100	47	5.5	0.3	1.3	2.3	1.7	1.3	0.9