Investigate feasibility of utilising a neural-networked set of inertial measurement units to compensate for variations in motion of a COTS RC vehicle in a dryland agricultural context.

A thesis

submitted in partial fulfilment

of the requirements for the Degree of

Master of Applied Science

at

Lincoln University

by

Brett Malcolm Davidson

Lincoln University

2023

Abstract of a thesis submitted in partial fulfilment of the requirements for the Degree of Master of Applied Science.

Investigate feasibility of utilising a neural-networked set of Inertial measurement units to compensate for variations in motion of a COTS RC vehicle in a dryland agricultural context.

by

Brett Malcolm Davidson

Image stabilisation is desired for efficient identification of objects in the path of a self-driving vehicle. The gyroscope and accelerometer of an inertial measurement unit (IMU) can be used to derive the movement of a vehicle, which can then be used by a rotation matrix to compensate for this movement, but a gyroscope has inherent “drift” errors, and while the accelerometer of an IMU is more accurate, it has a slower response time, which reduces the detection rate.

Kalman filters are often used in industry to fuse gyroscope and accelerometer data to reduce the effects of drift, noise, and other gaussian-based errors but these are computationally intensive for the sort of lightweight processor that a radio-controlled car could be expected to power. A complementary filter such as Madgwick’s is a simpler and less processor-intensive solution with claims that the method is just as accurate.

This project compares complementary and Kalman filter operation with a neural network when performing sensor fusion and investigates if using a neural network of multiple IMUs reduces errors and enhances performance compared to a single IMU in the context of establishing Euler angles of roll and pitch movement to stabilise a video feed of a consumer-level camera on a moving commercial off-the-shelf radio control vehicle.

It is demonstrated that there is no statistically significant advantage in using multiple IMUs if these are kept in the same horizontal plane, that a minimum of a three-layer MATLAB NARXNET filter provides the equivalent accuracy of a Kalman filter with similar processing times, and that the Madgwick $IECF_6$ complimentary filter, the radial basis factor neural network and multi-level perceptron neural network models are not fit for this purpose.

Keywords: IMU, image stabilisation, neural network, translational sensor drift.

Acknowledgements

I would like to thank my supervisors, Stuart Charters, and Mos Sharifi, for their expertise and guidance throughout this project. I would also like to acknowledge the support of Shane Dye of Ara Institute of Canterbury for his assistance in helping me understand the mathematical concepts in this project.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(Brett Malcolm Davidson)

Table of Contents

Abstract **1**

Acknowledgements **2**

Declaration **2**

Table of Contents **3**

List of Tables **5**

List of Code **7**

List of Equations **7**

List of Figures **7**

Chapter 1 Introduction **11**

1.1 Motivation 11

1.1.1 NZ Farm information and desired data 11

1.1.2 Data Gathering Methods 12

1.1.3 Utilising a rover as a data mule 20

1.1.4 Obstacle Avoidance 21

1.1.5 Image stabilisation 30

1.2 Research Objective 42

1.3 Thesis Outline 42

Chapter 2 Literature Review **43**

Chapter 3 Method **49**

3.1 Experimental Setup and Calibration 51

3.1.1 Calibration. 52

3.2 Experiments 60

Chapter 4 Results **67**

4.1 Stationary vehicle filter performance 67

4.2 Rolling vehicle filter performance 69

4.3 Pitching vehicle filter performance 74

4.4 Performance on a varied movement vehicle. 76

4.4.1 Neural network evaluation using the CIMU 82

4.4.2 Filter and Neural network performance with the three Front IMUs 100

4.4.3 Neural network of a varied-motion vehicle: All IMUs 107

4.5 Would using a magnetometer assist accuracy? 111

4.6 Examining forward (rotating) movement. 118

4.7 Exploring Z-axis bias values. 131

4.8 Exploring X and Y axis bias values. 133

4.9 Summary of results. 136

Chapter 5 Discussion and Analysis **137**

5.1 Introduction 137

5.2 Experimental Design Analysis 137
5.3 Filter results analysis. 139
5.4 Neural Network results analysis. 140
5.5 Research Objectives analysis. 141
5.6 Summary 142

Chapter 6 . Conclusion and Future Works 144
6.1 Future work. 145
6.2 Summary 146

Chapter 7 Appendices 147
7.1 Appendix 1. Code 147
7.2 Appendix 2. Calibration results. 148
7.3 Appendix 3. Equipment Specifications. 150
7.4 Appendix 4. IMU-29048 IMU specifications 152

Chapter 8. References 153

List of Tables

Table 1.1 Commerical Long-range IoT radio standards compared. 14
Table 1.2 Backhaul (long range) communication protocols compared. 16
Table 1.3 Short range communication protocols compared 18
Table 1.4 Communication protocol options available on a range of microcomputers.....18
Table 1.5 Single obstacle-avoiding Sensor Summary 27
Table 1.6 Simple representation of a single video image. 29
Table 3.1 First three lines of stationary gyroscope and accelerometer results (central IMU). 58
Table 3.2 Central IMU gyroscope calibration values.....59
Table 3.3 Central IMU accelerometer slope and offset table calibration values.....59
Table 3.4 Sample of Euler angle results from the robot arm using tool arm sensors while at rest..60
Table 3.5 Joint angles of Robot at rest 60
Table 3.6 Experiments. 61
Table 3.7 Robot Movement Program: Roll (Steps 2 and 3 are repeated).....62
Table 3.8 Robot Movement Program: Pitch.....62
Table 3.9 Robot Movement Program: Varied 62
Table 3.10 Robot Movement Program: Varied-Untrained. 63
Table 3.11 Robot Movement Program – Rotation 63
Table 3.12 Robot Movement Program – Rotation-Untrained. 64
Table 4.1 Madgwick filter results (CIMU) from a stationary vehicle.....68
Table 4.2 MLP NN training results (CIMU) on a varied-motion vehicle with default layer size of 10. 82
Table 4.3 MLP NN training progress (CIMU) on a varied-motion vehicle with default layer size of 10. 82
Table 4.4 MLP NN training progress (CIMU) on a varied-motion vehicle with layer size of 20. 85
Table 4.5 MLP NN training results (CIMU) on a varied-motion vehicle with layer size of 20. 85
Table 4.6 MLP NN training progress (CIMU) on a varied-motion vehicle with layer size of 5. 87
Table 4.7 MLP NN training results (CIMU) on a varied-motion vehicle with layer size of 5. 88
Table 4.8 Training results on CIMU MLP NN with three layers. Training took 0.8999 seconds. 88
Table 4.9 Training results on CIMU MLP NN with two layers. Training took 0.6931 seconds. 88
Table 4.10 Training results on CIMU MLP NN with one layer. Training took 0.6315 seconds.....88
Table 4.11 CIMU 10-layer MLP NN Training Results from Untrained Varied angle movements.....89
Table 4.12 Normalised CIMU 10-layer MLP NN Training Results from varied motion. 89
Table 4.13 Normalised CIMU 10-layer MLP NN Training progress from varied motion. 89
Table 4.14 Normalised CIMU 10-layer MLP NN results from Untrained motion. 89
Table 4.15 Normalised CIMU 10-layer NARXNET NN Training results from varied motion. 90
Table 4.16 Normalised CIMU 10-layer NARXNET NN Training progress from varied motion. 90
Table 4.17 Normalised CIMU 10-layer NARXNET NN results from untrained motion. 90
Table 4.18 Normalised Front IMUs 10-layer MLP NN Training progress from varied motion.....100
Table 4.19 Normalised Front IMUs 10-layer MLP NN Training results from varied motion. 100
Table 4.20 Normalised Front IMUs 10-layer MLP NN results from untrained motion. 100
Table 4.21 Normalised Front IMUs 5-layer MLP NN training progress from varied motion. 101
Table 4.22 Normalised Front IMUs 5-layer MLP NN training results from varied motion. 101
Table 4.23 Normalised Front IMUs 5-layer MLP NN results from untrained varied motion. 101
Table 4.24 Normalised Front IMU 10-layer NARXNET NN training progress from varied motion. 101
Table 4.25 Normalised Front IMUs 10-layer NARXNET NN training results from varied motion ...102
Table 4.26 Normalised Front IMUs 10-layer NARXNET NN results from untrained varied motion102
Table 4.27 Normalised Front IMU 5-layer NARXNET NN training progress from varied motion. ..102
Table 4.28 Normalised Front IMUs 5-layer NARXNET NN training results from varied motion 102
Table 4.29 Normalised Front IMUs 5-layer NARXNET NN results from untrained varied motion..102
Table 4.30 Normalised Front IMU 3-layer NARXNET NN training progress from varied motion. ..102
Table 4.31 Normalised Front IMUs 3-layer NARXNET NN training results from varied motion 103

Table 4.32 Normalised Front IMUs 3-layer NARXNET NN results from untrained varied motion..103
Table 4.33 Normalised All IMUs 10-layer MLP NN training progress from varied motion. 107
Table 4.34 Normalised All IMUs 10-layer MLP NN training results from varied motion. 107
Table 4.35 Normalised All IMUs 10-layer MLP NN results from untrained varied motion. 107
Table 4.36 Normalised All IMUs 5-layer MLP NN training progress from varied motion. 107
Table 4.37 Normalised All IMUs 5-layer MLP NN training results from varied motion. 107
Table 4.38 Normalised All IMUs 5-layer MLP NN results from untrained motion. 108
Table 4.39 Normalised All IMUs 10-layer NARXNET NN training progress from varied motion.....108
Table 4.40 Normalised All IMUs 10-layer NARXNET NN training results from varied motion.....108
Table 4.41 Normalised All IMUs 10-layer NARXNET NN results from untrained motion. 108
Table 4.42 Normalised All IMU 5-layer NARXNET NN training progress from varied motion. 108
Table 4.43 Normalised All IMUs 5-layer NARXNET NN training results from varied motion.....109
Table 4.44 Normalised All IMUs 5-layer NARXNET NN results from untrained varied motion 109
Table 4.45 Normalised All IMU 3-layer NARXNET NN training progress from varied motion. 109
Table 4.46 Normalised All IMUs 3-layer NARXNET NN training results from varied motion.....109
Table 4.47 Normalised All IMUs 3-layer NARXNET NN results from untrained varied motion 109
Table 4.48 Normalised CIMU 10-layer MLP NN training results with magnetometer under varied motion. 113
Table 4.49 Normalised CIMU 10-layer MLP NN training progress with magnetometer under varied motion. 113
Table 4.50 Normalised CIMU 20-layer MLP NN training progress with magnetometer under varied motion. 115
Table 4.51 Normalised CIMU 10-layer MLP NN training results from rotational motion.....120
Table 4.52 Normalised CIMU 10-layer MLP NN training results from rotational motion.....120
Table 4.53 Normalised Front IMUs 10-layer MLP NN training results from rotational motion.....122
Table 4.54 Normalised Front IMUs 10-layer MLP NN training progress from rotational motion. .122
Table 4.55 Normalised All IMUs 10-layer MLP NN training results from rotational motion. 123
Table 4.56 Normalised All IMUs 10-layer MLP NN training progress from rotational motion. 123
Table 4.57 Normalised All IMUs 20-layer MLP NN training results from rotational motion. 123
Table 4.58 Normalised All IMUs 20-layer MLP NN training progress from rotational motion. 123
Table 4.59 Normalised All IMUs 10-layer MLP NN results from untrained rotational motion. 124
Table 4.60 Normalised CIMU 10-layer NARXNET NN training progress from rotational motion. ..124
Table 4.61 Normalised CIMU 10-layer NARXNET NN training results from rotational motion. 124
Table 4.62 Normalised CIMU 10-layer NARXNET NN results from untrained rotational motion...124
Table 4.63 Front-left IMU Accelerometer measurements after inverting the IMU 132
Table 4.64 Front-left IMU Gyrometer measurements after inverting the IMU.....133
Table 4.65 Accelerometer measurements from rotating rear IMUs by pi/2(90 degrees).....134
Table 4.66 Gyrometer measurements from rotating rear IMUs by pi/2(90 degrees) 135
Table 7.1 Script Names and Functions 147
Table 7.2 IMU Error Coefficients produced from initial calibration (rounded to 8 decimal places) 148

List of Code

All code can be found at the project github location, https://github.com/BratNZ/Thesis

Details of code can be found in Table 7.1 found in Appendix 7.1.

List of Equations

Equation 1.1 Acceleration at rest where g=9.81m/s^2. 34
Equation 1.2 Acceleration on a tilted frame 34
Equation 1.3 The mathematical relationship between level and tilted frames.....34
Equation 1.4 Mathematical equations for pitch and roll 34

List of Figures

Figure 1.1 Obstacle detecting Sensor Types (Image from author).....22
Figure 1.2 Diagram of conventional roll, pitch and yaw angles (Image from Author). 33
Figure 1.3 Fisheye Lens capture example. Copyright free Image from (Fpatrocini, n.d.) 37
Figure 1.4 Neuron construction (unit part of the diagram). Image from (Samarasinghe, 2006).....38
Figure 1.5 Example functionality of neural networks for scientific data modelling: (a) fitting models to data or regression, (b) complex classification tasks, (c) discovering clusters in data, and (d) time-series forecasting. Image from (Samarasinghe, 2006) 39
Figure 1.6 Some neural network topologies: (a) single-layer perceptron, (b) linear neuron, (c) multilayer perceptron, (d) competitive networks, (e) self-organising feature maps and (f) recurrent networks. Image from (Samarasinghe, 2006).....40
Figure 2.1 10 layer MLP model diagram with six inputs and two outputs. 46
Figure 2.2 10-layer NARXNET Model diagram with six inputs and two outputs.....47
Figure 2.3 RBF model diagram with six inputs and two outputs. 48
Figure 3.1 Baseboard diagram showing components (Image from Author).....49
Figure 3.2 Method Flowchart (Image from Author) 50
Figure 3.3 Robotic Arm position showing mounted baseboard for roll and pitch experiments 53
Figure 3.4 Robotic Arm position showing mounted baseboard for all other experiments 53
Figure 3.5 Varied movement (CIMU) poor alignment plot (Samples: 1000 to 1300) from 4999. 54
Figure 3.6 Overall “best-fit” alignment plot (CIMU) showing OK alignment at this level.....55
Figure 3.7 “Best-fit” alignment (CIMU) – enlarged view of samples from 100-300. 55
Figure 3.8 “Best-fit” alignment (CIMU) – enlarged view of samples from 1000-3000. 55
Figure 3.9 “Best-fit” alignment (CIMU) – enlarged view of samples from 3000-3300. 56
Figure 3.10 “Best-fit” alignment (CIMU) – enlarged view of samples from 4500-4999. 56
Figure 3.11 Overall alignment of IMU and robot arm.....57
Figure 3.12 Alignment of IMU and robot arm from samples 1000-1999 57
Figure 3.13 Allignment of IMU and robot arm from samples 4000-4999.....57
Figure 4.1 Madgwick filter enlarged results (CIMU) on stationary vehicle with default settings.....68
Figure 4.2 Kalman Filter results (CIMU) on stationary vehicle with default settings.....69
Figure 4.3 Madgwick results from rolling movement (CIMU) with beta of 0.1 and sample rate of 1001. 70
Figure 4.4 Madgwick results from rolling movement (CIMU) with a beta value of 1.0. 70
Figure 4.5 Madgwick filter on a rolling vehicle (CIMU) with a beta value of 10. 71
Figure 4.6 Madgwick filter on a rolling vehicle (CIMU) with a gain of 15. 72
Figure 4.7 Filter and Robot arm data for a rolling vehicle (CIMU): Samples 1-600. 73

Figure 4.8 Averaged roll value of filters against robot arm movements (CIMU): Samples 1-600. ...73
Figure 4.9 Filter and robot arm results for a pitching vehicle (CIMU): Samples 1-600.....74
Figure 4.10 Average Filter Pitch values against robot arm movements (CIMU): Samples 1-600 75
Figure 4.11 Average of front 3 IMU results against robot arm movements: Samples 1-600 76
Figure 4.12 Madgwick and Kalman results on varied motion with SR of 8.33 (CIMU): Samples 1-600. 77
Figure 4.13 Madgwick and Kalman results on varied motion with SR of 4999 (CIMU): Samples 1-600. 78
Figure 4.14 Madgwick and Kalman results on varied motion with SR of 100 (CIMU): Samples 1-600. 78
Figure 4.15 Madgwick and Kalman results on varied motion with SR of 50 (CIMU): Samples 1-600. 79
Figure 4.16 Madgwick and Kalman results on varied motion with SR of 500 (CIMU): Samples 1-600. 79
Figure 4.17 Averaged results on varied motion with SR of 50 (CIMU): Samples 1-600.....80
Figure 4.18 Filter results on varied motion with SR of 100 and beta of 0.5 (CIMU): Samples 1-600. 80
Figure 4.19 Filter results on varied motion with SR of 100 and beta of 5.0 (CIMU): Samples 1-600. 81
Figure 4.20 Averaged results on varied motion with SR of 100 and beta of 5.0 (CIMU): Samples 1-600. 81
Figure 4.21 MLP NN performance plot (CIMU) on varied-motion with default layer size of 10. 83
Figure 4.22 MLP NN Error histogram (CIMU) on varied-motion with default layer size of 10. 83
Figure 4.23 MLP NN regression plot (CIMU) on varied motion with default layer size of 10.....84
Figure 4.24 MLP NN training state plot (CIMU) on varied-motion with default layer size of 10.....84
Figure 4.25 MLP NN performance plot (CIMU) on varied-motion with layer size of 20.....85
Figure 4.26 MLP NN error histogram (CIMU) on varied-motion with layer size of 20.....86
Figure 4.27 MLP NN regression plot (CIMU) on varied-motion with layer size of 20. 86
Figure 4.28 MLP NN training state plot (CIMU) on varied-motion with layer size of 20. 87
Figure 4.29 Normalised NARXNET NN training state (CIMU) on varied-motion with layer size of 10. 90
Figure 4.30 Normalised NARXNET NN performance (CIMU) on varied-motion with layer size of 10. 91
Figure 4.31 Normalised NARXNET NN error histogram (CIMU) on varied-motion with layer size of 10. 91
Figure 4.32 Normalised NARXNET NN time-series state (CIMU) on varied-motion with layer size of 10. 92
Figure 4.33 Normalised NARXNET NN autocorrelation (CIMU) on varied-motion with layer size of 10. 92
Figure 4.34 Normalised NARXNET NN input-error plot (CIMU) on varied-motion with layer size of 10. 93
Figure 4.35 Determining minimal MSE values for RBF network. 94
Figure 4.36 RBF NN Performance Plot from CIMU on varied motion. Spread=1 and Neurons=400. 94
Figure 4.37 RBF Roll Plot using untrained varied motion (CIMU). MSE=6.96e-06, SF=1. 95
Figure 4.38 RBF Pitch Plot using untrained varied motion (CIMU). MSE=6.96e-06, SF=1. 95
Figure 4.39 RBF Roll Plot using untrained varied motion (CIMU). MSE=9.49e-06, SF=2. 96
Figure 4.40 RBF Pitch Plot using untrained varied motion (CIMU). MSE=9.49e-06, SF=2. 96
Figure 4.41 RBF Roll Plot using untrained varied motion (CIMU). MSE=9.49e-06, SF=0.5 97
Figure 4.42 RBF Pitch Plot using untrained varied motion (CIMU). MSE=9.49e-06, SF=0.5 97
Figure 4.43 RBF Roll Plot using untrained varied motion (CIMU). MSE=6.49e-06, SF=0.5 98
Figure 4.44 RBF Pitch Plot using untrained varied motion (CIMU). MSE=6.49e-06, SF=0.5. 98
Figure 4.45 RBF Roll Plot using untrained varied motion (CIMU). MSE=1.037-e05, SF=4. 99
Figure 4.46 RBF Pitch Plot using untrained varied motion (CIMU). MSE=1.037-e05, SF=4. 99

Figure 4.47 RBF MSE Determination from Front IMU data 103
Figure 4.48 RBF Roll Plot using untrained varied motion (Front IMUs). MSE=3.15123e-06, SF=1. 104
Figure 4.49 RBF Pitch Plot using untrained varied motion (Front IMUs). MSE=3.15123e-06, SF=1 104
Figure 4.50 RBF Roll Plot using untrained varied motion (Front IMUs). MSE= 4.12939e-06, SF=2 105
Figure 4.51 RBF Pitch Plot using untrained varied motion (Front IMUs). MSE= 4.12939e-06, SF=2 105
Figure 4.52 RBF Roll Plot using untrained varied motion (Front IMUs). MSE= 4.12939e-06, SF=0.5 106
Figure 4.53 RBF Pitch Plot using untrained varied motion (Front IMUs). MSE= 4.12939e-06, SF=0.5 106
Figure 4.54 RBF MSE determination for all IMUs.....110
Figure 4.55 RBF Roll Plot using untrained varied motion (All IMUs). MSE=1.93051e-06, SF=1.0 ..110
Figure 4.56 RBF Pitch Plot using untrained varied motion (All IMUs). MSE=1.93051e-06, SF=1.0 111
Figure 4.57 Kalman and Madgwick CIMU angles (with Magnetometer) against robot arm angles. 112
Figure 4.58 MLP NN training state (CIMU) on varied-motion with 10 layers, using magnetometer. 113
Figure 4.59 MLP NN performance (CIMU) on varied-motion with 10 layers, using magnetometer. 114
Figure 4.60 MLP NN error histogram (CIMU) on varied-motion with 10 layers, using magnetometer. 114
Figure 4.61 X-Axis CIMU Magnetometer values against varied-motion movements: Samples 1-600. 115
Figure 4.62 Y-Axis CIMU Magnetometer values against varied-motion movements: Samples 1-600. 116
Figure 4.63 Z-Axis CIMU Magnetometer values against varied-motion movements: Samples 1-600. 116
Figure 4.64 All axes CIMU Magnetometer values against varied-motion movements: Samples 1-600. 117
Figure 4.65 All axes CIMU Raw Magnetometer values: Samples 1-600. 117
Figure 4.66 Kalman results (CIMU) on rotational-motion: Samples 1-600. 118
Figure 4.67 Madgwick results (CIMU) on rotational-motion: Samples 1-600. 119
Figure 4.68 Normalised MLP NN regression (CIMU) on rotational-motion with layer size of 10...120
Figure 4.69 Normalised MLP NN training state (CIMU) on rotational-motion with layer size of 10. 121
Figure 4.70 Normalised MLP NN performance (CIMU) on rotational-motion with layer size of 10. 121
Figure 4.71 Normalised MLP NN error histogram (CIMU) on rotational-motion with layer size of 10. 122
Figure 4.72 Normalised NARXNET NN reponse (CIMU) on rotational-motion with layer size of 10. 125
Figure 4.73 Normalised NARXNET NN training state (CIMU) on rotational-motion with layer size of 10. 125
Figure 4.74 Normalised NARXNET NN Performance (CIMU) on rotational-motion with layer size of 10. 126
Figure 4.75 Normalised NARXNET NN Error histogram (CIMU) on rotational-motion with layer size of 10. 126
Figure 4.76 RBF MSE determination for CIMU under rotational movement.....127
Figure 4.77 RBF Roll Plot using untrained rotational motion (CIMU). MSE=2.5806e-05, SF=1.0 ...127
Figure 4.78 RBF Pitch Plot using untrained rotational motion (CIMU). MSE=2.5806e-05, SF=1.0 .128
Figure 4.79 RBF MSE determination for Front IMUs under rotational movement. 128
Figure 4.80 RBF Roll Plot using untrained rotational motion (Front IMUs). MSE=1.05961e-05, SF=1.0 129

Figure 4.81 RBF Pitch Plot using untrained rotational motion (Front IMUs). MSE=1.05961e-05, SF=1.0 129
Figure 4.82 RBF MSE determination for All IMUs under rotational movement. 130
Figure 4.83 RBF Roll Plot using untrained rotational motion (All IMUs). MSE=5.76837e-06, SF=1.0 130
Figure 4.84 RBF Pitch Plot using untrained rotational motion (All IMUs). MSE=5.76837e-06, SF=1.0 131
Figure 4.85 Front-Left IMU Zaxis Accelerometer values compared after inverting the IMU. 132
Figure 4.86 Front-left IMU Zaxis Gyrometer values compared after inverting the IMU 133
Figure 4.87 Accelerometer values after rotating rear IMUs by pi/2 (90 degrees).....134
Figure 4.88 Gyroscope values after rotating rear IMUs by pi/2 (90 degrees) 135
Figure 7.1 IMU Calibration graphs for central IMU.....148
Figure 7.2 IMU Calibration graphs for Front Left IMU 149
Figure 7.3 IMU Calibration graphs for Front Right IMU 149
Figure 7.4 IMU Calibration graphs for Rear Left IMU.....149
Figure 7.5 IMU Calibration graphs for Right Rear IMU 150

Chapter 1 Introduction

1.1 Motivation

This chapter outlines the requirements around obtaining animal information on a New Zealand farm in a timely and cost-effective manner and a broad discussion of viable solutions for readers not familiar with New Zealand animal data management. The chapter discusses the types of data often recorded and methods of delivering that data back to the farmer. Where an unattended ground vehicle (UGV or rover) is required for data capture, various obstacle detection methods are discussed (including choices of sensors) and an overview of camera-based avoidance techniques is presented. The issue of image stabilisation when using camera-based systems and common techniques for mitigating this is presented and the research objective is outlined.

1.1.1 NZ Farm information and desired data

To maintain animal welfare and to obtain the best possible profit, beef cattle farmers in rural New Zealand need to acquire accurate and timely data on the state of their livestock (Swain et al., 2013) . In New Zealand, 80% of beef production is exported and international importers wish to purchase pasture-fed healthy beef (Beef and Lamb NZ, 2017).
The total area of New Zealand comprises 26.8 million hectares with approximately three quarters of this area above 200 metres in altitude. In 2017, 8.765 million hectares were used for beef and sheep farming spread across 23,403 holdings, averaging 374.5 hectares per farm. 92% of these commercial sheep and beef farms were owner-operated and most farmers farm both sheep and beef cattle as these are complementary for pasture management. In 2020, these farms hosted 26 million sheep and 3.9 million beef cattle, averaging 1,110 sheep and 166 beef cattle per farm. The average figures are somewhat skewed as approximately 45% of the total stock are on farms holding more than 500 cattle (Beef & Lamb NZ, 2021).

Examples of the type of data collected on a particular animal might include animal movement (to determine most visited feeding and watering areas and to track cattle movement in case of disease), weight, as this is the main measurement used to determine price, along with animal health and fat percentage. Monitoring of farm and pasture conditions (weather, pasture quality, water quality, feed and water levels, images, etc) is also desirable as these have an impact on cattle quality (Swain et al., 2013). Due to the higher value of cattle versus sheep, sensors are normally only implemented on cattle (Beef and Lamb NZ, 2017). Optionally, data on the activity of each animal (often obtained by accelerometers around the animal's neck) could be used to determine if an animal is in distress compared to resting or sleeping (Bailey et al., 2018).

To track animal movement, a Global Positioning System (GPS) sensor with data logging can be attached to the animal (Ropert-Coudert & Wilson, 2005). To gather weight statistics, the animal must walk over some form of weight scale or via estimation (hoisting an animal to weigh it is impractical) (Wangchuk et al., 2017). Gathering weighbridge data where a farmer is not present to ensure the animal stays still (to ensure an accurate measurement) may introduce errors, but this project will not explore these further.

The main problem with gathering data from an animal that moves is efficiently transmitting that data back to the farmer. In the following sections various data gathering methods are discussed, and the rationale for using a rover for data capture is outlined. Besides battery life and route planning, the next major issue of using a rover on alpine terrain is object avoidance, and various sensor types are presented, and the reason for using a camera as the main sensor (with assistance from others) is outlined. One issue effecting the use of a camera is image stabilisation, especially when a rover could be travelling at high speeds. Various image stabilisation methodologies are discussed and the utilisation of IMUs is outlined.

1.1.2 Data Gathering Methods

There are three main methods available to collect data from a remote location.
Radio Frequency (RF), a data mule (autonomous device travels to sensors to collect data) and a hybrid solution of these.

1.1.2.1 Radio frequency data collection.

Possible radio frequency (RF) implementations include satellite (De Sanctis et al., 2016) , cellular service (Gaddam & Rai, 2018) , dedicated radio links such as WiMAX (Andreev et al., 2015), Long Range Wireless Area Network (LoRaWAN) (Adelantado et al., 2017; P. S. Cheong et al., 2017; Haxhibeqiri et al., 2018b; Lavric & Popa, 2018),Wi-Fi-Halo (Tian et al., 2021), Bluetooth (Gomez et al., 2012; Team, 2019), ZigBee (Baronti, P Pillai, P Chook, V.W.C Chessa, S Gotta, A Hu, 2007; Gheorghiu & Iordache, 2018) or similar (listed in order of greatest range) (Al-Sarawi et al., 2017).
These systems transmit the signals from the sensors to either a central location (this design is called “hub and spoke”) or in a form of “mesh“ where sensors transmit via other sensors until they are in range of the farmer (Sethi & Sarangi, 2017) .
Beef cattle are generally farmed in hill country so a direct line of sight from a sensor on the animal to a central radio (or satellite) point is not always possible (Spark NZ, 2023; Starlink, 2022).
Animals tend to travel in groups so a mesh configuration is not practical and would be cost-prohibitive over the distances involved (Ramseyer et al., 2009).

RF techniques can be split into two types; long range (backhaul) or short range.

1.1.2..1 Long range or Back haul

Long range radio frequency systems include satellite, cellular service, dedicated radio links and Long-Range Wireless Area Network (LoRaWAN). The strengths and weaknesses of these systems in an alpine farmland context is discussed in this section.

Satellite systems need a clear line of sight to a minimum of 4 satellites greater than 15 degrees above the horizontal plane for GPS location, but data communication can be achieved with one satellite. While GPS is a free service, satellite data is expensive, location-restrained, power hungry and slow although companies such as Iridium are improving their 1.4kbs links to 512kbs and StarLink has recently expanded its operations in New Zealand with promises of greater than 50Mb/s upload speeds. StarLink, as of writing, costs $370 to install and $160 per month and has extensive coverage (Starlink, 2023). The main advantage of using satellite communications is that the architecture can be flattened to just sensors and a back-end infrastructure, which aids simplicity.
Disadvantages are that satellite data transmission is expensive as each animal would need a satellite transmitter and data plan. Confidentiality of data would also need to be considered.
GPS use is free, and transmitters are ubiquitous and are low-cost, so satellite transmission should be reserved for GPS location data only.

<u>Cellular systems.</u>

Cellular technologies (4G, 5G). Public cellular technology support in the high country is sparse although recent OneNZ announcements are supplementing cellular support with the use of StarLink satellites, to be released in 2025 (OneNZ, 2023a).

Current fourth generation (4G) coverage is possible in rural high country but there are pockets of no coverage from commercial providers (OneNZ, 2023b; Spark, 2021). Coverage would need to be evaluated and confirmed on each site.
Purchasing a subscriber identity module (SIM) card for each animal would quickly become expensive, even without considering potential damage from animal activity, but like GPS data, does reduce the architecture down to sensor devices and the cloud.
It is possible to utilise generic consumer SIM technology to perform connectivity but the customer would be paying for voice capability where that is not required.

Fifth generation (5G) cellular technologies (Cat-M1, LTE-M, Narrowband-IoT) chipsets are cheaper than 4G chips but due to limited demand, costs are still high. This will change as demand increases. Cat-M1 operates at 1.4 MHz bandwidth and this wider bandwidth allows Cat-M1 to achieve particularly good data rates (up to 1 Mbps – generally 200-400kbs) with low latency and device positioning capabilities. Cat-M1 supports voice calls and connected mode mobility. Cat-M1 has a similar range as 4G (10-15km) since it extends the same LTE-based cellular technology underpinning these.

LTE-M is a competitive system to Cat-M1 with similar features, described in Table

Narrowband-Internet of Things (NbIoT) sacrifices throughput (250kbs maximum upload with a payload size of up to 1600 bytes) for extended range and improved power usage.

All 5G technologies mentioned here can support more than 55,000 simultaneous clients.

Table 1.1 Commerical Long-range IoT radio standards compared.

	LTE Cat M1	NB-IoT	GSM-IoT
Deployment	In-band LTE	In-band & Guard-band LTE, standalone	In-band GSM
Coverage*	155.7 dB	164 dB	164 dB with 33dBm power class 154 dB, with 23dBm power class
Downlink	OFDMA, 15 KHz tone spacing, Turbo Code, 16 QAM, 1 Rx	OFDMA, 15 KHz tone spacing, 1 Rx	TDMA/FDMA, GMSK, PSK (optional), 1 Rx
Uplink	SC-FDMA, 15 KHz tone spacing Turbo code, 16 QAM	SC-FDMA, 15 KHz and 3.75KHz tone spacing, Turbo code	TDMA/FDMA, GMSK and 8PSK (optional)
Bandwidth	1.08 MHz	180 KHz	200kHz per channel. 600KHz to 2.4MHz
Peak rate (DL/UL)	1 Mbps for DL and UL	DL: ~50 kbps UL: ~50 kbps (multitone) : 20 kbps (single tone)	For DL and UL (4 timeslots): 70 kbps (GMSK), 240kbps (8PSK)
Duplexing	FD & HD (type B), FDD & TDD	HD (type B), FDD	HD, FDD
Power saving	PSM, ext. I-DRX, C-DRX	PSM, ext. I-DRX, C-DRX	PSM, ext. I-DRX
Power class	23 dBm	23 dBm	33 dBm or 23dBm

<u>Dedicated link RF options</u>

WiMAX (Worldwide operability over Microwave) is a technology using Microwave links.
WiMAX does not require line of sight to operate and has further range than conventional "Wi-Fi" wireless networking (30-50km maximum limit for line of sight). Power usage is very good compared to cellular networks (Deruyck et al., 2010). There are two main types of WiMAX - fixed and mobile, and these are quite different in infrastructure. There are few public WiMAX deployments in NZ and none in rural spaces so the farmer would need to invest in WiMAX infrastructure and deploy stations in the appropriate places to get coverage to and from the remote areas. WiMAX appears to be losing the battle for spectrum to cellular providers and a "recent" (2017) spectrum analysis report indicates there is little intention for WiMAX to be implemented in a large scale in NZ (New Zealand IoT Alliance, 2019). As such, for futureproofing rationales, WiMAX Is not recommended for use.

SigFox is a proprietary system where devices connect to base stations which connect to each other via point-point links (maximum range of 50km) back to the SigFox cloud. Each sensor can report up to twelve bytes in a message (26 bytes in a frame), with a maximum of 140 messages per day. Transmission speed is either 100 or 600 bits per second (SigFox, 2023b).
Coverage of some areas in New Zealand is suggested at Sigfox's commercial website (SigFox, 2023a) however both the proprietary nature of SigFox and it's low transmission rates, make it unappealing as a solution, and SigFox is not recommended for use.

LoRa is a competitor to SigFox. This is a low power communications protocol designed to send small packets of data at regular intervals with low-power usage. It is a proprietary protocol owned by Semtech. LoRaWAN is an open source point to multipoint messaging protocol built to utilise the Lora communications protocol (*What Is LoRaWAN® | LoRa Alliance®*, n.d.).
LoRaWAN has very low power requirements and a practical range of up to 3km in an urban environment (Augustin et al., 2016) and up to 15km in an rural environment (Haxhibeqiri et al., 2018a) but has a duty cycle limit (1% duty equals 36 seconds of communications time per hour for each device). It prefers line-of-sight communication to perform well (Haxhibeqiri et al., 2018a). Transmission speed varies from 300bps to 37.5kbps with a maximum payload of 246 bytes.
The theoretical maximum nodes in a LoRa system are 10,000 but LoRaWAN has scaling issues of a maximum of 8,000 devices (1000 nodes per channel with 8 channels) at a collision rate of 95% (Lavric & Popa, 2018). More stations are required to alleviate this, which is possible as LoRaWAN operates in a star of stars fashion where every station receives the transmissions of all devices around it.

In NZ, Spark concentrate LoRaWAN in the main urban centres (Spark NZ, 2023), so in many sites, the farmer would need to invest in LoRaWAN infrastructure and deploy stations in the appropriate

places to get coverage to and from the back-end stations.

There are no fees at present for operating LoRaWAN systems but a license is required. (*Short Range Devices GURL | Radio Spectrum Management New Zealand*, n.d.)

Each LoRaWAN edge station requires a TCP/IP-based backhaul protocol to deliver content to the cloud. This could be possible within an area covered by one of the 4G or 5G satellite protocols at an additional cost and complexity.

Dedicated radio links are a solution but would require specific designs for each station. Annual licensing costs also apply (*Fixed Link Licence | Radio Spectrum Management New Zealand*, n.d.) Depending on the distances it might be more cost-effective to implement direct links but this technology implementation would be site-specific and costly, and will not be evaluated in this review.

Table 1.2 Backhaul (long range) communication protocols compared.

Protocol	4G	Cat-M1/LTE-M	NBIoT	LoRaWAN
Standard		3GPP LTE CatNB1	3GPP LTE CatNB2	IEEE 802.15.4g
Frequency	Licensed	Licenced	Licensed	915 MHz
Range	10 km	10 km	12 km	5 km
Transfer Rate	2 Mbps	300-400 kbs	20-127 kbs	50 kbs
Power usage	250 mW	220 mW	200 mW	125 mW
# clients	55,000	55,000	55,000	10,000
Availability	Very high	Very high	Very high	Limited
Major Limitations	Range	Data rate and capacity	Range and data rate	Data rate, capacity and range
Recommended Applications	Where Video/Voice are required	Where Images/Voice are required	Where Images are required	For small datasets generated hourly.
Farm Use	Not ideal	Ideal for large datasets if available	Ideal for medium sized datasets if available	Ideal for small datasets if available

1.1.2..2 ***Short Range RF options***

ZigBee is based on the IEEE standard 802.15.4 (Baronti, P Pillai, P Chook, V.W.C Chessa, S Gotta, A Hu, 2007) for small personal LAN communication with a range from 10-100m (for the standard variant) or a practical range of 3.9km (for the long range Zigbee variant - 15km is advertised) (Jawad et al., 2017). Maximum nodes are 65,000 which should be ample for even a large farm. Range limitations can be overcome by using repeaters. ZigBee requires line of sight communication which is a limiting factor but has low power requirements. Transmission rate varies from 20 to 250kbs. Zigbee is designed to support multi-hop networks. There are three types of devices in a Zigbee network; FFD (Fully functional device which can also act like a router), RFD (Reduced Functionality device) and a

single coordinator. Routing traffic is in-band with the data.
Zigbee is a potential solution provided that the distance between stations does not exceed the line-of-sight (LOS) communication range. The main advantage of ZigBee is that it can use bursts which enables the rapid transmission of data however, it is a short-range protocol, and an intermediary device would be needed to communicate data back to the farmer. Zigbee is a possible solution for communication between cattle and to a very local transmission site but other devices will be needed to transmit data back to the farmer.

Dash7 (*DASH7 Alliance – An Open Specification*, n.d.) arose from military RFID use and primarily utilises the 433MHz band which gives it multi-km range. To improve performance, some network devices support the 868 and 915MHz unlicensed ranges as well. Bandwidth is either 25kHz or 200kHz which gives transfer rates of 9.6kbit/s, 55.55kbit/s or 166.7kbit/s. Max packet size is 256 bytes. (Arsalan et al., 2018). There are no retail providers of Dash7 devices in New Zealand, so this technology is not recommended.

Wi-Fi HaLow is based on the IEEE 802.11ah standard (Shanmuga Sundaram, 2016) and consumes less power than standard Wi-Fi with a longer range; up to 750m using the “unlicensed” frequencies from 915-928MHz. Being a superset of Wi-Fi, Wi-Fi HaLow supports IP based communication and is designed to support up to 8191 simultaneous clients (as thirteen bits are used for an ID) in a star-shaped network with data transmission rates from 150kbs to 347Mbs. As such, Wi-Fi HaLow is only possible for cattle monitoring. Up to 6960 stations can be simultaneously connected over 1km while transmitting 100 bytes of data every 60 seconds without packet loss when tuning TIM (12 groups) and RAW (2 slots) parameters. In terms of power efficiency, 802.11ah enables 500 IoT devices to turn on their radios 3% of the time when transmitting every 60 seconds (Šljivo et al., 2018). Wi-Fi HaLow is included here despite there being no retail supply of network cards in New Zealand as it has distinct advantages that may make it worthwhile to source parts from overseas. One of these advantages is the multi-hop relay support integrated with the technology.

Bluetooth V5 was released as an enhancement for IoT devices. The theoretical range (240m@125kbps) is 2 times that of Bluetooth Low Energy (mentioned below) but requires significantly more rf output power (from 10mW to 100mW) (Team, 2019). The Bluetooth V5 protocol stack supports IP based communication natively which is an advantage in designing a system. There are four data transfer rates of 2Mbps, 1Mbps, 500kbps and 125kbps, with increased range at each reduction of rate.
For both power usage and connection speed advantages, Bluetooth is attractive. Bluetooth 5 also includes mesh networking, making this a significant challenger to ZigBee.

RFID is short range only (less than 2m) but has extremely low power usage (Duroc, 2022; Jia et al., 2012; Landaluce et al., 2020). RFID tags are required for cattle identification in NZ (NZ Government, 2018) so these tags will be present, regardless of any other technologies chosen (Williams et al., 2019).

LoRaWAN can be used as a short haul system but the other systems are more cost-effective, albeit with reduced range.

Table 1.3 Short range communication protocols compared

Parameter	RFID	ZigBee Default	ZigBee Long Range	Bluetooth Low Energy	Bluetooth Version5	LoRaWAN	Wi-Fi HaLow
Standard	RFID	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.4g	IEEE 802.11ah
Frequency	433 Mhz	2.4 GHz	915 MHz	2.4 GHz	2.4 GHz	915 MHz	915-930 MHz
Range	2m	60 m	14km / 6.5km	10m	750m	5km	1km
Data rate	n/a	250 kbs	10 / 200 kbs	1 Mbps	125kbps-2 Mbps	50 kbps	150 kbs (1km range) – 80Mbps
Power use		36.9 mW	<250mW	10 mW	10 mW	125 mW	11mW
Network size		65,000	65,000	App-defined	App-defined	10,000	8,191 per AP
Network topologies		P2p, Tree, Star, Mesh	P2p, Tree, Star, Mesh	Star	Star	Star of stars	Star and Relays
Native IP		No	No	NIC-Specific	NIC-Specific	No	Yes
Limitations		Line of sight	Power usage	Short range	No mesh support	Data rate, capacity	Limited devices
Farm use ideals		Not ideal as line of sight.	For small datasets only	Not ideal as range is short	Only useful for animal to collector site use	Not ideal for short-range but is good for long-range.	Ideal but imported cards may be expensive and not supported in NZ

The choice of short range RF solution for an alpine farm will depend on the amount of data transmitted and the number of animals. Wifi HaLow is an interesting option but it's cost and lack of support (at present) makes recommending Bluetooth V5 an easy choice.

Table 1.4 Communication protocol options available on a range of microcomputers

Comms option	Pi V4	Rock Pro64	Jetson Nano	Tinker EdgeT	Arduino MegaR3	Microbit	ESP32
GPS	M	M	M	M	M	O	O
802.11ah	O	O	O	O	O	O	O
BT 5	I	M	M	M	M	X	O
BT BLE	M	I	M	I	M	X	I
ZigBee Std	M	M	M	M	M	O	O
ZigBee LR	M	M	M	M	M	O	O
433MHz	M	M	M	M	M	O	O
LoRaWAN	M	M	M	M	M	M	M
NbIOT	M	M	M	O	M	O	O
Cat-M1	M	M	M	O	M	O	O
LTE-M	M	M	M	O	M	O	O

Key: I=Inbuilt, M=Module available in NZ, O=Module available from Overseas and X=Not Suitable

If reliable frequent communication is not possible extra storage may need to be provisioned on the sensor device to act as a data buffer, raising the costs and lowering the battery life of each sensor unit.

1.1.2.2 Data Mule

To avoid the monthly costs and infrastructure investments in RF backhaul technology, an autonomous travelling device (rather than the actual farmer) could visit the herd, download the data, and upload this data on return. This concept is called a “data mule” as the device is purely a carrier for data in a similar manner as the (humorous) ethernet over avian carrier protocol (Waitzman, 1990). At the time of writing, this approach would incur significant range limitations due to current battery technology and requires efficient pathfinding requirements of the device, along with the requirement that the device distinguish between animals and other obstacles. The device would need to navigate close enough to each animal to read the passively powered RFID sensor information without being trampled on and/or startling the animal and obtain data from the other sensors on the animal. Most research on data mules in a rural context has concentrated on the use of unattended aerial vehicles (UAV or “drone”) as these significantly reduce the issues involved in efficiently traversing terrain and avoiding obstacles between the various sensor locations. This approach is not feasible at present in New Zealand as the current civil aviation laws in the country prohibit autonomous operation and require an operator to have line-of-sight visibility of a drone at all times unless an expensive license is obtained. (Aviation Authority of New Zealand, 2015)

An unmanned ground vehicle (UGV or “rover”) could be utilised to visit each animal however these face issues of more difficult route planning and obstacle avoidance concerns. As such, using a rover data mule to visit individual animals is not addressed in this project.

1.1.2.3 Hybrid of RF Short-Range and Data Mule solution

Beef cattle tend to revisit feed and water lots (Johnstone-Wallace & Kennedy, 1944; Martina et al., 2015) so a hybrid of the RF short-range option and a rover-based data mule could be implemented where cattle visit a solar-powered data-aggregation feed/water site and transmit their data by one of the short range RF options (Bluetooth version 5 for example) (Jawad et al., 2017) and then, a rover data mule can be used to deliver the data from the aggregation point to the farmer at regular intervals. An advantage of this method is that photographs of animals can be taken when they visit the site to enable a visual inspection of animal health and scales to measure animal weight could also be implemented. Each data-collection site's short-range RF infrastructure will need to be repeated for every feed/water site where observations are required. The data will need to be stored for as long as it takes for a communications link to transfer the data or for a data mule to visit the site. It is envisaged that a maximum period of at least once every 24 hours would provide timely data.

1.1.3 Utilising a rover as a data mule

The main difficulties in using a rover as a data mule are ensuring reliability, effective range/battery life, efficient sensor data capture/storage, efficient route planning and obstacle avoidance (Borges De Sousa & Andrade Gonçalves, 2010; Chemhengcharoen et al., 2019; Manderson & Hunt, 2013; Petterson, 2020).

Reliability (Nguyen-Huu & Titus, 2009), Battery life (Hall, 2021; Zogopoulos, 2021) and Data storage methodologies and systems are improving all the time so the current limitations of these will not be addressed as it is assumed they will be alleviated in the near future.

Route planning and obstacle avoidance are somewhat related as an efficient route planning system must also address any potential obstacles.

The route planning problem is generally treated as an NP-Neighbourhood solution (Ab Wahab et al., 2015; Koenig et al., 2004; Sugihara & Gupta, 2011) however the presence of potential obstacles requires that the route-planning system adjust a planned route dynamically to account for a new localised route to avoid the obstacle and continue (Masehian & Katebi, 2014).

To avoid obstacles the rover must first be able to detect them, and the following section outlines detection sensors and their limitations.

1.1.4 Obstacle Avoidance

As a rover traverses the terrain, it is essential that it avoids objects that could cause damage to itself and/or vegetation. Animal (including human) object avoidance is paramount. Despite considerable research into rover obstacle avoidance (Ahmed et al., 2021; Cho et al., 2019; De Simone et al., 2018a; Irsyad Bin Sahalan, 2018; Katiyar & Dutta, n.d., 2019; KISHORE et al., 2018; Krotkov et al., n.d.; Kwaśniewski & Gosiewski, 2018; W. Li et al., 2019; Mannar et al., 2018; Masehian & Katebi, 2014; Moeller, 2020; Mohanty & Dewang, 2021; Moravec, n.d.; Parhi & Mohanty, 2016; Simmons et al., n.d.; Tang et al., 2015; Yibin et al., 2006), there has been surprisingly little research undertaken on obstacle avoidance in an agricultural context with its unique challenges (Blanche et al., n.d.; Chemhengcharoen et al., 2019; Hoffman et al., 1999; Lecun et al., n.d.; Schäfer et al., 2005; Spenko et al., 2006).

Obstacle Types

Obstacles can be defined as either static (fixed location) or dynamic (mobile).
Many obstacle-avoidance systems treat all objects as dynamic objects, a safe default, but this approach requires significant processing power and prevents a rover from passing these objects closer than it might otherwise, reducing rover range.
Where systems do distinguish between static and dynamic obstacles, this is often determined when these are scanned ("is the object moving right now?") which offers little processing benefit over the purely dynamic approach. Static objects can be predefined before the rover ventures on its run when an aerial map is loaded to form an initial route, but a map may not identify all objects.
Classifying obstacles via a static map loaded at initialisation on site and classifying obstacles and updating this map as the robot passes each object would provide the best efficiencies.

To avoid an object we need to know what types of obstacle are expected and how to identify them. The nature of rural obstacles will depend on the topology of the farm, but in New Zealand, will likely comprise of:
1. Various vegetation types of various lengths, some of which may be taller than the rover, along with trees and shrubs at various stages of growth.
2. Various soil types, ranging from very sandy to extremely clay-heavy, which retain water differently, so the rover must be able to distinguish between dry or boggy conditions and ideally differentiate between deep and low-lying pools of water.
3. Rocks and stones, which are numerous on NZ farms and the rover must decide to either climb or circumnavigate these obstacles.
4. Various weather conditions - sun, snow, rain, ice, and fog.

5. The terrain gradient which needs to be traversable or avoided.

6. Rabbit warren holes and/or other large depressions which need to be avoided.

7. Farm fencing, where the rover must either stay within the defined boundary and/or detect and utilise methods to pass through fences, gates, cattle-stops, etc.

8. Humans, animals, and any farm equipment, which must be avoided wherever they are.

Typical Obstacle-detection sensors

Sensor classification falls broadly under two functional axes: proprioceptive/exteroceptive and passive/active (Siegwart et al., 2011).

Proprioceptive sensors detect the rover's state such as position, orientation and speed via velocity sensors (W. H. Zhu & Lamarche, 2007), inclinometers/gyroscopes (Dai et al., 1996) , position sensors (Chao et al., 2013), heading sensors or accelerometers (Beliveau et al., 1999), etc.

Exteroceptive sensors collect information from the environment around the rover via such sensors as time-of-flight (Foix et al., 2011), lidar (Yan et al., 2015), laser (Suh, 2019), sonar (Choit et al., 2005), microwave radar (C. Li et al., 2017), and cameras (Bernini et al., 2014).

The other axis delineates between passive and active sensors.

A passive sensor such as a CCD camera (Fossum & Hondongwa, 2014) or thermal camera (Akagawa, 1996) receives environmental information energy whereas an active sensor generates and emits energy and measures the response. Passive sensors use less power but often require more processing time to process resulting datasets.

Active sensors tend to require less processing time and have further range but require power to emit the signals and can be influenced by other sources of similar energy.

Figure 1.1 Obstacle detecting Sensor Types (Image from author).

Radio detection and ranging (RADAR) is an active system often utilised on full-size vehicles as it has a long range and can cope with dusty conditions. The system works by emitting radio signals and timing how long it takes for the signal to return after being bounced off an obstacle. Rain and snow can cause attenuation, but in a large vehicle, power may be boosted to compensate for this. The data returned is of high resolution (normally a point), but categorisation of an object is not possible without further processing.

Forward looking ultra-wideband (UWB) RADAR can penetrate grasses (Wong et al., 2003) to determine rocks hidden behind tall grass, etc. but developing a data model for the reflected nature of holes such as rabbit warrens may be indistinguishable from background noise and may not measure far enough ahead of the rover for the rover to be able to avoid the obstacle. This is an aspect of object avoidance that does not appear to be well-researched, besides Wong's work, but is a project more suited to an RF engineer to design the required antennae array and is out of scope for this project.

Sonic navigation and ranging (SONAR) (typically an active ultrasonic sensor) works like RADAR but the system emits sound rather than radio waves. SONAR suffers from environmental constraints such as wind, temperature and humidity which effect the measurement accuracy (Mohammed et al., 2020). Ultrasonic sensors are also limited to the speed of sound (the signal attenuates heavily in air) and thus have limitations on their range and sensing rate. The wavelength is approximately 4mm and so can miss narrow surfaces, such as a narrow bush branch or a fence wire. As SONAR uses reflection, like RADAR, any surfaces at oblique angles (soft edges) to the rover may not reflect back (Mubarak, 2013) and reflections from multiple objects may also confuse a sensor. Vegetation tends to confuse a sonar system as well, particularly if the grass is blowing in a wind. The wide beam angle reduces directional resolution (overlapping ultrasonic sensors have been used to reduce this limit). (Kleeman & Kuc, 2008). SONAR's main advantages are price and that, being sound based, it is immune to visibility issues such as dust, fog and night.

Light detection and ranging (LIDAR) has been used effectively in many rover deployments and is the method of choice for many household vacuum cleaner robots (Mohammed et al., 2020). LIDAR is an active light (light amplification by stimulated emission of radiation or "laser") reflection system and is less affected by background solar radiation or night but is affected by dust, fog, rain, ice and snow (Formsma et al., 2010). 360degree LIDARs are a cheap form of object detection, enabling edge detection of an object, but only scan in a single dimension, missing objects higher or lower than the narrow scan angle. Scanning vertically while a motor rotates the lidar unit provides a complete viewpoint but reduces scanning time, limiting predictive ability. Wind effects on grass may also

confuse the sensor. Maximum range is about 80m with an accuracy of +/- 50mm over a range of 20m (Crane et al., 2006) but consumer-level LIDAR units tend to have a range of 12m (Adafruit, 2022).

Far-infra-red (FIR) passive camera sensors have long range and are immune to most environmental factors such as rain, snow, etc. and provide improved resolution at night. They suffer from inflated cost and the high computer processing power required to analyse the visual data, along with low resolution (compared to a conventional camera), and only produce grey-scale images. Distinguishing objects in cold environments may be difficult (Mohammed et al., 2020). Matthies and Rankin (L. Matthies & Rankin, 2003) found that thermal signatures are effective at determining depressions and other holes in the ground, although their approach worked best at night and would need to take solar radiation into account for daylight use. Other Infra-red systems (typically active systems) are more effected by environmental conditions, especially solar radiation from the sun.

FIR passive camera systems can be used as rangefinders (Druzhkov & Kustikova, 2016; Gao et al., 2016; S. Huang et al., 2020; Mohammed et al., 2020; Moravec, n.d.; Rajavarshini et al., 2021; Schäfer et al., 2005). Backlighting and lens flare are the main issues with using cameras besides their limitation of daylight-only use, without additional lighting.

Monocular cameras use the size and shape difference between successive image frames to determine depth and can be confused by areas of sharp contrast, especially shade in a bright sun (Michels, 2005).
Stereo cameras can have better noise immunity and these are easier to derive range information from (via triangulation from the two cameras) but processing two sets of images requires more computing power.
Schäfer, etc. implemented a “depth discontinuity” method to determine obstacles above and below ground level using a stereo camera, recognising that traditional ground level references do not apply in a typical outdoor topology (Schäfer et al., 2005). The weakness of this approach is that the camera needed to be angled towards the ground, and thus there is a contradictory relationship between angle, the size of the obstacle that can be determined, and rover speed, which was not explored in their work. It also means that the camera may have limited forward vision, depending on the angle, which may require additional sensors to address.
Sharma and Shah used image processing to determine if animals were present with an 82.5% accuracy (S. U. Sharma & Shah, 2017) however this required significant processing resources (Corei5) to produce 10 fps images.

Camera-based detection of holes in the ground (rabbit warrens or natural depressions) has been briefly explored (Ghaffari et al., 2004; Kusuma Arbawa et al., 2021; L. Matthies & Rankin, 2003; J. Wang et al., 2016) by using thermal signatures (infra-red) which works well at night provided that ground cover is not too dense but has limited effectiveness on a sunny day (L. Matthies & Rankin, 2003). Matthies and Rankin propose modelling solar illumination for their approach to be usable in daylight conditions.

Determining and adapting to ground conditions has primarily been limited to detecting traction loss on the driven wheels and compensating by pulsing torque in an analogous manner to anti-lock braking. The methodology is well developed and is the approach taken by full-size vehicles. Determining ground conditions before the rover reaches them has had less research. Khan and Ahmed (Khan & Ahmed, 2021) used a CNN (convolutional neural network) to detect snow on road images and Kawai (Kawai et al., 2014) used a car-mounted webcam to distinguish road conditions at night via colour differences but a road is a known surface and rural topologies vary considerably. Wading sensors as utilised by Tran (Tran et al., 2015) only face downwards and can't predict the depth of water ahead of a vehicle.

Compensating for ground elevation and slope has been treated extensively in both research and general industry, with a gyroscope being the most-used sensor used to detect the current gradient the vehicle is travelling on. Predicting upcoming gradients has had less research. A stereo camera techniques such as Schäfer's (Schäfer et al., 2005) holds some promise for holes or negative gradients but their technique is not as applicable to positive gradients.

The standard post and wire fence used in NZ farming is difficult to detect as the diameter of the wire is less than the detection scan resolution of many sensors. An ultrasonic sensor requires an object of at least 4mm in width, for instance. In this regard a LIDAR sensor ought to be better but exceedingly small objects may be missed. RADAR would be the best possible sensor to detect this but a RADAR system requires considerable power resources.

If the fence needs to be traversed (a common occurrence on a large farm), determining a suitable place to cross is somewhat difficult, especially where long grasses may cover a tunnel and/or a gap in the bottom wire.

Wind effects acting on objects can make object identification difficult for any sensor. Camera-based object identification when facing into the sun or other lighting challenges are also likely to be issues in a real-world environment.

Sensor fusion approaches

As a rover should operate in all weathers and conditions, multiple sensors are likely to be required to compensate for the deficiencies in each sensor type. Multiple sensor calibration has been investigated thoroughly at a basic level of integrating two sensors (Rodriguez F. et al., 2008; Zhou et al., 2018) but integration of more than two sensors appears to be lightly touched in research.

Manduchi (Manduchi et al., 2005) implemented a combination of a colour-stereo camera to categorise grass, bark, soil and rocks, holes, etc. along with analysis of a single-axis lidar to detect rocks partially hidden by grass. Manduchi found soil/dry grass categorisations was difficult with pure colour matching and they suggested additional techniques such as visual texture mapping or multispectral thermal analysis. (Castano et al., 2014; Gilmore et al., n.d.)

Table 1.5 Single obstacle-avoiding Sensor Summary

Sensor	Radar	Ultrasonic	Lidar	IR	FIR	Stereo Camera	Single Camera
Type	Active	Active	Active	Passive	Active	Passive	Passive
Cost	High	Low	Med	Med	Med	High	Med
Range (m)	<250	<10	<50	<12	<50	<10	<100
Precision	Low	Low range	V. High	V. High	V. High	Med	Low
Resolution	Low	Low	Precise	V. High	V. High	Med	Med
Reliability	High	Low	High	High	High	Med	Med
Power	High	Med-High	Med-High	Low	Med	Low	Low
Processing	Fast	Speed of sound limits	Fast	Med	Fast	Fast CPU needed	Fast CPU needed
Rain influences	No but reduces range	Yes	Yes	Yes	No but reduces range	Yes	Yes
Dust influences	No	No	Yes	Yes	No but reduces range	Yes	Yes
Fog influences	No	No	Yes	Reduces range	Reduces range	Yes	Yes
Sound influences	No	If High pitch	No	No	No	No	No
Light influences	No	No	No	Direct sun	Direct sun	Lens flare	Lens flare
Temp influences	No	Yes	No	Can reduce contrast	Can reduce contrast	No	No
Other influences		Echoes Wind	Snow	Snow		Wind	Wind
Classifies object	No	No	In single plane	Yes	Yes	Yes	Yes
Light required	No	No	No	No	No	Yes	Yes
Single point Reliability	No	No	-	-	-	-	Static objects only
Research	(Blanche et al., n.d.; J. Huang et al., 2001; Mohammed et al., 2020; Norouzian et al., 2019; Schneider & Wenger,	(De Simone et al., 2018b; Jiménez et al., 2014; Kapoor et al., 2018; Kleeman & Kuc, 2008; Mohamme	(Byeon & Yoon, 2020; Goodin et al., 2019; Lebakul a et al., 2021; Lu et	(L. Matthies & Rankin, 2003; Ren et al., 2020; Sulaimo n Alli et al., 2018)	(Dwork et al., 2006; L. Matthies & Rankin, 2003)	(M. K. Cheong et al., 2016; Gao et al., 2016; Hautì et al., 2006;	(Chaudh ary et al., 2019; Hautì et al., 2006; Hoffma n et al., 1999;

	2003; Yamauchi, 2008)	d et al., 2020; Rosique et al., 2019; Shing et al., 2008; Sulaimon Alli et al., 2018)	al., 2020; Mohammed et al., 2020; Wallace et al., 2020; J. Wang et al., 2016; Yamauchi, 2006, 2008; Zhong et al., 2020)			Karuppuswamy, 2000; Lecun et al., n.d.; Lwowski et al., 2014; Manducchi et al., 2005; Mannar et al., 2018; Michels, 2005; Nguyen Viet & Marshall, n.d.; Noori et al., n.d.; Odeh & Faqeh, 2009; Schäfer et al., 2005; P. S. Sharma & Chitaliya, 2007; Simmons et al., n.d.; Sun et al., n.d.; van Hecke et al., 2018; S. Wang et al., 2021; Yamauchi, 2008; Z. Zhang, 2012)	Hoffmann et al., 2004; Karuppuswamy, 2000; Lecun et al., n.d.; Lwowski et al., 2014; Manducchi et al., 2005; Mannar et al., 2018; Michels, 2005; Nguyen Viet & Marshall, n.d.; Sun et al., n.d.; van Hecke et al., 2018; S. Wang et al., 2021; J. Zhang et al., 2017; Z. Zhang, 2012)

A very high-resolution camera/s is the most suitable solution, supplemented with a non-visual sensor such as SONAR or RADAR, however these will come with their own benefits and limitations that need to be mitigated.

Camera-based obstacle-recognition solutions.

Cameras are often used in industry to perform obstacle detection as they provide many advantages. To provide obstacle detection, a video camera can be used, as differences between successive image frames can be analysed to determine if a recognised object is getting closer or not.

A single colour video image is a matrix of pixels (a single piece of information in the image), with each pixel having a combination of three luminance values, each measuring the luminance value of a particular colour. Pixels are normally manufactured in red, green, and blue variants (RGB) and these systems are discussed in this report. An individual image can be represented as a matrix of values as per Table 1.5. The resolution of an image depends on how many pixels are in the image and the size of the number storing the luminance values.

Table 1.6 Simple representation of a single video image.

	Horizontal Pixel 1	Horizontal Pixel 2 to (n-1)	Horizontal Pixel n
Vertical Pixel 1	Red, green, and blue luminance values	Red, green, and blue luminance values	Red, green, and blue luminance values
Vertical Pixel 2 to (n-1)	Red, green, and blue luminance values	Red, green, and blue luminance values	Red, green, and blue luminance values
Vertical Pixel n	Red, green, and blue luminance values	Red, green, and blue luminance values	Red, green, and blue luminance values

Obstacle identification is based on edge detection of an image – determining objects by recognising the boundaries of the object. "Canny" (extension of "Sobel") Edge Detection (Ganesan & Sajiv, 2018) and "Fuzzy" edge detection (Haq et al., 2015) (for images with a high noise floor) are the standard industry-utilised methods of edge detection.

The Matched Filter extension to the Canny approach (Ofir et al., n.d.) is interesting as the algorithm is designed for soft (curved) edges (more likely to be present in a natural environment) but the curved edges evaluated were letters and numbers in a 2D image rather than a receding three-dimensional image plane. Run times for their optimised algorithm (using sampling) were 0.6 seconds using C++ on an I7 with 16Gb of Ram with an image of 129x129 pixels). For a 257x257 image, run time was 5 seconds which is too slow for real-time use on a rover.

Utilising wavelet analysis is a very interesting approach, is almost perfect in resolution and would work with a receding plane BUT the authors state the process is not useful in noisy environments (Damlamian & Jaffard, 2019) and the processing requirements of a “Datacube MV200”, 68040 CPU and a “Sparcstation” workstation, as implemented, are too computationally (and economically) expensive for a low cost battery-powered rover.

Utilising a multi-spectral camera system enables detection of soil and vegetation (Larry Matthies et al., 1995) but would involve power and computation usage beyond the rover’s capabilities unless filters could be dynamically applied to a camera.

The main problem of using any form of edge detection algorithm is the quality of the image.
An ideal image has no noise, and perfectly describes the scene it represents. In the real world, images are of a limited pixel and luminance value count and the lens used to capture the light introduces distortions of scale in the image. The image may consist of extremes of light that the imaging system may not perfectly capture (commonly called over and under exposure). Rain, snow, or other environmental conditions may also reduce image quality. These are all well-known problems, and many different solutions exist to reduce their impact on image quality.

One external issue impacting quality that is particularly relevant to a rural rover is image stabilisation.

1.1.5 Image stabilisation

As a video stream is a successive stream of images, the motion of objects around a vehicle can be determined by locating an object (often called a “feature”) in an image frame, and then comparing its location in subsequent frames. (If a feature is static, the same process can be used to determine the velocity and direction of the vehicle). For this approach to work, initial feature detection is required and as the vehicle passes this feature, new feature/s will need to be acquired and used for comparison. The vehicle’s own movement (including any chassis vibrations) will also need to be accounted for and mitigated against.

What makes this approach more problematic is that, in rural New Zealand, the topology may not be smooth, and so our vehicle (and therefore it’s camera), are susceptible to jostling while travelling, making feature detection difficult. At an excessive level of movement, the feature/s may not be present from frame to frame.

Image stabilisation works by compensating for the movement of a vehicle by recognising that

movement and applying a counter movement in the opposite direction. As such, some image resolution is lost as the resulting usable image is of smaller dimensions than the original and the system will not work if the movement results in an entire frame being out of position compared to the previous frame.

There are two main types of image stabilisation systems; Optical Image Stabilisation (OIS) and Digital Image Stabilisation (DIS) or a hybrid of these two approaches. We will concentrate on the first two methods.

1.1.5.1 Optical image stabilisation (OIS)

One solution to the problem of image stabilisation is to use optical image stabilisation (OIS) where a control system is placed within the camera lens (this being the ideal place as it acts as a force multiplier compared to the sensor location) and/or to the camera sensor. This technique is designed for countering the vibrations caused by hands shaking whilst holding a camera. If the system senses movement to the left (normally via a hall-effect sensor), it will move the lens or sensor to the right to compensate. This approach works well for small movements but limits on the mechanical range of the servomotors (servos) used prevent this technique from compensating large movements and there may be time lags due to inertia of the detection and mechanical components. Another issue with this technique is that the cost to implement OIS in a camera system is high.

An extension to this technique is to use a gimbal – an external system mounted to the camera system comprising of a gyroscope that is controlled by larger servos. This system copes with larger movements than an in-camera system but still suffers from mechanical range and inertia issues and requires a hysteresis system to prevent oscillations if a vehicle is moving fast with large displacements. This method may also prevent the camera system from looking ahead down a path if the vehicle is travelling on an incline or decline.

1.1.5.2 Digital Image stabilisation (DIS)

Digital image stabilisation uses computers to perform the image stabilisation.
Three main methods exist. Feature Mapping and Sensor Fusion or a hybrid of these methods.

1.1.5..1 DIS Method One. Feature Mapping

The first method is to use the same feature-mapping solutions used to determine obstacles but to select a static feature that can be tracked from frame to frame. One problem with this approach is that, as the vehicle is moving, the change in perspective will lead to distortion as shown at (Android Authority, n.d.-b). A potentially greater problem is that the purpose of image stabilisation is to enable feature detection, yet feature detection is required to stabilise the image, so the selection of which features to track is critical to the success of this method.

Feature tracking is also computationally expensive and complex. Feature mapping has limitations on image size and the amount of displacement as increasing both increases processing requirements. Current research has focused on efficiency improvements such as the SIFT (Battiato et al., 2007; Chao et al., 2013) and SURF (Ljubičić et al., 2021; Shene et al., 2016) methods, along with other approaches such as affine-transform matrices (Mai et al., 2012; Mingkhwan & Khawsuk, 2017; Schwertfeger et al., 2011; Shen et al., 2009; Thillainayagi & Senthil Kumar, 2017) which compares separate video frames, particle filters (J. Zhu et al., 2016), linear and curve filters (L. Wang et al., 2012) and iFMI spectral registration (Schwertfeger et al., 2011). Hsu (Hsu et al., 2005) concentrated on hand-held camera shake using an inverse triangle technique while Morimoto (Morimoto & Chellappa, 1998) compared 4 DIS algorithms and found that simpler models performed better than more complex algorithms, due to them being less sensitive to tracking errors. Ljubičić et al. (Ljubičić et al., 2021) outlines freely-available software applications for applying digital image stabilisation. Once the direction and velocity of the camera movement has been determined, mathematical rotational and/or translational matrix operations apply this vector in the opposite direction to counter the effects of the movement.

1.1.5..2 DIS Method 2. Sensor Fusion

The second method is to apply sensor fusion where camera movement can be compensated by an external sensor such as a gyroscope and/or accelerometer. This approach is like the OIS method except that the calculated movement is mitigated by translating the image in the opposite direction and velocity of the movement using software. The same matrix mathematical translations used with feature tracking are then applied. The most common sensor utilised with this approach is using an inertial measurement unit (IMU), a sensor combining a gyroscope and accelerometer together to give 6 degrees of freedom detection (6DOF). Adding a magnetometer gives 9DOF, adding an additional barometer/temperature gauge outputs 10DOF and adding a GPS sensor to that gives 11DOF.

For an inertial management unit (IMU) to compensate for unwanted movement in video streams, a vector of the resulting movement needs to be created so that the pixels in each image frame can be rotated/translated by the inverse of the unwanted movement.

The gyroscope on an IMU outputs data in angular velocity which is a measure of rotation measured in angles of movement per second and is expressed in revolutions, radians or degrees per second per each axis of the three dimensions, x, y and z. An accelerometer measures acceleration in the same three dimensions and outputs data in m/s^2. Magnetometers output magnetic field strength in the same three dimensions and output data in micro tesla (µT).

For a vector to be created, the rotation measured by the gyroscope needs to be converted into angles of roll (how much the vehicle tilts from side to side or rotation around the x axis), pitch (how much the vehicle rises and falls or rotation around the y axis) and yaw (how much the vehicle moves horizontally to the left or right or rotation around the z axis). These angles are commonly called Euler angles. Note that the convention of x, y, and z axes is not strictly defined, and one may place the x, y, and z -axes in any direction that is convenient, providing the angled relationships between them remains fixed, but here, we will use the terms demonstrated in Figure 1.2. A value of 0 means no rotation around an axis and positive values refer to clockwise rotation.

Figure 1.2 Diagram of conventional roll, pitch and yaw angles (Image from Author).

The mathematical process to convert between gyroscope angles to Euler angles incorporates integration which introduces a cumulative error, the size of which depends on the number of samples. Along with this problem is the issue of gyroscope drift, where the gyroscope values deviate over time due to temperature variances, gyroscope tolerances and external disturbances such as the jostling of a moving vehicle. The results of these two errors can be seen in Section 4.

The general process of using an IMU accelerometer to measure Euler angles is expressed in the following mathematical equations, Equations 1.1 to 1.4.

At rest, gravity is the only acceleration acting on the IMU, and assuming it is mounted so the z axis points straight down, the acceleration matrix is:

$$a_{level} = [0 \quad 0 \quad g]$$

Equation 1.1 Acceleration at rest where g=9.81m/s^2.

The acceleration in a tilted frame is

$$a_{tilted} = [\, a_x \quad a_y \quad a_z \,]$$

Equation 1.2 Acceleration on a tilted frame

where ai refers to the acceleration values for each axis.

Euler angles are generally used (in particular, the 3-2-1 set) referred to as yaw, pitch, and roll although quaternions are often preferred as these do not present multiple readings for the same position (a product of using the inverse sine and cosine functions to generate Euler angles).

Using a direction cosine matrix, the relationship between the tilted frame and level frame is:

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}_{tilted} = \begin{pmatrix} \cos\emptyset & 0 & -\sin\emptyset \\ \sin\theta\sin\emptyset & \cos\emptyset & \cos\theta\sin\emptyset \\ \sin\theta\cos\emptyset & -\sin\emptyset & \cos\theta\cos\emptyset \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix}_{level}$$

Equation 1.3 The mathematical relationship between level and tilted frames.

Solving for pitch and roll, we get:

$$Pitch = \theta = arcsin \frac{a_x}{\sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}}$$

$$Roll = \emptyset = arctan \frac{a_y}{a_z}$$

Equation 1.4 Mathematical equations for pitch and roll

Using an accelerometer in this manner is subject to high levels of noise, as amongst other errors, any movement of the vehicle will corrupt the measured value of gravitational acceleration.

Magnetometers can give an accurate measurement of the gravitational field but are influenced by other sources of magnetism, and in this case, as the camera and front IMUs are mounted close to an

electric motor on a steel chassis, the magnetometer readings are not initially utilised, but are utilised in Section 4.5 to confirm this assumption.

To measure orientation using an accelerometer, an initial orientation must be known and then measurement values are integrated over time (as acceleration is differentiation of speed). Integration of any errors will lead to increasing error components in the calculated value.

Once the roll and pitch values are known, a rotation matrix can apply these angles to our video frame data to transform the moving frame data to a purely horizontal reference as described by Odelga (Odelga et al., 2017).

This approach has been explored previously. Karpenko used the gyroscope on an iPhone to stabilise videos by iteratively calculating the differences between frames rather than directly using immediate orientation values (Karpenko et al., 2011). Jiang used an IMU to adjust both OIS (in strict x=0 and y=0 directions) and DIS stabilisation algorithms for the z-axis (Jiang et al., 2010) but did not provide real-world results.

Stegagno et al used integrated IMU and Dense visual odometry pose estimation using a red-green-blue-depth (RGB-D) camera (requiring indoor operation) and pan-scanned the drone to increase field of view for a haptic-tele-operational drone. They proposed using a stereo camera for outdoor operation. Their experiment constrained the drone to 3 degrees of freedom, where roll is present in this project, requiring 6DOF.

Wiriyaprasat and Ruchanurucks (Wiriyaprasat & Ruchanurucks, 2015) used a full attitude and heading reference system (AHRS) unit (this has an onboard processor to provide attitude and heading rather than outputting raw gyroscope and accelerometer values as an IMU does) with Iterative Least Squared Error (ILSE) methods to modify a homograph matrix to accommodate more rotational differences than a normal homograph matrix. Calibration of the sensor to the camera was required as these were not installed in the same plane. A constant was required to change the pitch angle which they could not explain.

Odelga's team used both gyroscopes and accelerometers of the IMU on a drone and used the accelerometer values to compensate for the gyroscope drift, using a complementary filter. They also implemented fish-eye RGB-D lenses to get an improved field of view and compensated for the distortion introduced by this lens (Odelga et al., 2017).

All the approaches listed above only considered single IMU units.

1.1.5..3 DIS Method 3. Hybrid of Feature mapping with sensor fusion.

The third method combines feature detection with IMU data to improve accuracy and speed of a feature detection system like Ryu and Auysakul (Auysakul et al., 2018; Ryu et al., 2010) who implemented an IMU to assist with a KLT tracker algorithm. Auysakul later used the IMU to drive servo motors as an OIS technique (Auysakul et al., 2019). The same issues with a feature system still apply with this method but accuracy and speed are increased. Chang et al (Chang et al., 2016) extended the use of an IMU by utilising a TLD (Tracking, Learning, Detection) multi-layer classification algorithm coupled with mechanical servos to compensate for the camera movement using OIS techniques. The main benefit of the TLD algorithm is the fuzzy logic controller algorithm dynamically updates the feature points. Utilising servos is only applicable if the motive movement and jostling are slow enough for the servos to compensate, and the servos have sufficient range of motion, both of which did not seem practical when implemented on a rover. As discussed earlier, feature detection is a computationally expensive process for the typically limited processing resources available on a rover.

The common issue with all stabilisation systems is that a portion of the image will need to be cropped to act as a buffer for varying movement, the effects of which can be seen at (Android Authority, n.d.-a). If these effects are to be avoided, further cropping of the image is required, reducing quality. A direct approach to reduce the impacts of large jostling movements is to utilise a wide point of view lens such as a “fisheye” so a feature will stay in the frame even with large movements however this introduces significant lens distortion which will need correction in software.

Figure 1.3 Fisheye Lens capture example (Fpatrocini, n.d.).

The fish-eye lens solution is a workable solution for a fast-moving rover and is recommended for use in a production system but will not be evaluated in this experiment for simplicities sake.

Utilising multiple IMUs to compensate for errors in a single IMU has been investigated (Madgwick et al., 2013) but Madgwick only utilised accelerometers in the array.

It is possible that a Neural Network could perform this form of regression task of both single and multiple IMUs.
The neural network or artificial neural network (ANN) or multilayer perceptron (MLP) has come back into fashion in recent years due to its use in machine learning applications. Originally designed to mimic the biological brain, these supervised machine learning models can predict an output from given inputs and are optimised for classification (discrete outputs – typically a category of some sort) and/or regression (usually continuous outputs) tasks.

A neural network is a collection of interconnected neurons that incrementally learn from data to capture both linear and non-linear trends to provide predictions for new situations containing even noisy and partial information.

The basic unit of a neural network is the neuron. A neuron applies a weighted average function across input data and then applies a nonlinear “input/output” function as shown in Figure 2.2

Figure 1.4 Neuron construction (unit part of the diagram). Image from (Samarasinghe, 2006).

One of the inputs is usually given a value of 1 and called a bias input. This input is used to account for output effects that are not accounted for by the defined inputs.

Training data is used to train a supervised neural network and must consist of input values and associated correct output values. Training involves iteratively randomly changing the value of the weights until the system learns to perform the task properly (the system’s output matches the training data output for the given training data).
The differences in output between the training data and the neural network’s outputs during training (or error) is determined by either simple subtraction for simple networks or a least-squared-error approach for multilayer networks. As the least-squared-error function is parabolic, the network calculates the error gradient in a downward direction along the curve to determine an optimal solution. A process called backpropagation is used to determine the error contribution of each weight, and therefore, how much each weight should change in the next iteration of the learning process.

The overall function of a neural network is determined by the network structure, the connection strengths between neurons (called a weight) and what activation functions are used at each neuron.

Some examples of the type of functions and their use is show in Figure 2.3

Figure 1.5 Example functionality of neural networks for scientific data modelling: (a) fitting models to data or regression, (b) complex classification tasks, (c) discovering clusters in data, and (d) time-series forecasting. Image from (Samarasinghe, 2006)

Examples of the types of network topologies and their indicative uses are shown in Figure 2.4

Figure 1.6 Some neural network topologies: (a) single-layer perceptron, (b) linear neuron, (c) multilayer perceptron, (d) competitive networks, (e) self-organising feature maps and (f) recurrent networks. Image from (Samarasinghe, 2006)

Multilayer Perceptron (MLP) Neural Networks have an input layer, an output layer and, commonly, one or more “hidden” layers between the input and output layers. These tend to be configured as supervised learning systems where valid responses are included in the training data. The weights of the neurons are adjusted until the difference between the correct output and the neural network output reach an acceptable level, as determined by utilising a least square error method along with determining the negative gradient of the error function.

Other types of neural network such as convolutional neural networks (CNN) exist which are optimised to create feature maps of data which can be used to determine shape and texture of data. A CNN neural network identifies features very well and is ideal for the classification of obstacles. For image stabilisation, the operation is more regressive in nature, so a CNN network implementation is not necessary, and a multilayer perceptron approach is initially selected.
A Recurrent Neural Network (RNN) is a network designed for recognising patterns over time. It uses a unique design compared to conventional neural networks (where weighted inputs are applied to a function which then passes the function result to the next stage), in that outputs from some of the neurons are fed back into the inputs of the same neurons as a feedback mechanism. These networks provide a localised "memory" feature in the feedback-equipped neurons which makes RNN networks excel at relational time-series problem-solving tasks such as language processing.

The types of neuron input/output functions are varied and depend on the purpose of that neuron in the neural network.
Common functions are a threshold function where the output is mapped to either 0 or 1 for use in classifying data.
With competitive learning models, various neurons become active, and the ones that receives the largest input values "win" over the others and are weighted more. Over time the model can be said to become more sensitive to types of input data that they respond best to, and so, over time, various neurons become specialists at specific input types. This type of model is designed for competitive situations and/or self-organising feature maps and is not applicable to this project.

Supervised learning uses a sample of the input/output data as training data and uses the following three basic techniques:
1) forced learning, which is somewhat similar to the competitive learning model
2) Reinforcement learning, where feedback is applied to improve the responses over time.
3) Supervised learning, where the actual error differences between input and output are used to change weight values, based on an error gradient.

An epoch is defined as one pass of all training data input patterns into the network. Many epochs may be (usually are) needed across the input data until the weight optimisation process is complete.

Accuracy of a neural network is defined as the percentage of patterns correctly classified for each classification over the entire dataset. Once a suitable level of accuracy has been achieved, the neural network is ready for live data. There are two main parameters of the network that are examined to ascertain accuracy. MSE is the mean square sum of any differences that appear between the input

and the output of a neural network, compared to the expected training results. The MSE gives a measure of how accurately the network can reproduce a single set of output values from the input values. The Pearson Coefficient, R, is used in a neural network context to measure how well the neural network understands the entire problem and reflects the accuracy of the trained algorithm across all inputs and outputs.

As can be seen, neural networks are not magical and will not solve all problems. They require the practitioner to understand the datasets involved to design and configure a neural network system that will correctly classify/regress incoming data.

1.2 Research Objective

The primary objective of this research is to determine whether multiple inertial measurement units in conjunction with a neural network can improve image stabilisation of a camera on an RC vehicle, compared with a single inertial measurement unit.
A secondary objective is to determine the least number of inertial measurement units required to provide a significant measurable improvement.
A tertiary objective is to determine the most basic neural network architecture required to provide acceptable (above 90%) accuracy, when calculating the Euler angles used to rotate an image.

1.3 Thesis Outline

Chapter 1 (this chapter) provided an overview of the motivation, objectives, and results of the project. Chapter 2 discusses prior research that is relevant to this project, which involves the concept of a data mule, obstacle avoidance approaches, image stabilisation and neural networks. Chapter 3 describes the methods used in the project, including the approach taken and the resources and tools required. Chapter 4 presents the results of the project, and these are discussed, and analysed in Chapter 5. Chapter 6 provides conclusions and suggests paths for further research. Chapter 7 provides a place for Appendices and Chapter 8 contains a list of references.

Chapter 2

Literature Review

This chapter reviews current literature on the application of inertial measurement units (IMUs to image stabilisation problems. It outlines approaches that have been used before, and their benefits and drawbacks. The chapter concludes by identifying research gaps and identifying the research question(s) for this project.

As mentioned in the introduction, an IMU can provide accurate positioning, however using acceleration and magnetic data can introduce noise, and mathematical processing artifacts (integration) and gyroscope data has inherent drift issues. Various methods have been proposed to counter these concerns.

One mitigating factor is to obtain a gyroscope with a low zero-rate offset value to minimise this integration error. Gyroscope selection is critical. (La Rosa et al., n.d.)

Compensating for gyroscope errors with accelerometer (with a gravity vector) and magnetometer (with magnetic north) readings has been explored, as the latter do not depend on prior state knowledge, as a gyroscope does.
The most common algorithms for performing this fusion are complementary filters (Odelga et al., 2017; Yoo et al., 2011), Kalman filters and optimisation filters (Yean et al., 2018).

Complementary filters are computationally efficient and easy to implement, but are subject to noisy and biased data and are susceptible to gyroscopic drift around the z axis and are thus not reliable or robust enough for regular use over time (Madgwick, 2010).
Kalman filters provide very accurate orientation (Higgins, 1975; Marins et al., 2001; Sabatini, 2006) in the presence of large noise values such as persistent acceleration however are computationally expensive and complex (Yuan et al., 2019).
Optimisation filters estimate a vector representing the sensor output at the current orientation and attempt to minimise the difference between predicted and measured results. Optimisation filters have acceptable accuracy with lower computational expense than Kalman filters but can suffer from unpredictable convergence (Fan et al., 2017; Mahony et al., 2005; Yean et al., 2018).
To counter gyroscope drift and bias errors, accelerometer arrays have been proposed (Madgwick et al., 2013) but these lack real world testing and implementation.

A software approach proposed by Odelga et al (Odelga et al., 2017) feeds both gyroscope and accelerometer data from a drone into a complementary filter, with a constant gain value, determined empirically to reduce the drift error without ignoring vehicle acceleration. The use of a complementary filter rather than the more widely implemented Kalman filter reduces processing requirements. Using this approach with a very wide-angle ("fisheye") lens permits a wide viewpoint, allowing for compensation of large movement displacements.
Odelga found that the IMU solution worked well, and utilising "fisheye" cameras with large fields of view reduced the issue of a feature disappearing between subsequent images but introduced barrel distortion which needed to be compensated for.

In 2011 Madgwick et al introduced a gradient-descent algorithm commonly called Madgwick's algorithm (Madgwick et al., 2011), which is more computationally efficient than Kalman filters, and is currently widely used in industry (Fan et al., 2017; Yean et al., 2018), but requires two sequential minimisation steps (first on the magnetometer and then on the accelerometer), which can lead to slow convergence. In Madgwick's algorithm, the calculations of roll and pitch are not decoupled from each other which can lead to unpredictable orientation errors, and a single adjustable parameter makes it difficult to combine accelerometer and gyroscope values (Fan et al., 2017).

In 2020 Madgwick et al extended a complementary filter in quaternion form (Euston et al., 2008) based on Mahoney's work (Mahony et al., 2008) to increase robustness against noise while maintaining low computation cost and predictable convergence efficiency (Madgwick et al., 2020) and made this algorithm open source. (X-IO, 2008). The algorithm decouples roll and pitch data and varies the gain factor from a high value at initialisation to a lower running value to improve convergence time.
There are two variants of this algorithm, one in which magnetometer information is combined (IECF) and the other with only accelerometer and gyroscope values included ($IECF_6$). Fan found that when magnetic disturbances were present, the $IECF_6$ algorithm performed better. Fan also outlined that when including magnetometer data, it was important to provide a single point of reference when using multiple IMU units (Fan et al., 2017). Yean's approach of using a complementary filter to counter gyroscope drift, combined with a Kalman filter / gradient descent algorithm worked well only for slow and controlled ranges of motion, and so, is not suited for a bouncing vehicle (Yean et al., 2018).

Madgwick's $IECF_6$ algorithm is a popular algorithm used to determine Euler Angles as it is open-source and requires minimal processor resources. This filter algorithm uses quaternion representations of angles internally to avoid the singularities caused by Euler angles approaching $\pi/2$

radians. A single beta weighted-gain value acts as a fusion factor in a complementary filter, feeding accelerometer data into the gyroscope data to compensate for gyroscope drift. A gradient-descent algorithm enables performance at low sampling rates but, internally, the lowest sampling rate is recommended to be 10 Hz. The sampling rate of the experimental equipment averages at 8.3 Hz so it is expected that the Madgwick filter may not perform as well as the Kalman.

The standard MATLAB Kalman-based imufilter is also compared with the Neural Network results as this method is the industry standard where processing resources permit (Auysakul et al., 2019; Ryu et al., 2010; Sabatini, 2006).

Many articles discuss the use of neural networks to derive Euler angles but these concentrate on single IMU solutions (Aparna et al., 2020; Lv et al., 2021; Weber et al., 2020; M. Zhang et al., 2021). Neural network solutions typically require more processing resources than complementary filters like Madgwick's or even Kalman filters, but, once trained correctly, perform better over wider environmental conditions. Multiple IMU solutions have been proposed before (see above) but, to the author's knowledge, none have suggested using a neural network across multiple IMUs. The concept of a neural network has been covered in the introduction (Section 1.1.5..3) but the specific neural networks evaluated in this work are detailed below.

Using a back-propagating or Levenberg-Marquardt (LVM) trained supervised multilevel perceptron (MLP) neural network (NN) with multiple IMUs should effectively compensate for drift and other errors present in a single IMU implementation and the MLP model is included for evaluation in this project. Due to the computation requirements of a NN, it makes most sense to apply the NN directly to sensor readings, rather than complementing a filter algorithm, as the NN should be able to determine the relationships. The network model diagram for the MLP NN model is presented in Figure 2.1.

Figure 2.1 10 layer MLP model diagram with six inputs and two outputs.

Other NN types that will be evaluated are a non-linear autoregressive Exogenous (NARX) recurrent neural network (RNN) and Radial Basis Function (RBF) NNs.

As the relationships between accelerometer, gyroscope values and resultant Euler angles are non-linear and time related, a non-linear autoregressive Exogenous RNN (NARX) model could be utilised. This approach has been proven in previous research (Aparna et al., 2020; M. Zhang et al., 2021). The NARX neural network is a type of Recurrent Neural Network (RNN). This network model specialises in processing time-series data and is often used in natural language processing applications. The distinctive feature of RNNs is that they take information from other neurons as inputs, unlike the traditional NN approach where inputs and outputs are independent from each other. The most important parameter is the number of layers as this mitigates issues with small and large gradients. The MATLAB NARXNET neural network model is presented in Figure 2.2.

Figure 2.2 10-layer NARXNET Model diagram with six inputs and two outputs.

A Radial Basis Function (RBF) Neural Network is a network design that works on the premise that an input value that is similar in value to another input variable is likely to produce the same result. Unlike the other networks examined so far, the RBF has only three layers, an input, one hidden layer, and an output layer. The input layer contains as many neurons as input streams. The hidden layer uses as many neurons as required to turn the input data into higher dimensional maps as Cover's theorem states that a non-linear separable problem is highly separable in high dimension space compared to low dimension space. The distance (errors) between the input variables and the target value within these dimensional maps moves the neuron to a position closer to the inputs. The distance between neurons and input variables relates to how much influence that input variable has on the target value and is called a radius or spread value. The output layer is selected for either classification or regression. The main advantages of RBF NNs are training speed and universal approximation abilities.

Figure 2.3 RBF model diagram with six inputs and two outputs.

In Chapter 3, the experiment design and methods are outlined, in Chapter 4, the results from these experiments are detailed and discussed, with analysis of these in Chapter 5, and conclusions reached in Chapter 6.

Chapter 3

Method

In this chapter, the method is outlined. The design is presented first with the implementation setup and calibration details following in Section 3.1 and Section 3.1.1, respectively. Details of the experiments undertaken follow in Section 3.2. All angles are expressed in radians unless otherwise mentioned.

The primary objective is to determine whether multiple inertial measurement units in conjunction with a neural network can improve image stabilisation of a camera, compared with a single inertial measurement unit. The various neural network results will be compared with both a Madgwick complementary filter and a Kalman-based filter – the two most common industry filter implementations for Euler angle conversions from IMU data.

To address this objective, an experiment was designed where multiple inertial management units (IMU) were attached to a rigid board as shown in Figure 3.1. A single IMU in the centre at the front to provide data as a typical single IMU system would. Four further IMUs are placed one at each corner to provide multiple IMUs for comparison against the single central IMU. To provide controlled motion of the IMUs a Universal Robotics UR5 robotic arm is used to manipulate the board to known angles.

Figure 3.1 Baseboard diagram showing components (Image from Author).

A method flowchart is presented In Figure 3.2 which outlines the process followed.

Figure 3.2 Method Flowchart (Image from Author)

The experiment setup and calibration process are described in Section 3.1. The experiments are described in Section 3.2, including the Madgwick and Kalman filter configuration and neural network setup and training.

3.1 Experimental Setup and Calibration

To determine if a neural network with multiple IMUs is more accurate than with a single IMU, a replica of a vehicle chassis is constructed and the mounted IMUs are moved in various angles by a robotic arm (to provide a “ground truth” of measurements). IMU gyroscope, accelerometer (and for one experiment, magnetometer) data is captured and then processed by the two filters and the neural network to gather Euler angle data which is compared against the robot arm movements.

A baseboard chassis is constructed of Signboard (two layers of plastic sandwiching an aluminium panel) to provide rigidity so that all IMUs record the same movements (Figure 3.1).
Mounted to the baseboard is a Raspberry Pi connected to five Sparkfun IMU-20948 hobbyist level IMUs via a Sparkfun TCM9548A I^2C multiplexor as shown in Figure 3.1. (Details of all equipment used in the experiments can be found in Appendices 3 and 4). The TCM9548A multiplexor is required as ICM-20948 IMUs only have two configurable I^2C addresses. For single IMU data collection, the front “central” IMU is used. For “Front IMU” IMU data collection, the front three (“FL”, “FR” and “central”) IMUs are sampled, and all IMUs are sampled under “All IMU” experiments.

A Universal Robotics UR5 robotic arm provides the “ground truth” of correct angles used to verify the filter and neural network results.

Calibration of all devices is discussed in Section 3.1.1.

Data capture is managed by a laptop connected to the both the robotic arm and the Raspberry Pi by an unmanaged one gigabyte ethernet switch. A python script (GetRobotData.py in github) uses the Universal Robotics “ut_rtde” python module to connect to the robot arm over ethernet and poll the arm for position data.

Various movement experiments are performed on the robotic arm and IMU and robot arm data is captured as outlined in the method process of Figure 3.2 with more detail in Table 3.1.

The data for all IMUs is captured even though the initial experiments only utilise the central IMU data to mitigate against artifacts brought in by measuring the IMUs differently when later experiments are undertaken. Outlying data is kept. This emulates real world performance (as removing this data could be problematic to differentiate from vehicle movement in production).

Details of the data processing will follow later in the experiments section (Section 3.2).

3.1.1 Calibration.

This section outlines the calibration of the IMU devices and outlines how the IMU data will be synchronised to the robot arm data. The contents of all scripts and configurations can be found in github (see Appendix 1). Detailed specifications of the equipment can be found in Appendices 3 and 4. Unless specified, IMU calibration and alignment steps discuss the central IMU only, although all IMUs are calibrated in an identical manner.

The UR5 robot arm needed to be calibrated first as the UR5 robot joint angles showed slight errors when manually compared to a 1.5m long spirit level. It is hypothesised that this is due to the visible bow in the wooden table used to mount the robot. These were initially compensated for but meant that the robot sensor data showed slight variations away from true positions. Using tool centre point (TCP) sensors on the tool section of the arm initially showed true values of the pose compared to the spirit level however so the TCP sensors were used first in initial measurements. When robotic movement was performed, it became evident that the TCP sensors calculate positioning based on weight and size of the tool mounted on the arm which introduced some subtle errors on the baseboard, depending on orientation of the board. The actual joint angles of the arm components were used instead to obtain direct angular positions and angles altered in math if the joint angles were required to be mounted at non-zero angles in order to get full range of motion of the chassis.

When operating the robot, it was noticed that there is some tolerance and play in moving repeatedly between the same defined angles. In section 5.1 of the user manual, worst case joint position accuracy is given as 1.15 degrees with a detection time of 100ms. The tool centre point (TCP) sensors have a worst-case accuracy error of 20mm in positioning and 1.15 degrees in orientation. These are worst-case values, however, and the robot operated well within these limits.

The baseboard was screwed directly to the robot arm. The robotic arm with the baseboards mounted in the starting positions for the experiments are shown in Figures 3.3 and 3.4.

Figure 3.3 Robotic Arm position showing mounted baseboard for roll and pitch experiments

Figure 3.4 Robotic Arm position showing mounted baseboard for all other experiments

The IMU's gyroscopes and accelerometers are calibrated using static measurements at increments of π/2 radians on all axes, using recorded gravity as a control measurement (tilting the board by 90 degrees in all directions). The calibrate.py program (see github) is used to undertake this calibration and the resulting offset values added to the imudata.py IMU data-capture script (see github) so that captured data has the calibration offsets applied as data is recorded.

The IMU and robotic arm measurements are manually synchronised.
Alignment is based off the central IMU (CIMU) as this is the IMU used for single IMU experiments. To undertake calibration, when an experiment is performed, measurement data from the central IMU and the robotic arm should be visually investigated to determine when movement is first detected. The two elapsed time positions can be entered into the appropriate MATLAB command script for that experiment and the MATLAB processing steps can then be re-applied to the data to achieve alignment.

It is tempting to just align the data samples with an offset based on sample position rather than time but, as is the nature of general-purpose multitasking operating systems, the data sampling rate will not be consistent and results like Figures 3.5 to 3.10 will be seen. Figure 3.5 shows the result of poor alignment. Figure 3.6 is the result of aligning data over the entire dataset, which is acceptable over the entire dataset, but Figures 3.7-3.10 show the inevitable variations in timing of this approach.

Figure 3.5 Varied movement (CIMU) poor alignment plot (Samples: 1000 to 1300) from 4999.

Figure 3.6 Overall “best-fit” alignment plot (CIMU) showing OK alignment at this level

Figure 3.7 “Best-fit” alignment (CIMU) – enlarged view of samples from 100-300.

Figure 3.8 “Best-fit” alignment (CIMU) – enlarged view of samples from 1000-3000.

Figure 3.9 “Best-fit” alignment (CIMU) – enlarged view of samples from 3000-3300.

Figure 3.10 “Best-fit” alignment (CIMU) – enlarged view of samples from 4500-4999.

Even with the alignment of elapsed time, rather than by the number of samples, the alignment is not perfect as can be seen in Figures 3.11 to 3.13, showing that the two devices do not always keep time in the same way. The misalignment is consistent however, as distinct from using sample alignment, and arises from using the middle IMU (central) as the alignment method (the initial robot arm movement may be recorded on one of the other IMUs), as well as differences in performance between the devices.

Figure 3.11 Overall alignment of IMU and robot arm

Figure 3.12 Alignment of IMU and robot arm from samples 1000-1999

Figure 3.13 Allignment of IMU and robot arm from samples 4000-4999

A sample of the raw central IMU data taken from the stationary vehicle is shown below in Table 3.1. All IMU data was captured but only the central IMU data is displayed in this table. The large value of the z (vertical) axis is a measure of gravitational acceleration. This was measuring an average of 10.22617 m/s^2 instead of the expected 9.80665m/s^2. The maximum variation between sea level to the top of Mt Everest is approximately 0.02m/s^2 so the discrepancy does not relate to altitude. (These measurements are taken at approximately 19m above sea level). Varying the mounting positions of the sensors slightly by tightening the mounting nuts changed these values by significant amounts, demonstrating that sensor positioning is of critical importance for accurate results. Once the sensors were mounted firmly on the chassis board, results showed that some sensors were mounted incorrectly in terms of consistent X and Y directions. The incorrect sensors were remounted, tightened down and readings taken again. Results from this initial calibration can be obtained from Appendix 2. To reduce the impact of outlying values, the 95th percentile values of the data was derived and used to calculate the average value of the data. The first three lines of data are shown in Table 3.1.

Table 3.1 First three lines of stationary gyroscope and accelerometer results (central IMU).

Accel_X average ms/s^2	Accel_Y average ms/s^2	Accel_Z average m/s^2	Gyro_X average degree/sec	Gyro_Y average degree/sec	Gyro_Z average degree/sec
-0.19154	0.11971	10.24718	0.027712	0.006262	-0.01092496
-0.1652	0.043096	10.27831	0.031043	0.004663	-0.01652067
-0.17957	0.050278	10.20888	0.033441	0.001732	-0.0085268

The rationale for calibration is based on the works of Qureshi et al (Qureshi & Golnaraghi, 2017) but utilised a set of setsquares (permitting fixed angles of 0,30,45,60 and 90 degrees) to align the IMU sensors in increments of π/2 radians (90 degrees) to apply appropriate offset and bias values (access to the robot arm was not available at the time).

The gyroscope calibration is easiest to determine so is dealt with first.

At rest, all gyroscope readings should be zero so averaging the deviations from this should suffice for a bias offset value in each axis direction. Table 3.2 lists gyroscope offset values for the central IMU.

Table 3.2 Central IMU gyroscope calibration values.

Axis	Flat, Facing Up	Flat, Facing Down	90deg, tilting left	90deg, tilting right	Tilting Up	Tilting Down	Average Offset
Gyro_X	0.029521	0.029307	0.028777	-0.02387	0.028599	-0.02385	0.029078
Gyro_Y	0.004314	0.004279	0.018543	0.018462	0.003848	0.018154	0.004103
Gyro_Z	-0.01118137	-0.0112	-8.4E-05	0.000296	-0.01144	0.000378	0.00743

Table 3.3 lists the Accelerometer offsets for the central IMU. These were obtained from the axes shown in the table and the MATLAB polyfit function (MATLAB, 2023) was used to determine the first order equation parameters of slope and offset. X values used were -1, 0 and 1.

Table 3.3 Central IMU accelerometer slope and offset table calibration values.

Axis	Flat, Facing Down (gravity = -1)	90deg, tilting left (gravity = 0)	Flat, Facing Up (gravity=1)	Slope	Offset
Accel_X	0.02984	9.720203	-0.20886	0.0013	0.0315
Accel_Y	-0.10395	-0.59628	0.032943	-2.338e-4	0.0325
Accel_Z	-9.4977	0.368226	10.22617	-1.0244e-4	0.0325

The offset and scaling values from Tables 3.2 and 3.3 are applied to each IMU measurement to reduce calibration errors. The values are entered into the imucapture.py script (see github).

Variation in the output matters as it may mask any differences in technique. Note that there is considerable variation in the output for a completely stationary vehicle, indicating high noise floor of the sensors. The Madgwick, Kalman and Neural Network processes are meant to reduce these errors, providing they are gaussian in nature.

With the robotic arm in the neutral position as shown in Figure 3.3, a sample of static results from the robot arm sensors is shown in Tables 3.4 and 3.5.

Table 3.4 Sample of Euler angle results from the robot arm using tool arm sensors while at rest.

Euler Z (Radians)	Euler Y (Radians)	Euler X (Radians)
0.278238171	0.52853157	-90.0214431

Table 3.5 Joint angles of Robot at rest

Angle	Base	Shoulder	Elbow	Wrist 1 (Roll)	Wrist 2 (Pitch)	Wrist 3 (Yaw)
Radians	-0.0002993	-1.570928	-2.3667 e-05	8.39233 e-05	1.570547	-7.230440248 e-05
Degrees	-0.0171474	-90.00756	-0.001356	0.00480845	89.985754	-0.004142737

In the next section, the experimental process is outlined, and details of the post processing of the IMU data using the various filters and neural network models is presented.

3.2 Experiments

In this section the experiments undertaken on the chassis are outlined and details of the post processing of the IMU data using the various filters and neural network models is presented.

Table 3.6 contains a list of the experiments based on the method process of Figure 3.2.
The first column ties the experiment to the method process outlined in Figure 3.2. The table also contains an experiment name, which IMUs will be evaluated during the experiment, which post-processing processes will be undertaken on the data, the names of the MATLAB processing instruction files, and robot movement programs required to undertake the experiment.

The central IMU will be measured for all experiements. For experiments where multiple IMUs are investiated, the front three IMUs form one measurement set and all of the IMUs form the other. The front three IMUs were originally selected as the two outer IMUs share the same alignment as the central IMU, and, if mounted on a vehicle should move in similar directions. This also allows evaluation of the rear IMUs by looking at the differences between the front and the All IMU results. The rear IMUs are further away from the central IMU and, on a production vehicle, might show some differences, as a small change in angle at the front may lead to large movements at the rear, and vice versa.

Tables 3.7 through 3.12 describe the robotic arm movements for each robot movement program listed in Table 3.6. It is to be noted that these angles are endpoints for the robotic arm. Arm movement acceleration is not constant. The first and last quarters of a given movement involve gradual acceleration and deceleration respectively, and the middle period is a constant

250mm/second.

Yaw movement data will not be derived as the IMUs would realistically require magnetometer data to calculate this information and the aluminium baseboard (and steel construction of the vehicle chassis when mounted in production), along with robot motor noise, would likely introduce too much variability to the measurements for a magnetometer to be useful as per the findings of Fan (Fan et al., 2017). This assumption is investigated in Section 4.5.6 using the baseboard chassis.

Table 3.6 Experiments.

Purpose	Experiment Name	IMUs sampled	Process'	Capture Name	Robot Movement Program
Determining sensor noise	Stationary	Central All IMUs	Madgwick Kalman	Stationary	-
Roll sensitivity	Roll	Central	Madgwick Kalman	Roll	Roll
Pitch sensitivity	Pitch	Central	Madgwick Kalman	Pitch	Pitch
Comparisons- Varied movement	Varied	Central Front IMUs All IMUs	Madgwick Kalman NN	Varied (NN training data)	Varied
Comparisons- Untrained data for NN	Varied- Untrained	Central Front IMUs All IMUs	NN	Varied (NN Untrained data)	Varied- Untrained
Is a magnetometer useful?	Magnetometer	Central Front IMUs All IMUs	Madgwick Kalman NN	Magnetometer (NN training data)	Varied
Can complex movements be measured?	Rotation	Central Front IMUs All IMUs	Madgwick Kalman NN	Rotation (NN training data)	Rotation
Complex movements - Untrained data	Rotation- Untrained	Central Front IMUs All IMUs	NN	Rotation (NN Untrained data)	Rotation- Untrained
Is there a bias in the Z axis?	Z Axis Bias	Central FL IMU	Madgwick Kalman	Z Axis bias	Varied
Is there bias in the X and Y axes?	X-Y Axis Bias	Central RL IMU RR IMU	Madgwick Kalman	XY Axis bias	Varied

Table 3.7 Robot Movement Program: Roll (Steps 2 and 3 are repeated)

Name: Sequence Number:	IMU Angles Rotation:Roll:Pitch Angles in brackets are in degrees	Robot Joint Angle (Radians) Base:Wrist1:Wrist2:Wrist3 Rotation:Roll:Pitch:Yaw	Robot Angles (Degrees) Base:Wrist1:Wrist2:Wrist3 Rotation:Roll:Pitch:Yaw
0	0 : 0 : 0	0 : 0 : π/2 : 0	0 : 0 : 90 : 0
1	0 : -π/4(45) : 0	0 : -π/4 : π/2 : 0	0 : -45 : 90 : 0
2	0 : -π/3(-60) : 0	0 : -π/3 : π/2 : 0	0 : -60 : 90 : 0
3	0 : π/3(60) : 0	0 : π/3 : π/2 : 0	0 : 60 : 90 : 0

Table 3.8 Robot Movement Program: Pitch

Name: Sequence Number:	IMU Angles Rotation:Roll:Pitch. Angles in brackets are in degrees	Robot Joint Angle (Radians) Base:Wrist1:Wrist2:Wrist3 Rotation:Roll:Pitch:Yaw	Robot Angles (Degrees) Base:Wrist1:Wrist2:Wrist3 Rotation:Roll:Pitch:Yaw
0	0 : 0 : 0	0 : 0 : π/2 : 0	0 : 0 : 90 : 0
1	0 : 0 : π/6(30)	0 : 0 : 2π/3 : 0	0 : 0 : 120 : 0
2	0 : 0 : π/4(45)	0 : 0 : 3π/4 : 0	0 : 0 : 135 : 0
3	0 : 0 : π/3(60)	0 : 0 : 5π/6 : 0	0 : 0 : 150 : 0
4	0 : 0 : -π/6(30)	0 : 0 : π/3 : 0	0 : 0 : 60 : 0
5	0 : 0 : -π/4(-45)	0 : 0 : -π/4 : 0	0 : 0 : 45 : 0
6	0 : 0 : -π/3(-60)	0 : 0 : -π/6 : 0	0 : 0 : 30 : 0

Table 3.9 Robot Movement Program: Varied

Name: Sequence Number:	IMU Angles Rotation:Roll:Pitch Angles in brackets are in degrees	Robot Joint Angle (Radians) Base:Wrist1:Wrist2:Wrist3 Rotation:Roll:Pitch:Yaw	Robot Angles (Degrees) Base:Wrist1:Wrist2:Wrist3 Rotation:Roll:Pitch:Yaw
0	0 : 0 : 0	0: 0 : π/2 : π/2	0 : 0 : 90 : 90
1	0 : π/12(15) : -π/6 (-30)	0 : π/12 : π/3 : π/2	0 : 15 : 60 : 90
2	0 : π/6(30) : -π/4 (-45)	0 : π/6 : π/4 : π/2	0 : 30 : 45 : 90
3	0 : π/4(45) : -π/6 (-30)	0 : π/4 : π/3 : π/2	0 : 45 : 60 : 90
4	0 : π/4(45) : -π/4 (-45)	0 : π/4 : π/4 : π/2	0 : 45 : 45 : 90
5	0 : π/6(30) : 0	0 : π/6 : π/2 : π/2	0 : 30: 90 : 90
6	0 : π/4(45) : -π/6 (-30)	0 : π/4 : 2π/3 : π/2	0 : 45 : 120 : 90
7	0 : -π/3(-60) : π/6 (30)	0 : -π/3 : π/3 : π/2	0 : -60: 60 : 90
8	0 : -π/3(-60) : 0	0 : -π/3 : π/2 : π/2	0 : -60: 90 : 90

Table 3.10 Robot Movement Program: Varied-Untrained.

Name: Sequence Number:	IMU Angles Rotation:Roll:Pitch Angles in brackets are in degrees	Robot Joint Angle (Radians) Base:Wrist1:Wrist2:Wrist3 Rotation:Roll:Pitch:Yaw	Robot Angles (Degrees) Base:Wrist1:Wrist2:Wrist3 Rotation:Roll:Pitch:Yaw
0	0:0:0	0 : 0 : π/2 : π/2	0 : 0 : 90 : 90
1	0:π/3(60):-π/6(-30)	0 : π/3 : π/3 : π/2	0 : 60 : 60 : 90
2	0:π/6(30):π/4(-45)	0 : π/6 : π/4 : π/2	0 : 30 : 45 : 90
3	0:-π/3(-60):π/6(30)	0 : -π/3 : 2π/3 : π/2	0 : -60 : 120 : 90
4	0:π/3(60):π/6(30)	0 : π/3 : 2π/3 : π/2	0 : 60 : 120 : 90
5	0:π/6(30):-π/6(-30)	0 : π/6 : π/3 : π/2	0 : 30 : 60 : 90
6	0:-π/12(-15):π/6(30)	0 : -π/12 : 2π/3 : π/2	0 : -15 : 120 : 90
7	0:-3π/36(-65):π/18(10)	0 : -13π/36 : 5π/9 : π/2	0 : -65 : 100 : 90
8	0:-5π/18(-50):π/6(30)	0 : -5π/18 : 2π/3 : π/2	0 : -50 : 120 : 90
9	0:-5π/18(-50):-5π/18(-50)	0 : -5π/18 : 2π/9 : π/2	0 : -50 : 40 : 90
10	0:2π/9(40):-5π/18(-50)	0 : 2π/9 : 2π/9 : π/2	0 : 40 : 40 : 90
11	0:5π/18(50):π/6(30)	0 : 5π/18 : 2π/3 : π/2	0 : 50 : 120 : 90

Table 3.11 Robot Movement Program – Rotation

Name: Sequence Number:	IMU Angles Rotation:Roll:Pitch Angles in brackets are in degrees	Robot Joint Angle (Radians) Base:Wr1:Wr2:Wr3 Rotation:Roll:Pitch:Yaw	Robot Angles (Degrees) Base:Wr1:Wr2:Wr3 Rotation:Roll:Pitch:Yaw
0	0:0:0	0 : 0 : π/2 : π	0 : 0 : 90 : 180
1	π/6(30):π/12(15):6π/12(15)	π/6 : π/12 : 7π/12 : π	30 : 15 : 105 : 180
2	π/3(60):0:0	π/3 : 0 : π/2 : π	60 : 0 : 90 : 180
3	π/2(90):-π/12(-15):-π/6(-30)	π/2 : -π/12 : π/3 : π	90 : -15 : 60 : 180
4	2π/3(120):0:0	2π/3 : 0 : π/2 : π	120 : 0 : 90 : 180
5	5π/6(150):-π/12(-15): 6π/12(15)	5π/6:-π/12:7π/12:π	150 : -15 : 105 : 180
6	π (180):0:0	π : 0 : π/2 : π	180 : 0 : 90 : 180
7	7π/6(210):-π/12(-15):6π/12(15)	7π/6:-π/12:7π/12:π	210 : -15 : 105: 180
8	4π/3(240):π/12(15):-6π/12(-15)	4π/3 : π/12 : 5π/12 : π	240 : 15 : 75 : 180
9	3π/2(270):π/9(20):π/36(5)	3π/2 : π/9 : 19π/36 : π	270 : 20 : 95 : 180
10	5π/3(300):-π/9(-20):π/6(30)	5π/3 : -π/9 : 2π/3 : π	300 : -20 : 120 : 180
11	11π/6(330):π/12(15):-5π/36(-25)	11π/6:π/12:13π/36:π	330 : 15 : 65 : 180

Table 3.12 Robot Movement Program – Rotation-Untrained.

Name: Sequence Number:	IMU Angles Rotation:Roll:Pitch Angles in brackets are in degrees	Robot Joint Angle (Radians) Base:Wr1:Wr2:Wr3 Rotation:Roll:Pitch:Yaw	Robot Angles (Degrees) Base:Wr1:Wr2:Wr3 Rotation:Roll:Pitch:Yaw
0	0 : 0 : 0	0 : 0 : π/2 : π	0 : 0 : 90 : 180
1	π/6(30):π/6(30):π/6(30)	π/6 : π/6 : 2π/3 : π	30 : 30 : 120 : 180
2	π/3(60):0:0	π/3 : 0 : π/2 : π	60 : 0 : 90 : 180
3	π/2(90):-π/6(-30):-π/4(-45)	π/2 : -π/6 : π/4 : π	90 : -30 : 45 : 180
4	2π/3(120):0:0	2π/3 : 0 : π/2 : π	120 : 0 : 90 : 180
5	5π/6(150):-π/4(-45):π/6(30)	5π/6 : -π/4 : 2π/3 : π	150 : -45 : 120 : 180
6	π (180) : 0 : 0	π : 0 : π/2 : π	180 : 0 : 90 : 180
7	7π/6(210):-π/4(45):5π/36(25)	7π/6 : -π/4 : 23π/36 : π	210 : 45 : 115: 180
8	4π/3(240):π/12(15):π/12(15)	4π/3 : π/12 : 5π/12 : π	240 : 15 : 75 : 180
9	3π/2(270):π/9(20):-π/3(-60)	3π/2 : π/9 : π/6 : π	270 : 20 : 30 : 180
10	5π/3(300):-π/9(-20):π/6(30)	5π/3 : -π/9 : 2π/3 : π	300 : -20 : 120 : 180
11	11π/6(330):π/4(45):-π/6(-30)	11π/6 : π/4 : π/3 : π	330 : 45 : 60 : 180

Conversion of the IMU data into Euler angles is undertaken by three different main processes. These are a Madgwick Filter, a Kalman filter and , where specified in the experiments table (Table 3.6), at least one neural network model. For each process, a single central IMU is evaluated and then, where specified in the experiments table (Table 3.6), the front three IMUs and then all IMUs. A later experiment investigates the use of magnetometer data.

MATLAB 2023a is used for data processing and analysis, using a MATLAB implementation of the Madgwick filter provided by the author of the filter, an internal MATLAB implementation of a Kalman filter and internal MATLAB-based neural network models. Defaults are used for all filters unless these produced noticably incorrect results, wherepon the adjustments are noted.

The Madgwick filter implementation for MATLAB was created by the author (Madgwick, 2009) and has two internal procedures. The first, UpdateIMU, is used for accelerometer and gyroscope data only, and the second, Update, also adds support for magnetometer data. MATLAB wrapper scripts were created to call these functions (madgwickfilter and madgwickmagfilter, respectively, found in github).

The Kalman filters utilised are provided by MATLAB and use an indirect Kalman filter approach. The first, imufilter, accepts gyrometer and accelerometer data, and the second, ahrsfilter, utilises gyroscope, accelerometer, and magnetometer data to produce the resultant Euler angles. MATLAB wrapper scripts were created to call these functions (kalmanfilter and kalmanmagfilter, respectively, found in github).

Neural network models evaluated are a multilayer Perception (MLP) neural network, a nonlinear

autoregressive exogenous (NARX) neural network and a Radial Basis Function (RBF) neutral network as these have previously been used to perform IMU Euler calculations (Aparna et al., 2020; Mundt et al., 2021; Nadimi et al., 2012; Russell & Fausett, 1996; Sreenivasulu & Deka, 2011; Zeng et al., 2008).

The three different systems (Madgwick filter, Kalman filter and the three Neural network models) are compared on speed of processing, and overall accuracy.

The pseudocode for the MATLAB processing steps is outlined below.

a)Read in IMU and robot arm data.
b)Adjust for mounting angles (add or subtract the mounting angle, π/2, as required.)
c)Get sampling times from robot arm data and store in a new array.
d)Generate individual IMU arrays and set array sizes the same (sampling times will cause differences)
e)Using visual alignment, align robot entries with central IMU entries.
f)Run Madgwick and Kalman filters on data. (Using Magnetometer requires different function calls)
g)Calculate average Euler angles from Front IMUs and all IMUs across all Euler angles.
h)Create Prediction arrays for neural network for central IMU, Front IMUs and all IMUs
i)Create Response array from robot data.
j)Normalise Prediction and Response arrays, and where required, convert row arrays to cells.
k)Train Neural network model/models on data.
l)Evaluate models on untrained data.
m)Plot outputs.

The initial neural network utilised for each experiment is a MATLAB 10-layer multilayer perceptron (MLP) neural network with each individual axis measurement treated as an input (six to nine inputs for a single IMU) to the neural network as shown in Figure 3.14. Outputs of the neural network shall be two - pitch and roll. Training will use the Levenberg-Marquardt (LVM) method. (Bayesian and scaled conjugate training methods were also initially evaluated but these did not offer any significant gains compared to the LVM method). The MLP hidden layers use sigmoid functions, and the output layer will use a linear function. By default, the system will use a random division of data into training, validation, and testing sets of 70%, 15% and 15%, respectively. Different neural network topologies may need to be considered during the results section to increase accuracy.

If a NARX neural network is to be evaluated, the NARX model selected shall be a MATLAB NARXNET model with default settings of 10 layers as shown in Figure 2.1.

The Radial Basis Function (RBF) neural network selected is a standard MATLAB RBF model as shown in Figure 2.2.

After initial training, different numbers of layers and/or different process functions may be implemented to obtain better performance.
An untrained set of data shall be used to verify model performance.

Output plots are generated, and the various results discussed.

In this chapter, the experimental method was outlined, consisting of five IMUs which undergo movement induced by a robotic arm. Data from a single central IMU is processed by Madgwick, Kalman filters and, initially, a 10-layer Sigmoid/Linear MLP neural network to determine Euler angles. Data from the front three and then all five IMUs are then processed by the same three algorithms and compared against the single IMU results. An additional experiment investigates the use of a magnetometer. Experiment details are outlined in Tables 3.6 to 3.12.

In Chapter 4, the results from these experiments are detailed and discussed, with analysis of these in Chapter 5, and conclusions reached in Chapter 6.

Chapter 4

Results

This section contains results from the experiments outlined in Table 3.6, found in Section 3.2. Section 4.1 examines static bias and offset values from the stationary chassis. Sections 4.2 and 4.3 examine rolling and pitching accuracy respectively using the Madgwick and Kalman filters to obtain a reference for comparison with the neural network results. Section 4.4 examines varied angular movements using the single central IMU (Section 4.4.1), the front three IMUs (Section 4.4.2) and all IMUs (Section 4.4.3). The front and all IMUs were selected for evaluation in a live environment where increased separation of the IMUs may offer movement insights beyond Euler angles. These insights are not in scope for this work.

Section 4.5 examines the use of magnetometer data.

Section 4.6 examines roll and pitch Euler conversion while the chassis is under rotational movement with Sections 4.7 and 4.8 examining Z-axis and X and Y axes directional bias in the IMUs, respectively. A summary of the results is presented in Section 4.9.

4.1 Stationary vehicle filter performance

The stationary experiment positions the chassis at rest to record sensor noise and any other deviations from true-zero position.

The robot arm was positioned as per Figure 3.3. This meant that the robot's wrist 2 joint reports an angle of $\pi/2$, at rest, which is corrected when processing the robot position data.

Applying the MATLAB Madgwick function using the UpdateIMU procedure produced the following Euler results on a stationary vehicle, demonstrated in Table 4.1. Settings for the Madgwick filter are defaults. These defaults were a sample period of 1/996 (Author's filter use examples show sample periods set to data size), and a beta value of 0.1. The time taken to process 996 sensor readings from the central IMU was 0.0427 seconds using the tic/toc method in MATLAB. As the code runs faster than the recommended minimum of 1/10 second code runtime recommended for the tic/toc approach, the code was looped 10 times, and the tic/toc result is an average of all runs. The results of the Madgwick filter from the Central IMU (CIMU) are shown in Table 4.1.

Table 4.1 Madgwick filter results (CIMU) from a stationary vehicle.

Angle X (Roll)	Angle Y (Pitch)
Varied from 0.0143 to 0.0381, with a median of 0.02286	Varied from -0.0372 to –0.0082, with a median of -0.0278

An enlarged graph of the calculated data is shown in Figure 4.1. The roll and pitch values shown in Figure 4.1 vary slightly but values are well within the noise floor of the sensor data.

Figure 4.1 Madgwick filter enlarged results (CIMU) on stationary vehicle with default settings.

The default MATLAB Kalman-based imufilter filter was applied. Settings were configured in the kalmanfilter.m script included in the github repository and set to defaults.

Running the imufilter Kalman filter on the 996 sensor readings took 0.7581 seconds and produced an array of Euler angles after the conversion. Figure 4.2 shows a magnified view of the area around the x axis showing that the MATLAB Kalman filter will produce accurate results to within 0.03208 degrees for roll values and within 0.0257 of a degree for pitch values. The Kalman filter processing time was longer than the Madgwick filter processing time (individual conversion times of 0.7581 vs 0.0427 seconds for all 996 sensor readings).

Figure 4.2 Kalman Filter results (CIMU) on stationary vehicle with default settings.

The robot arm was not sampled for this experiment as it was stationary.
The purpose of this experiment was to obtain noise floor data from the IMU so neural network models were evaluated.

4.2 Rolling vehicle filter performance.

In this experiment, IMU sensitivity to rolling (side to side) movements is obtained as per Table 3.6 in Section 3.2. Central IMU data was used for this experiment.

The robot arm was initially positioned as per Figure 3.3. As the Wrist2 joint was positioned at a π/2 (90 degree) angle, results from this joint are rotated by π/2 before processing. The robot arm was repetitively rotated to the angles in listed in Table 3.7 and the Robot arm data synchronised with the IMU data via manual alignment. The single central IMU data produced 1001 readings in the 120 second period. All angles are referenced in the conventional clockwise direction, using the North-East-Down (NED) reference frame.

The Madgwick filter was applied to the central IMU data with standard settings of a sample period of 1/1001 and a default beta value of 0.1 and took 0.0417 seconds to run. The beta value represents the sum of all measurement errors of the gyroscope and is used to weight the accelerometer data in the complementary filter operation. Increasing the beta value weights the accelerometer values higher to compensate for erroneous gyroscope values. It can also be used to obtain quicker convergence but, according to Madgwick, the value would need to be reduced dynamically once convergence is achieved (Madgwick, 2010). The reason for this is that increasing the beta value also increases the step size of the gradient descent algorithm which introduces noise, so optimisation of this value is

critical. The sample period utilised was defined as one over the sample rate, which was set to 1001 to cover all samples, as per the examples provided in Madgwick's code.

The results were clearly incorrect as can be shown in Figure 4.3 below. Results from various beta values are shown in Figures 4.4 through to 4.6.

Figure 4.3 Madgwick results from rolling movement (CIMU) with beta of 0.1 and sample rate of 1001.

Figure 4.4 Madgwick results from rolling movement (CIMU) with a beta value of 1.0.

Using a beta value of 10, the Madgwick filter took 0.0440 seconds but produced better results as can be seen in Figure 4.5.

Figure 4.5 Madgwick filter on a rolling vehicle (CIMU) with a beta value of 10.

By repeated experimentation, a beta value of 15 was the lowest gain required to produce the acceptable results show in in Figure 4.6. This clearly indicates that the gyroscope output is either noisy and/or has significant drift. From looking at the data, and from examining the output of the gyroscope when the vehicle is stationary (Figures 4.1 and 4.2), the amount of noise is significant. The Madgwick filter operation took an average of 0.04997 seconds for the central IMU (10 samples were taken and the average of these used as the individual execution run times were below MATLAB's timing abilities). The results from Figure 4.6 show a maximum measured value of .9948 radians instead of the expected ±π/3 (1.047) radians, and the π/4 (0.7854) radian position were calculated as 0.7487 radians. If the 0.0201 radian maximum tool angle error plus the IMU's own 1.5% inaccuracy values are combined, then this error is within the operating conditions of the sensors but does indicate high noise in the results.

Figure 4.6 Madgwick filter on a rolling vehicle (CIMU) with a gain of 15.

The imufilter Kalman filter was then applied to generate Euler angles. The filter operation took 0.6516 seconds for the central IMU. Both Madgwick and Kalman filter results for the central IMU are shown in Figure 4.7. The average elapsed time for all IMUs was 0.3772 with a median of 0.3051 and max and min values of 0.6516 and 0.2905, respectively, taking 8.0958 times longer than the Madgwick filter.

A low-pass filter should be applied to the Kalman imufilter results to smooth the output, but the results track the movement of the vehicle adequately (varying the Kalman filter settings did not show any significant difference in the output results). The average calculated value for the desired π/3 (1.0472) angle is 1.0297 which is well within specifications of the sensors and tracks better than the 0.9948 reading of the Madgwick result. It is noted that as the Madgwick beta value increases, the starting values of the filter Euler outputs also increase and then converge. Zero outputs occur at initialisation at lower beta levels, rising to almost 1 with a beta value of 15. This appears to be a unique feature of the algorithm as it is consistent across samples.

The main reason for the difference in phasing between the signals is that manual alignment of the robot arm is set based on first recognised movement time, which may not coincide with the sampling time of the central IMU value, however, the performance differences of the two computers will also play a small part in timing differences. The phasing difference is consistent across the entire dataset which confirms an alignment discrepancy.

The other likely reason for timing discrepancies is that it was discovered after the experiments were undertaken that the robotic arm has a maximum stable sampling rate of 500Hz. (This value was

obtained by experimentation). With samples taken at approximately 8 times that rate, it is likely that some discrepancies could occur.

Figure 4.7 Filter and Robot arm data for a rolling vehicle (CIMU): Samples 1-600.

Figure 4.8 Averaged roll value of filters against robot arm movements (CIMU): Samples 1-600.

Plotting the mean of the Madgwick IMU values and the mean of the Kalman IMU values against the robot arm movements shows an amplitude change as seen in Figure 4.8. This indicates that, as expected, averaging the different IMUs decreases peak amplitude in both positive and negative peak values, indicating a wide spread of values obtained from the various IMUs. The experiment shows that utilising the single central IMU only gives more accurate results, with both Kalman and Madgwick results tracking closely, with small phase timings being the most significant point of difference. The Kalman phasing remains constant whereas the Madgwick timing varies. The difference in phasing between Figures 4.5 and 4.6 shows that the Madgwick beta value influences

this phasing difference. Increasing the beta value stabilises the phasing but introduces more noise at the peak transitions. In terms of tracking the robot arm movements, both the Kalman and the Madgwick filters make good attempts. Roll accuracy is within π/180 radians for both filters and there is little between the filter results to merit the use of the more computationally expensive Kalman filter if the best beta values are used on the Madgwick filter.

A neural network was not configured to analyse a simple rolling movement as the purpose of this experiment was to discover IMU sensitivity to rolling movements.

4.3 Pitching vehicle filter performance.

In this experiment, IMU sensitivity to pitching (up and down) movements is obtained. Central IMU data was used for this experiment.

The robot arm was initially positioned as per Figure 3.3. As the Wrist2 joint was positioned at a π/2 (90 degree) angle, results from this joint are rotated by π/2 before processing. The robot arm was repetitively rotated to the angles in listed in Table 3.8 and the Robot arm data synchronised with the IMU data via manual alignment. The single central IMU data produced 995 readings in the 120 second period.

Applying the Madgwick filter with a beta value of 15 (previously determined from Section 4.2) on the central IMU data took an average of 0.04758 seconds.
The Kalman filter took 0.4268 seconds on the same data.
The Kalman filter took 8.97 times longer than the Madgwick filter to process the pitch values.
Figure 4.9 shows the results of the two filters compared to the robot arm data.

Figure 4.9 Filter and robot arm results for a pitching vehicle (CIMU): Samples 1-600.

The results displayed in Figure 4.9 demonstrate that the two filters adequately track the robotic arm movement, with both filters giving comparable results with the same phasing differences as discussed in Section 4.2. Pitch accuracy is within π/90 radians, which is twice the error of the roll sensitivity however there is minor difference between the filter results to merit the use of the more computationally expensive Kalman filter when processing pure pitch movements.

Although not part of the method process, the average pitch accuracy results were examined to see if data from all IMUs could improve the accuracy. Figure 4.10 shows the mean of Madgwick and Kalman results from all IMUs against the robot arm data.

Figure 4.10 Average Filter Pitch values against robot arm movements (CIMU): Samples 1-600

Averaging the values over all IMU pitch results produced considerable errors in amplitude and phase, concentrating at zero crossing points, as seen in Figure 4.10. The Madgwick filter in particular shows considerable oscillation around the zero crossing points when averaged that did not appear when using the single central IMU. The Kalman filter faltered at an average of -0.655 radians while the Madgwick filter oscillated between this value and between +0.589 and +0.606.

These results indicate there are significant data discrepancies between the IMU units and/or variations in the input data between the IMUs. Figure 4.11 shows that the averaging the front three IMUs gives similar results to the single central IMU plot, albeit with predictably smoother plots.

The front three IMUs were selected in the method because these lie on the same horizontal plane of the central IMU which is useful to measure differences in roll values between IMUs, and, on a vehicle, comparing front IMU results against the rear IMUs may show differences when a vehicle traverses a mound or depression, as the rear IMUs are positioned away from the front IMUs and may not register as much movement. When using robotic arm as a control method, there should be no differences in movement between the IMUs, but the sets were kept so that data on a live vehicle can

be compared in future works.

Figure 4.11 Average of front 3 IMU results against robot arm movements: Samples 1-600

The results from Figure 4.11 clearly indicate data errors with the rear IMUs. The reason for the discrepancy was that the left rear IMU chipset was upside down, despite mounting it according to the labels on the unit. Re-mounting and re-calibrating the IMU resolved the issue however replotting the data was not necessary as the results of this experiment have already determined the sensitivity of the central IMU to pitch movements (peak values of π/90 radians).

It was not deemed necessary to configure a Neural Network for a simple pitching movement as the purpose of this experiment was to discover IMU sensitivity to pitching movements.

4.4 Performance on a varied movement vehicle.

In this experiment, IMU Euler results when varied roll and pitch movements (up and down and side to side) are applied to the chassis are obtained. The Madgwick and Kalman filters are evaluated first, with neural network evaluation on the central IMU commencing in Section 4.4.1 and evaluation on the front IMUs and all IMUs contained in Sections 4.4.2 and 4.4.3, respectively.

The robot arm was initially positioned as per Figure 3.4. As the Wrist2 joint was positioned at a π/2 (90 degree) angle from the arm's 0-degree position, results from this joint are counter rotated by π/2 before processing. The robot arm was repetitively rotated to the angles in listed in Table 3.9 and the Robot arm data synchronised with the IMU data via manual alignment. The single central IMU data produced 4998 readings in the 599.5 second period.

The Madgwick filter was applied with standard settings and a previously determined (Section 4.2) beta value of 15 but this time with a sample-rate of 8.33 (averaged sample rate of the data gathering

script for the central IMU). This took an average of 0.2626 seconds to run. The average elapsed time for processing all IMUs was 0.2610 with a median of 0.2626 and max and min values of 0.2675 and 0.2532, respectively. Figure 4.12 shows these results, along with the results of running the imufilter Kalman filter, using default settings. The Kalman filter took an average of 1.8195 seconds to process the data, with a median of 1.6008 and max and min values of 2.3740 and 1.4445 seconds, respectively.

Figure 4.12 Madgwick and Kalman results on varied motion with SR of 8.33 (CIMU): Samples 1-600.

The results shown in Figure 4.12 indicate that the Kalman filter adequately tracks the robot arm movement, as does the Madgwick roll component, but using the actual sample rate (SR) as the sampling rate in the Madgwick filter is clearly producing nonsensical results for Madgwick pitch results. In Figure 4.13, a sampling rate of the size of the sampled data (4998 samples in this case) is selected as per the examples in Madgwick's code, which also does not produce sensible results.

Figure 4.13 Madgwick and Kalman results on varied motion with SR of 4999 (CIMU): Samples 1-600.

Since the Madgwick filter operates by integrating past data, it was surmised that setting the sample rate to the number of time periods used to describe a complete movement set would be ideal. For this set of movement data, the period is approximately 100 samples. Utilising this value in the Madgwick filter parameters produced a better (albeit noisy) result as can be seen in Figure 4.14 but does indicate the importance of setting the correct sampling rate. Too small and the descent algorithm overfits and too large makes the descent algorithm underfit and the gradient descent step value becomes large and introduces additional noise.

Figure 4.14 Madgwick and Kalman results on varied motion with SR of 100 (CIMU): Samples 1-600.

Experimenting with different sample rate values (Figures 4.15 and 4.16 show two examples of this experimentation) indicated that the best fit was the surmised opinion that the sample rate should contain a set of samples that describe a group of movements. While standard processing practice would align the sample rate with the video frame rate (say 1/30th of a second if the frame rate is 30fps), from these results, that may be an oversimplistic approach. With the Madgwick sample rate set to the actual IMU sample rate of 8.33 samples per second, performance was poor. (Figure 4.12). Over a prolonged period with varied motion, the Madgwick method will become increasingly inaccurate without dynamically altering the sample rate. In production, with varied terrain and speed, this will prove difficult to implement reliably and will need to be adjusted based on determined topology. One reason for this interesting result may be the slow sample rate. Madgwick states that his algorithm can process slow sample rates down to 10Hz (with 2 degrees of static error and 7 degrees of dynamic error) but demonstrates that error rates increase logarithmically below this rate in section 5.4 of his work (Madgwick, 2010). With an effective sampling rate of 8.33 Hz, it is likely that the limitations of Madgwick's approach have been reached.

Figure 4.15 Madgwick and Kalman results on varied motion with SR of 50 (CIMU): Samples 1-600.

Figure 4.16 Madgwick and Kalman results on varied motion with SR of 500 (CIMU): Samples 1-600.

Averaging the results across IMUs reduces noise which both tidies up output and partially mitigates the issue, but the problem remains when sampling over an extended period. as shown in Figure 4.17.

Figure 4.17 Averaged results on varied motion with SR of 50 (CIMU): Samples 1-600.

The interplay between sample rate and beta value is demonstrated in Figures 4.18 and 4.19. The sample rate was left at 100 and the beta value was altered down from the previously configured value of 15 (previously determined in Section 4.5.3) to the default value of 0.5 (Figure 4.18), and to 1/3 of the previously determined value of 15, which is 5.0 (Figure 4.19).

Figure 4.18 Filter results on varied motion with SR of 100 and beta of 0.5 (CIMU): Samples 1-600.

Figure 4.19 Filter results on varied motion with SR of 100 and beta of 5.0 (CIMU): Samples 1-600.

Taking the average values across all IMUs with the Madgwick SR set to 100 and the beta value set to 5.0 produced the output displayed in Figure 4.20.

Figure 4.20 Averaged results on varied motion with SR of 100 and beta of 5.0 (CIMU): Samples 1-600.

The interplay between beta value (convergence time and gradient-descent step size) and the sample rate (over how many samples will the algorithm integrate) is complex and depends on the situation. Both topology and vehicle speed are likely to be contributing factors to the correct setting of these

values, and both factors are not static in a rural environment. The correct settings of these values are not in scope for this thesis and is left for future work.

4.4.1 Neural network evaluation using the CIMU.

To examine how effectively a neural network can obtain Euler angles from the single Central IMU measurements, we first need to decide on a neural network model to use.

The general best-practice approach is to use a simple model and then change this to more complex models if this proves insufficient for the task.

The central IMU (CIMU) gyroscope and accelerometer measurement data from the varied movement robotic arm pattern was fed into the initial design of the neural network (NN), specified in the Method section. This design is a 10-layer MLP model with 10 hidden sigmoid layers and 2 linear output layers. Taking all network defaults, the network was trained, and the results obtained are shown in Tables 4.2 and 4.3 and Figures 4.21 to 4.24.

Table 4.2 MLP NN training results (CIMU) on a varied-motion vehicle with default layer size of 10.

	Observations	MSE	R
Training	3499	0.0033	0.9943
Validation	750	0.0053	0.9909
Test	750	0.0065	0.9888

Table 4.3 MLP NN training progress (CIMU) on a varied-motion vehicle with default layer size of 10.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	36	1000
Elapsed Time	-	00:00:01	-
Performance	0.866	0.00322	0
Gradient	1.28	0.00048	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Figure 4.21 MLP NN performance plot (CIMU) on varied-motion with default layer size of 10.

Figure 4.22 MLP NN Error histogram (CIMU) on varied-motion with default layer size of 10.

Figure 4.23 MLP NN regression plot (CIMU) on varied motion with default layer size of 10.

Figure 4.24 MLP NN training state plot (CIMU) on varied-motion with default layer size of 10.

The training time taken as expressed in the MATLAB training window is too granular to be used and so a wrapper was made to run the same commands but with the training function timed. Using this

wrapper, training took 1.5099 seconds. Doubling the number of layers from 10 to 20 produced the results shown in Tables 4.4 and 4.5 and Figures 4.25 to 4.28. Training took 3.2420 seconds.

Table 4.4 MLP NN training progress (CIMU) on a varied-motion vehicle with layer size of 20.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	37	1000
Elapsed Time	-	00:00:03	-
Performance	2.49	0.0023	0
Gradient	5.49	0.00843	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Table 4.5 MLP NN training results (CIMU) on a varied-motion vehicle with layer size of 20.

	Observations	MSE	R
Training	3499	0.0024	0.9960
Validation	750	0.0053	0.9908
Test	750	0.0027	0.9952

Figure 4.25 MLP NN performance plot (CIMU) on varied-motion with layer size of 20.

Figure 4.26 MLP NN error histogram (CIMU) on varied-motion with layer size of 20.

Figure 4.27 MLP NN regression plot (CIMU) on varied-motion with layer size of 20.

Figure 4.28 MLP NN training state plot (CIMU) on varied-motion with layer size of 20.

The regression plot of Figure 4.27 shows the fitting line in the middle of a pleasing random balance of data points on either side, indicating that the fit line is a good representation of the dataset. This good fit result occurs despite the presence of outlying data values.

Dropping the number of layers down to 5 gave the training results shown in Tables 4.6 and 4.7. Training took 0.8577 seconds. These are satisfactory results and only slightly less performant than the 10-layer model, showing that this problem is a simple problem for a neural network to solve.

Table 4.6 MLP NN training progress (CIMU) on a varied-motion vehicle with layer size of 5.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	55	1000
Elapsed Time	-	00:00:00	-
Performance	0.344	0.00714	0
Gradient	0.714	3/63e-05	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Table 4.7 MLP NN training results (CIMU) on a varied-motion vehicle with layer size of 5.

	Observations	MSE	R
Training	3499	0.0071	0.9877
Validation	750	0.0063	0.9892
Test	750	0.0062	0.9892

Results from training with other layer amounts is shown in Tables 4.8 to 4.10.

Table 4.8 Training results on CIMU MLP NN with three layers. Training took 0.8999 seconds.

	Observations	MSE	R
Training	3499	0.0137	0.9760
Validation	750	0.0126	0.9785
Test	750	0.0121	0.9802

Table 4.9 Training results on CIMU MLP NN with two layers. Training took 0.6931 seconds.

	Observations	MSE	R
Training	3499	0.0154	0.9731
Validation	750	0.0151	0.9743
Test	750	0.0153	0.9736

Table 4.10 Training results on CIMU MLP NN with one layer. Training took 0.6315 seconds.

	Observations	MSE	R
Training	3499	0.0751	0.8639
Validation	750	0.0756	0.8527
Test	750	0.0748	0.8610

As seen in Tables 4.8 through 4.10, the test R values did not significantly drop (from 0.9736 at 2 layers to 0.8610 at 1 layer) until the number of layers was lowered to 1 although the testing phase showed an increase in MSE errors (0.0121 with 3 layers from 0.065 at 10 layers) when the number of layers was reduced to 3. This indicates that the data matching element of the problem is relatively simple for the model but that understanding the data relationships requires at least 3 layers. This finding matches the theorem of universal approximators proposed by Hornik (Hornik et al., 1989) that any multi-layer perceptron network of a depth of at least 1 for simple problems and a depth of 3 for more difficult problems can perform as a universal approximator.

Table 4.11 shows the results from feeding the 10-layer MLP NN accelerometer and gryoscope values that have not been used for training.

Table 4.11 CIMU 10-layer MLP NN Training Results from Untrained Varied angle movements.

	Observations	MSE	R
Additional Test	6054	1.0968	-0.8768

The poor untrained results (negative R values indicate that the model is performing worse than just taking average values) along with particularly good, trained results indicates overtraining where the neural network has specialised in deciphering the training data to the detriment of understanding the relationships between the data. This is most likely because the input data was not normalised to values of -1 to 1, which is what the sigmoid neurons expect.

Re-training with normalised values for the CIMU and Robot arm measurements produced the results shown in Tables 4.12 to 4.14. The R values have improved but the untrained test value is still exceptionally low. The negative figure for R indicates that the model performs worse than just taking average values.

Table 4.12 Normalised CIMU 10-layer MLP NN Training Results from varied motion.

	Observations	MSE	R
Training	3499	2.3675e-06	0.9934
Validation	750	3.2994e-06	0.9909
Test	750	3.0495e-06	0.9916

Table 4.13 Normalised CIMU 10-layer MLP NN Training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	55	1000
Elapsed Time	-	00:00:01	-
Performance	0.000441	2.29e-06	0
Gradient	0.000913	4.01e-06	1e-07
Mu	0.001	1e-10	1e+10
Validation Checks	0	6	6

Table 4.14 Normalised CIMU 10-layer MLP NN results from Untrained motion.

	Observations	MSE	R
Additional Test	6054	5.1523e-04	-0.7906

The MATLAB NARXNET NN model was selected with default settings. The NN network model is shown in Figure 2.2. The results from this 10-layer NARXNET training (using LVM training to be consistent with the MLP approach) are shown in Tables 4.15 to 4.17 and Figures 4.29 to 4.34. Using a script, the time taken was 3.1137 seconds.

Table 4.15 Normalised CIMU 10-layer NARXNET NN Training results from varied motion.

	Observations	MSE	R
Training	3497	3.5433e-08	0.9999
Validation	750	3.5455e-08	0.9999
Test	750	4.1173e-08	0.9999

Table 4.16 Normalised CIMU 10-layer NARXNET NN Training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	28	1000
Elapsed Time	-	00:00:01	-
Performance	0.00131	3.54e-08	0
Gradient	0.00269	8.28e-08	1e-07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Table 4.17 Normalised CIMU 10-layer NARXNET NN results from untrained motion.

	Observations	MSE	R
Additional Test	6054	4.5808e-06	0.9902

Figure 4.29 Normalised NARXNET NN training state (CIMU) on varied-motion with layer size of 10.

Figure 4.30 Normalised NARXNET NN performance (CIMU) on varied-motion with layer size of 10.

Figure 4.31 Normalised NARXNET NN error histogram (CIMU) on varied-motion with layer size of 10.

Figure 4.32 Normalised NARXNET NN time-series state (CIMU) on varied-motion with layer size of 10.

Figure 4.33 Normalised NARXNET NN autocorrelation (CIMU) on varied-motion with layer size of 10.

Figure 4.34 Normalised NARXNET NN input-error plot (CIMU) on varied-motion with layer size of 10.

The NARXNET neural network model using 10 layers gave excellent training results, but processing took approximately twice as long (3.1137 seconds) compared to the MLP model (1.5099 seconds). On reducing the layer size to five, all training R values reduced slightly to 0.9998 and untrained test data R scores reduced to 0.9968. Processing the five layers took 2.4845 seconds.

Reducing the layers further did not change training R values but the test values dropped to 0.9974 at three layers and 0.8945 at two layers. With three layers, the time taken to process the CIMU measurements was 1.5774 seconds.

From this, it can be deduced that three layers is the minimum viable layer size for the NARXNET NN model to perform well. A 3-layer NARXNET NN model took 1.5774 seconds to process the CIMU data. The MLP model could not understand the relationships in that timeframe and the Kalman filter results are similar with an average Kalman CIMU processing time of 1.8195 seconds.

An RBF NN implemented was designed using the MATLAB NEWRB function. This function was run on the training data with a target MSE value of 6.96e-06 (evaluated as the lowest of the MSE values when the function was run without specifying a MSE value). (See Figure 4.35).

```
>> net=newrb(r_norm_nn_prediction_array,r_norm_nn_response_array);
NEWRB, neurons = 0, MSE = 0.000172758
NEWRB, neurons = 50, MSE = 8.86913e-06
NEWRB, neurons = 100, MSE = 7.47881e-06
NEWRB, neurons = 150, MSE = 7.36137e-06
NEWRB, neurons = 200, MSE = 7.14484e-06
NEWRB, neurons = 250, MSE = 7.24114e-06
NEWRB, neurons = 300, MSE = 7.23018e-06
NEWRB, neurons = 350, MSE = 7.09703e-06
NEWRB, neurons = 400, MSE = 7.09702e-06
NEWRB, neurons = 450, MSE = 6.95451e-06
NEWRB, neurons = 500, MSE = 6.96744e-06
NEWRB, neurons = 550, MSE = 6.96744e-06
NEWRB, neurons = 600, MSE = 7.23423e-06
NEWRB, neurons = 650, MSE = 6.98723e-06
NEWRB, neurons = 700, MSE = 6.98723e-06
NEWRB, neurons = 750, MSE = 7.25469e-06
NEWRB, neurons = 800, MSE = 7.25469e-06
NEWRB, neurons = 850, MSE = 7.27055e-06
NEWRB, neurons = 900, MSE = 7.31652e-06
```

Figure 4.35 Determining minimal MSE values for RBF network.

The Spread value was left at the default of 1.0. This resulted in an RBF NN of 400 neurons with a performance displayed in Figure 4.36. Training time took 8.6869 seconds.

Figure 4.36 RBF NN Performance Plot from CIMU on varied motion. Spread=1 and Neurons=400.

The RBF NN design does not perform testing, so the untrained measurement set was used with the MATLAB sim operation to evaluate the RBF NN's performance with the following results as seen in Figures 4.37 and 4.38.

Figure 4.37 RBF Roll Plot using untrained varied motion (CIMU). MSE=6.96e-06, SF=1.

Figure 4.38 RBF Pitch Plot using untrained varied motion (CIMU). MSE=6.96e-06, SF=1.

Experimenting with doubling the spread factor (SF) to 2 produced the following results shown in Figures 4.39 and 4.40. Training took 11.1187 seconds using an MSE of 9.49e-06 to produce a 40-neuron network.

Figure 4.39 RBF Roll Plot using untrained varied motion (CIMU). MSE=9.49e-06, SF=2.

Figure 4.40 RBF Pitch Plot using untrained varied motion (CIMU). MSE=9.49e-06, SF=2.

Halving the SF to 0.5 with the same MSE of 9.49e-06 produced the results shown in Figures 4.41 and 4.42. Training took 4.4361 seconds to produce a 15 neuron network.

Figure 4.41 RBF Roll Plot using untrained varied motion (CIMU). MSE=9.49e-06, SF=0.5

Figure 4.42 RBF Pitch Plot using untrained varied motion (CIMU). MSE=9.49e-06, SF=0.5

Using a spread of 0.5 but lowering the MSE to 6.49e-06 produced the results shown in Figures 4.43 and 4.44. Training took 15.9957 seconds to produce a network of 61 neurons.

Figure 4.43 RBF Roll Plot using untrained varied motion (CIMU). MSE=6.49e-06, SF=0.5

Figure 4.44 RBF Pitch Plot using untrained varied motion (CIMU). MSE=6.49e-06, SF=0.5.

Lowering the SF rate introduces noise and increasing the SF reduces overshooting. Increasing the SF to four could not reduce errors to the 6.49e-06 or 9.49e-06 targets. Increasing the MSE to 1.037e-05 produced a network of 62 neurons in 15.5886 seconds and produced the results shown in Figures

4.45 and 4.46.

Figure 4.45 RBF Roll Plot using untrained varied motion (CIMU). MSE=1.037-e05, SF=4.

Figure 4.46 RBF Pitch Plot using untrained varied motion (CIMU). MSE=1.037-e05, SF=4.

The best RBF results occurred when using a SF of one and selecting the lowest MSE that the network could resolve but the processing time is longer than any of the other methods evaluated. This is a surprising result as RBF NN are promoted as being quick to train.

In summary, the Kalman filter performed better than the Madgwick filter but took seven times longer to do so. The MLP NN was not able to recognise the data relationships with any practical number of

layers and the RBF NN could achieve results like the Madgwick filter but took too long to do so. The three-layer NARX NN performed well with similar times to the Kalman filter. It was initially suspected that the NARX implementation in MATLAB had floating-point assistance as the RBF filter is promoted as a fast-training NN but other research has shown that a NARX-style NN has lower processor overhead compared to a RBF NN so the results of this experiment are valid (Xu et al., 2019).

4.4.2 Filter and Neural network performance with the three Front IMUs

Using the inputs of all three front IMUs, the 10-layer MLP NN trained with the Levenberg-Marquardt method performed as outlined in Tables 4.18 to 4.20.

Table 4.18 Normalised Front IMUs 10-layer MLP NN Training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	56	1000
Elapsed Time	-	00:00:01	-
Performance	1.05	0.00185	0
Gradient	2.27	0.000444	1e -07
Mu	0.001	1e-07	1e+10
Validation Checks	0	6	6

Table 4.19 Normalised Front IMUs 10-layer MLP NN Training results from varied motion.

	Observations	MSE	R
Training	3499	0.0019	0.9968
Validation	750	0.0020	0.9964
Test	750	0.0022	0.9962

Table 4.20 Normalised Front IMUs 10-layer MLP NN results from untrained motion.

	Observations	MSE	R
Additional Test	6054	5.2511e-04	-0.7424

Decreasing the layers to five produced the tabulated outcomes shown in Tables 4.21 to 4.23.

Table 4.21 Normalised Front IMUs 5-layer MLP NN training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	46	1000
Elapsed Time	-	00:00:00	-
Performance	0.922	0.00414	0
Gradient	1.5	0.00183	1e -07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Table 4.22 Normalised Front IMUs 5-layer MLP NN training results from varied motion.

	Observations	MSE	R
Training	3499	0.0041	0.9929
Validation	750	0.0039	0.9933
Test	750	0.0052	0.9908

Table 4.23 Normalised Front IMUs 5-layer MLP NN results from untrained varied motion.

	Observations	MSE	R
Additional Test	6054	7.0035e-04	-0.8881

The results show that there is little statistically significant improvement between using the front three IMUs and the central IMU. This is likely because they are on a rigid board and the difference in spacing between the IMUs is too small to detect any differences. The untrained data tests show that the MLP model is not capable of understanding the relationships, even with the assistance of extra IMU input data.

As a NARXNET NN produced the best performance per processing time when using the central IMU alone (Section 4.4), this model was evaluated over the front IMUs. The results are shown in Tables 4.24 to 4.26.

Table 4.24 Normalised Front IMU 10-layer NARXNET NN training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	20	1000
Elapsed Time	-	00:00:01	-
Performance	0.0023	3.45e-08	0
Gradient	0.00431	8.69e-08	1e -07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Table 4.25 Normalised Front IMUs 10-layer NARXNET NN training results from varied motion

	Observations	MSE	R
Training	3497	3.4527e-08	0.9999
Validation	750	3.9120e-08	0.9999
Test	750	3.5716e-08	0.9999

Table 4.26 Normalised Front IMUs 10-layer NARXNET NN results from untrained varied motion

	Observations	MSE	R
Additional Test	6054	5.8264e-06	0.9935

Reducing the NARXNET NN layers to five produced the results shown in Tables 4.27 to 4.29.

Table 4.27 Normalised Front IMU 5-layer NARXNET NN training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	21	1000
Elapsed Time	-	00:00:00	-
Performance	0.000127	3.26e-08	0
Gradient	0.000175	7.64e-08	1e -07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Table 4.28 Normalised Front IMUs 5-layer NARXNET NN training results from varied motion

	Observations	MSE	R
Training	3497	3.2640e-08	0.9999
Validation	750	3.3420e-08	0.9999
Test	750	3.5217e-08	0.9999

Table 4.29 Normalised Front IMUs 5-layer NARXNET NN results from untrained varied motion

	Observations	MSE	R
Additional Test	6054	5.4972e-06	0.9879

Reducing the NARXNET NN layers to three produced the results shown in Tables 4.30 to 4.32.

Table 4.30 Normalised Front IMU 3-layer NARXNET NN training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	24	1000
Elapsed Time	-	00:00:00	-
Performance	0.00043	4.03e-08	0
Gradient	0.000708	9.88e-08	1e -07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Table 4.31 Normalised Front IMUs 3-layer NARXNET NN training results from varied motion

	Observations	MSE	R
Training	3497	4.0341e-08	0.9999
Validation	750	4.1707e-08	0.9999
Test	750	3.8911e-08	0.9999

Table 4.32 Normalised Front IMUs 3-layer NARXNET NN results from untrained varied motion

	Observations	MSE	R
Additional Test	6054	2.4956e-06	0.9953

The NARXNET results from the front IMUs are excellent, but do not differ from the CIMU results by any statistically significant amount. (The front IMUs untrained R result of 0.9953 compared to the CIMU untrained R result of 0.9902, while an improvement, is well within experimental error tolerances).

The RBF network was evaluated by specifying an MSE of 3.15123e-06 (derived from Figure 4.47) which produced an RBF NN of 250 neurons.

```
>> net=newrb(r_norm_nn_3IMUs_prediction_array,r_norm_nn_response_array);
NEWRB, neurons = 0, MSE = 0.000172758
NEWRB, neurons = 50, MSE = 4.98694e-06
NEWRB, neurons = 100, MSE = 3.763e-06
NEWRB, neurons = 150, MSE = 3.49645e-06
NEWRB, neurons = 200, MSE = 3.33359e-06
NEWRB, neurons = 250, MSE = 3.26101e-06
NEWRB, neurons = 300, MSE = 3.24927e-06
NEWRB, neurons = 350, MSE = 3.1629e-06
NEWRB, neurons = 400, MSE = 3.15123e-06
NEWRB, neurons = 450, MSE = 3.19103e-06
NEWRB, neurons = 500, MSE = 3.17285e-06
```

Figure 4.47 RBF MSE Determination from Front IMU data

The results from this RBF network on untrained data using the derived MSE from above and a default spread value (SF) of 1.0 are shown in Figures 4.48 to 4.49. Training took 73.4273 seconds.

Figure 4.48 RBF Roll Plot using untrained varied motion (Front IMUs). MSE=3.15123e-06, SF=1.

Figure 4.49 RBF Pitch Plot using untrained varied motion (Front IMUs). MSE=3.15123e-06, SF=1

Doubling the default SF value to 2.0 produced the following results as outlined in Figures 4.50 and 4.51. The MSE was 4.12939e-06 for a network of 400 neurons. Training time took 84.2626 seconds and produced a network of 250 neurons.

Figure 4.50 RBF Roll Plot using untrained varied motion (Front IMUs). MSE= 4.12939e-06, SF=2

Figure 4.51 RBF Pitch Plot using untrained varied motion (Front IMUs). MSE= 4.12939e-06, SF=2

Halving the default SF to 0.5 produced the results plotted in Figure 4.52 and 4.53. Training took 18.7563 seconds to produce a network of 50 neurons.

Figure 4.52 RBF Roll Plot using untrained varied motion (Front IMUs). MSE= 4.12939e-06, SF=0.5

Figure 4.53 RBF Pitch Plot using untrained varied motion (Front IMUs). MSE= 4.12939e-06, SF=0.5

The RBF network results do not track the robot arm to the same degree as the Kalman filter and are not as accurate as the NARXNET filter and training time took longer than any of the other methods. These results are like the CIMU results and show that the RBF NN is not fit for this workload.

4.4.3 Neural network of a varied-motion vehicle: All IMUs

Results from training with the MLP neural network model on a 10-layer NN with all five IMU inputs are shown in Tables 4.33 to 4.35. Note that MATLAB processing times resolve to one second intervals.

Table 4.33 Normalised All IMUs 10-layer MLP NN training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	31	1000
Elapsed Time	-	00:00:01	-
Performance	3.4	0.00138	0
Gradient	4.59	0.00137	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Table 4.34 Normalised All IMUs 10-layer MLP NN training results from varied motion.

	Observations	MSE	R
Training	3499	0.0014	0.9975
Validation	750	0.0022	0.9962
Test	750	0.0019	0.9968

Table 4.35 Normalised All IMUs 10-layer MLP NN results from untrained varied motion.

	Observations	MSE	R
Additional Test	6054	6.5797e-04	-0.8611

Training with five layers produced the results of Tables 4.36 to 4.38.

Table 4.36 Normalised All IMUs 5-layer MLP NN training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	31	1000
Elapsed Time	-	00:00:00	-
Performance	1.26	0.00277	0
Gradient	2.52	1.27e-05	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Table 4.37 Normalised All IMUs 5-layer MLP NN training results from varied motion.

	Observations	MSE	R
Training	3499	0.0028	0.9953
Validation	750	0.0030	0.9944
Test	750	0.0033	0.9945

Table 4.38 Normalised All IMUs 5-layer MLP NN results from untrained motion.

	Observations	MSE	R
Additional Test	6054	7.0407e-04	-0.8558

The MLP results with all IMUs are like the results from the single central IMU and the Front IMUs and demonstrates that the NN model does not understand the data relationships.

Utilising the NARXNET ten-layer NN on all IMU inputs took 5.3125 seconds and gave the results displayed in Tables 4.39 to 4.41.

Table 4.39 Normalised All IMUs 10-layer NARXNET NN training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	21	1000
Elapsed Time	-	00:00:05	-
Performance	0.00149	1.78e-08	0
Gradient	0.00324	9.69e-08	1e-07
Mu	0.001	1e-00	1e+10
Validation Checks	0	0	6

Table 4.40 Normalised All IMUs 10-layer NARXNET NN training results from varied motion.

	Observations	MSE	R
Training	3499	1.7837e-08	1.0000
Validation	750	2.3714e-08	0.9999
Test	750	3.1325e-08	0.9999

Table 4.41 Normalised All IMUs 10-layer NARXNET NN results from untrained motion.

	Observations	MSE	R
Additional Test	6054	3.6163e-06	0.9911

Reducing the NARXNET NN layers to five produced the results shown in Tables 4.42 to 4.44.

Table 4.42 Normalised All IMU 5-layer NARXNET NN training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	38	1000
Elapsed Time	-	00:00:01	-
Performance	0.000139	2.62e-08	0
Gradient	0.000428	9.26e-08	1e -07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Table 4.43 Normalised All IMUs 5-layer NARXNET NN training results from varied motion

	Observations	MSE	R
Training	3497	2.6227e-08	0.9999
Validation	750	2.9375e-08	0.9999
Test	750	2.9801e-08	0.9999

Table 4.44 Normalised All IMUs 5-layer NARXNET NN results from untrained varied motion

	Observations	MSE	R
Additional Test	6054	4.4583e-06	0.9942

Reducing the NARXNET NN layers to three produced the results shown in Tables 4.45 to 4.47.

Table 4.45 Normalised All IMU 3-layer NARXNET NN training progress from varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	26	1000
Elapsed Time	-	00:00:00	-
Performance	0.000278	3.37e-08	0
Gradient	0.000638	9.23e-08	1e -07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Table 4.46 Normalised All IMUs 3-layer NARXNET NN training results from varied motion

	Observations	MSE	R
Training	3497	3/3714e-08	0.9999
Validation	750	3.8030e-08	0.9999
Test	750	3.6047e-08	0.9999

Table 4.47 Normalised All IMUs 3-layer NARXNET NN results from untrained varied motion

	Observations	MSE	R
Additional Test	6054	2.5198e-06	0.9968

Utilising all IMUs offers no significant statistical benefit over utilising a single IMU, like the results of using the front three IMUs (Section 4.4.2). The untrained R value of 0.9911 with all IMUs is within experimental error when compared to the 0.9935 R value for the front IMUs and the 0.9902 R value for the single CIMU. The number of layers in the NN has more effect but these differences are also within the tolerances of experimental error.

The RBF network was evaluated by specifying an MSE of 1.93051e-06 (derived from Figure 4.54), which produced an RBF NN of 249 neurons.

```
>> net=newrb(r_norm_nn_5IMUs_prediction_array,r_norm_nn_response_array)
NEWRB, neurons = 0, MSE = 0.000172758
NEWRB, neurons = 50, MSE = 4.62331e-06
NEWRB, neurons = 100, MSE = 3.68925e-06
NEWRB, neurons = 150, MSE = 2.86176e-06
NEWRB, neurons = 200, MSE = 2.30426e-06
NEWRB, neurons = 250, MSE = 1.93051e-06
NEWRB, neurons = 300, MSE = 2.00942e-06
NEWRB, neurons = 350, MSE = 1.98984e-06
NEWRB, neurons = 400, MSE = 2.01265e-06
```

Figure 4.54 RBF MSE determination for all IMUs

The results from this RBF network on untrained data using the derived MSE from above and a default spread value (SF) of 1.0 are shown in Figures 4.55 to 4.56. Training took 78.8161 seconds.

Figure 4.55 RBF Roll Plot using untrained varied motion (All IMUs). MSE=1.93051e-06, SF=1.0

Figure 4.56 RBF Pitch Plot using untrained varied motion (All IMUs). MSE=1.93051e-06, SF=1.0

The RBF untrained results with all IMUs are significantly worse than with the Front IMUs or even the single CIMU. Clearly the RBF network has not understood the data relationships with the addition of more IMU input data.

Utilising all IMUs has no significant statistical benefit (with the RBF NN, results were worse with each additional IMU) over utilising a single IMU, like the results of using the front three IMUs (Section 4.4.2). The number of layers within the NARXNET neural network has more effect than the number of IMUs.

The main likely reason for these results is the difference in spacing between the front and rear IMUs is too small to detect any differences and that all the IMUs are on the same plane and should be recording the same angles.

4.5 Would using a magnetometer assist accuracy?

Previous research has shown that using a magnetometer to increase Euler conversion accuracy is not always beneficial because of magnetic interference from motors and the ferrous surroundings when mounted on a vehicle(Fan et al., 2017). When using a robot arm, the aluminium ferrous components (including the baseboard itself) are paramagnetic in that they magnetise in the presence of a magnetic field but will not generate such a field. The magnetometer information was utilised in this experiment to confirm if utilising magnetometer readings will increase Euler conversion accuracy.

The robot arm was initially positioned as per Figure 3.4. As the Wrist2 joint was positioned at a π/2 (90 degree) angle, results from this joint are counter rotated by π/2 before processing. The robot arm was repetitively rotated to the angles in listed in Table 3.9 (varied movement) and the robot arm data was synchronised with the IMU data via manual alignment. Magnetometer readings were utilised as well as gyroscope and accelerometer data in this experiment. After synchronisation, the experiment produced the following Madgwick and Kalman filter results, plotted in Figure 4.57.

Figure 4.57 Kalman and Madgwick CIMU angles (with Magnetometer) against robot arm angles.

These results are like the non-magnetometer-assisted filter results, indicating that the use of a magnetometer near any form of ferrous equipment (even the paramagnetic aluminium baseboard and robot arm) is of no additional benefit, compared to filter measurements made without magnetometer information. The magnetometer was only calibrated using the internal calibration system of the IMU, so some improvements could be possible, but clearly shows that there is marginal benefit, if any, in using magnetometer measurements for roll and pitch determination with this chassis implementation.

The main reason for this lack of improvement in results is that both the Madgwick and MATLAB Kalman-based filters use magnetometer data for heading correction only, and as yaw was not a desired output, the magnetometer input was unutilised in the roll and pitch output results.

Training a 10-layer MLP neural network (NN) on magnetometer-assisted central IMU (CIMU) data produced the results described in Tables 4.48 and 4.49 and Figures 4.58 to 4.60. Note that the timings for processing the 5000 datapoints took less than a second.

Table 4.48 Normalised CIMU 10-layer MLP NN training results with magnetometer under varied motion.

	Observations	MSE	R
Training	4604	0.0172	0.2773
Validation	987	0.0161	0.2020
Test	987	0.0176	0.2070

Table 4.49 Normalised CIMU 10-layer MLP NN training progress with magnetometer under varied motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	13	1000
Elapsed Time	-	00:00:00	-
Performance	0.613	0.0169	0
Gradient	1.38	0.00173	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Figure 4.58 MLP NN training state (CIMU) on varied-motion with 10 layers, using magnetometer.

Figure 4.59 MLP NN performance (CIMU) on varied-motion with 10 layers, using magnetometer.

Figure 4.60 MLP NN error histogram (CIMU) on varied-motion with 10 layers, using magnetometer.

Initial results of a 20-layer MLP neural network on magnetometer-assisted data (Table 4.50) shows that while the MSE readings indicate that the neural network can determine the correct results, the low R values show that the neural network is not understanding the relationships of the data.

Table 4.50 Normalised CIMU 20-layer MLP NN training progress with magnetometer under varied motion.

	Observations	MSE	R
Training	4604	0.0170	0.2669
Validation	987	0.0185	0.2605
Test	987	0.0163	0.1899

The training results of the 20-layer neural network do not differ much from the 10-layer neural network, indicating that increasing the layers of the network model alone is not sufficient to resolve the data relationships. Either the dataset is too small for the neural network to understand the relationships, or data is missing that would permit the neural network to understand the dynamics involved.

These results are surprising and faulty magnetometer data was suspected. A plot of individual magnetometer axis data against the robot arm movements produced the images shown in Figures 4.61 to 4.65. (All data is normalised).

Figure 4.61 X-Axis CIMU Magnetometer values against varied-motion movements: Samples 1-600.

Figure 4.62 Y-Axis CIMU Magnetometer values against varied-motion movements: Samples 1-600.

Figure 4.63 Z-Axis CIMU Magnetometer values against varied-motion movements: Samples 1-600.

Figure 4.64 All axes CIMU Magnetometer values against varied-motion movements: Samples 1-600.

Figure 4.65 All axes CIMU Raw Magnetometer values: Samples 1-600.

As seen in Figures 4.61 to 4.65, the magnetometer data does not align with the robot arm movements. Various possibilities exist for why this problem occurs. The magnetometers were not calibrated beyond the inbuilt calibration systems due to the lack of suitable calibration equipment, but it is assumed that the lack of calibration would alter the amplitude of the plotted waveforms and not the phasing. What is more likely is that the magnetometer is registering the magnetic fields of the motors as they move the arms. (The magnetic field is generated before movement commences). This interference from the motors in the robotic arm, as well as the paramagnetic effects of this interference on the aluminium baseboard chassis, is overwhelming detection of the actual board movement. Besides the phasing disparity, the X and Y axis magnetometer results do track the robotic arm movement, albeit with reduced accuracy on the second peak of robotic arm movement in both pitch and roll movements. From Figure 4.65, X axis second peak magnetic readings are lower than the first peak detection by 7.05µT and Y axis peak readings differ by 3µT. A lack of suitable equipment to independently analyse field strengths meant that these errors were not examined further.

Due to the excessive errors in magnetic measurement with the central IMU, the other NN experiments of comparing the front three and all IMU results with all 9DOF measurements were not evaluated.

4.6 Examining forward (rotating) movement.

As programming the robot to travel a straight-line path (with roll and pitch motions) was not possible due to both time constraints to develop the necessary control program, and the limited range of forward motion possible from the robot, the robot was programmed to travel “forwards” (with respect to the chassis) in an anti-clockwise circular (rotational) direction, and then would travel “backwards”, in a clockwise rotational direction, back to the starting position. The Madgwick and Kalman filters are evaluated first, followed by the neural network models.

The robot arm was initially positioned as per Figure 3.4. As the Wrist2 joint was positioned at a π/2 (90 degree) angle, results from this joint are counter rotated by π/2 before processing. The robot arm was repetitively rotated to the angles in listed in Table 3.9 (varied movement) and the Robot arm data was synchronised with the IMU data via manual alignment.

Results from this experiment are split in to two figures for clarity. Figure 4.66 shows the Kalman filter results from the central IMU against the robot arm movements and Figure 4.67 shows the Madgwick filter central IMU results against the robot arm movements. Note that magnetometer data was not used in this experiment.

Figure 4.66 Kalman results (CIMU) on rotational-motion: Samples 1-600.

Figure 4.67 Madgwick results (CIMU) on rotational-motion: Samples 1-600.

The Kalman filter is more accurate than the Madgwick, with average pitch errors of 0.13 radians (7.4 degrees), compared to the average Madgwick pitch error of 0.19 radians (10.8 degrees). For useful results, both filters need smoothing filters applied on their outputs which will change these results. While not perfect, these results are impressive since both filters specifically mention that their algorithms are not designed to consider other acceleration affects occurring on the chassis while roll and pitch are calculated and the IMU polling rate is below the algorithms' design parameters.

The 10-layer neural network training results from central IMU measurements is shown in Tables 4.51 and 4.52 and Figures 4.68 to 4.71.

Table 4.51 Normalised CIMU 10-layer MLP NN training results from rotational motion.

	Observations	MSE	R
Training	4137	0.0027	0.9710
Validation	887	0.0027	0.9714
Test	887	0.0027	0.9708

Table 4.52 Normalised CIMU 10-layer MLP NN training results from rotational motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	294	1000
Elapsed Time	-	00:00:04	-
Performance	0.507	0.00273	0
Gradient	1.13	8.14e-06	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Figure 4.68 Normalised MLP NN regression (CIMU) on rotational-motion with layer size of 10.

Figure 4.69 Normalised MLP NN training state (CIMU) on rotational-motion with layer size of 10.

Figure 4.70 Normalised MLP NN performance (CIMU) on rotational-motion with layer size of 10.

Figure 4.71 Normalised MLP NN error histogram (CIMU) on rotational-motion with layer size of 10.

These training results are excellent. Analysing the front IMUs gives training results as per Tables 4.53 and 4.54. Processing time was similar (resolution of 1 second) but increased the R value by an average of 0.013. Using all IMUs increased the R value again by an average of 0.005.

Table 4.53 Normalised Front IMUs 10-layer MLP NN training results from rotational motion.

	Observations	MSE	R
Training	4137	0.0015	0.9844
Validation	887	0.0015	0.9842
Test	887	0.0015	0.9845

Table 4.54 Normalised Front IMUs 10-layer MLP NN training progress from rotational motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	53	1000
Elapsed Time	-	00:00:04	-
Performance	0.518	0.00147	0
Gradient	0.924	3.39e-05	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Analysing all IMUs gives training results as per Tables 4.55 and 4.56.

Table 4.55 Normalised All IMUs 10-layer MLP NN training results from rotational motion.

	Observations	MSE	R
Training	4137	0.0010	0.9894
Validation	887	0.0011	0.9882
Test	887	0.0011	0.9885

Table 4.56 Normalised All IMUs 10-layer MLP NN training progress from rotational motion.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	35	1000
Elapsed Time	-	00:00:04	-
Performance	0.518	0.00101	0
Gradient	0.924	0.000451	1e-07
Mu	0.001	1e-08	1e+10
Validation Checks	0	6	6

Adding more layers (taking the total to 20) produced the results displayed in Tables 4.57 and 4.58. The extra second of processing merited a 0.0093 increase in the test R score, which could be argued is not worth the extra processing time.

Table 4.57 Normalised All IMUs 20-layer MLP NN training results from rotational motion.

	Observations	MSE	R
Training	4137	0.0019	0.9802
Validation	887	0.0021	0.9776
Test	887	0.0019	0.9801

Table 4.58 Normalised All IMUs 20-layer MLP NN training progress from rotational motion.

Unit	Initial Value	Stopped Value	Target Value
Epochs	0	222	1000
Elapsed Time	-	00:00:05	-
Performance	2.22	0.00187	0
Gradient	3.18	0.000806	1e-07
Mu	0.001	1e-07	1e+10
Validation Checks	0	6	6

To analyse the effects of complex movements on data not used for neural network training, the MLP 10-layer NN model was fed data from the untrained movement patterns. Results of this are demonstrated in Table 4.59. These results show a large reduction in accuracy compared to the training results above and are so different that it is clear the model is “overfitting” or, in other words, that the model has learnt the specifics of the training data but has not generalised these findings so that untrained data is also recognised.

Table 4.59 Normalised All IMUs 10-layer MLP NN results from untrained rotational motion.

	Observations	MSE	R
Additional Test	6426	0.1360	0.4015

As the data is normalised, and has trained well, it is likely that the MLP NN model type is incapable of recognising the relationships. The rotational movement MLP NN R value of 0.4015 is significantly improved over the varied movement results (-0.8768) but is still extremely low.

Utilising the NARXNET NN, the results presented in Tables 4.60 to 4.62 and Figures 4.72 to 4.75 were obtained. The training time took 4.4235 seconds.

Table 4.60 Normalised CIMU 10-layer NARXNET NN training progress from rotational motion.

Unit	Initial Value	Stopped Value	Target Value
Epochs	0	25	1000
Elapsed Time	-	00:00:04	-
Performance	0.00347	7.74e-08	0
Gradient	0.00477	9.95e-08	1e-07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Table 4.61 Normalised CIMU 10-layer NARXNET NN training results from rotational motion.

	Observations	MSE	R
Training	4137	7.7421e-08	0.9998
Validation	887	8.4386e-08	0.9998
Test	887	6.7285e-08	0.9998

Table 4.62 Normalised CIMU 10-layer NARXNET NN results from untrained rotational motion.

	Observations	MSE	R
Additional Test	6426	1.8488e-06	0.9937

Figure 4.72 Normalised NARXNET NN reponse (CIMU) on rotational-motion with layer size of 10.

Figure 4.73 Normalised NARXNET NN training state (CIMU) on rotational-motion with layer size of 10.

Figure 4.74 Normalised NARXNET NN Performance (CIMU) on rotational-motion with layer size of 10.

Figure 4.75 Normalised NARXNET NN Error histogram (CIMU) on rotational-motion with layer size of 10.

The RBF network was evaluated on the CIMU by specifying an MSE of 2.5806e-05 (derived from Figure 4.76, which produced an RBF NN of 47 neurons.

```
>> newrb(r_norm_nn_prediction_array,r_norm_nn_response_array)
NEWRB, neurons = 0, MSE = 0.000164731
NEWRB, neurons = 50, MSE = 2.63845e-05
NEWRB, neurons = 100, MSE = 2.59033e-05
NEWRB, neurons = 150, MSE = 2.58795e-05
NEWRB, neurons = 200, MSE = 2.58063e-05
NEWRB, neurons = 250, MSE = 2.60192e-05
NEWRB, neurons = 300, MSE = 2.60816e-05
```

Figure 4.76 RBF MSE determination for CIMU under rotational movement.

The results from this RBF network on untrained CIMU rotational motion measurements using the derived MSE from above and a default spread value (SF) of 1.0 are shown in Figures 4.77 to 4.78. Training took 15.5184 seconds.

Figure 4.77 RBF Roll Plot using untrained rotational motion (CIMU). MSE=2.5806e-05, SF=1.0

Figure 4.78 RBF Pitch Plot using untrained rotational motion (CIMU). MSE=2.5806e-05, SF=1.0

The RBF network was evaluated on the front IMUs by specifying an MSE of 1.05961e-05 (derived from Figure 4.79, which produced an RBF NN of 166 neurons.

```
>> newrb(r_norm_nn_3IMUs_prediction_array,r_norm_nn_response_array)
NEWRB, neurons = 0, MSE = 0.000164731
NEWRB, neurons = 50, MSE = 1.25485e-05
NEWRB, neurons = 100, MSE = 1.08939e-05
NEWRB, neurons = 150, MSE = 1.06796e-05
NEWRB, neurons = 200, MSE = 1.05961e-05
NEWRB, neurons = 250, MSE = 1.06279e-05
```

Figure 4.79 RBF MSE determination for Front IMUs under rotational movement.

The results from this RBF network on untrained Front IMU rotational motion measurements using the derived MSE from above and a default spread value (SF) of 1.0 are shown in Figures 4.80and 4.81. Training took 56.1377 seconds.

Figure 4.80 RBF Roll Plot using untrained rotational motion (Front IMUs). MSE=1.05961e-05, SF=1.0

Figure 4.81 RBF Pitch Plot using untrained rotational motion (Front IMUs). MSE=1.05961e-05, SF=1.0

The RBF network was evaluated on all IMUs by specifying an MSE of 5.76837e-06 (derived from Figure 4.82, which produced an RBF NN of 325 neurons.

```
NEWRB, neurons = 0, MSE = 0.000164731
NEWRB, neurons = 50, MSE = 8.3443e-06
NEWRB, neurons = 100, MSE = 7.24469e-06
NEWRB, neurons = 150, MSE = 6.5764e-06
NEWRB, neurons = 200, MSE = 5.90941e-06
NEWRB, neurons = 250, MSE = 5.89535e-06
NEWRB, neurons = 300, MSE = 5.79382e-06
NEWRB, neurons = 350, MSE = 5.76837e-06
NEWRB, neurons = 400, MSE = 5.79373e-06
```

c

Figure 4.82 RBF MSE determination for All IMUs under rotational movement.

The results from this RBF network on untrained rotational movement measurements using the derived MSE from above and a default spread value (SF) of 1.0 are shown in Figures 4.83 and 4.84. Training took 135.1606 seconds.

Figure 4.83 RBF Roll Plot using untrained rotational motion (All IMUs). MSE=5.76837e-06, SF=1.0

Figure 4.84 RBF Pitch Plot using untrained rotational motion (All IMUs). MSE=5.76837e-06, SF=1.0

The RBF NN model does not produce good results with any number of IMUs andis not recommended for use. The NARXNET NN model produces very good results with good process timings, similar to the Kalman results. Utilising additional IMUs offers no statistically significant benefit compared to using a single IMU with varied pitch and roll movements occuring while under rotation.

4.7 Exploring Z-axis bias values.

This experiment determines if there is any bias in the Z-axis of the IMU by inverting the IMU to determine if Z-axis values are the same in both directions.

From Section 4.3, the experimental setup accidentally included an IMU mounted upside down which caused average values to be miscalculated. In this experiment, a single IMU is investigated to explore if z-axis sensitivity is biased. The IMU selected was the front left IMU as it was the easiest to reposition. Figure 4.85 and Table 4.63 show the comparative accelerometer measurement values when re-orienting the FL IMU upside down. Ignoring the phasing issues (the timings between these two samples were not aligned) and the peak spike values, the mean values show an alignment discrepancy of 0.012m/s^2. It is interesting that the spikes in measurement values only appear when

the IMU is inverted. It is not yet clear why these spikes occur.

Figure 4.85 Front-Left IMU Zaxis Accelerometer values compared after inverting the IMU.

Table 4.63 Front-left IMU Accelerometer measurements after inverting the IMU

Measurement	Standard positioning (meters per second squared)	IMU inverted (meters per second squared)
Minimum Acceleration	-19.6133	-0.2227
Maximum Acceleration	19.6127	13.1633
Mean Acceleration	7.2518	7.2438
Median Acceleration	7.0150	7.1275

Figure 4.86 and Table 4.64 show the comparative gyrometer measurement values when re-orienting the FL IMU upside down. Ignoring the phasing issues (the timings between these two samples were not aligned), the mean values show an alignment discrepancy of 0.027rps. It is interesting to note that there are no spikes in gyrometer measurements when the IMU is inverted, unlike the accelerometer measurements. It is not yet clear why these spikes occur.

Figure 4.86 Front-left IMU Zaxis Gyrometer values compared after inverting the IMU

Table 4.64 Front-left IMU Gyrometer measurements after inverting the IMU

Measurement	Standard positioning (Radians per second)	IMU inverted (Radians per second)
Minimum Rotation rate	-0.3769	-0.3612
Maximum Rotation rate	0.3984	0.3949
Mean Rotation rate	0.0190	0.0177
Median Rotation rate	-0.0033	0.0028

This experiment found there is directional bias in the Z-axis of 0.0027rps for the front left IMU gyrometer measurements and 0.012m/s^2 for front left IMU accelerometer measurements.

4.8 Exploring X and Y axis bias values.

This experiment determines if there is any bias in the X and Y axes of the IMU by rotating two IMUs by 90 degrees (π/2 radians) to determine if data readings alter after the rotations.

Keeping the front-left IMU upside down, the left-rear IMU was turned 90 degrees to the left (-π/2 radians) and right-rear IMU was turned 90 degrees to the right (π/2 radians) and the experiment repeated. The accelerometer results are shown in Figure 4.87 and Table 4.65.

Figure 4.87 Accelerometer values after rotating rear IMUs by pi/2 (90 degrees)

Table 4.65 Accelerometer measurements from rotating rear IMUs by pi/2(90 degrees)

Measurement	IMU inverted (m/s squared)
CIMU X-axis Minimum Acceleration (Standard orientation)	-7.0868
CIMU X-axis Maximum Acceleration (Standard orientation)	4.7166
CIMU Y-axis Minimum Acceleration (Standard orientation)	-9.88557
CIMU Y-axis Maximum Acceleration (Standard orientation)	10.4722
RL IMU X-axis Minimum Acceleration (Rotated Left)	-8.9423
RL IMU X-axis Maximum Acceleration (Rotated Left)	10.2161
RL IMU Y-axis Minimum Acceleration (Rotated Left)	-7.4555
RL IMU Y-axis Maximum Acceleration (Rotated Left)	4.9608
RR IMU X-axis Minimum Acceleration (Rotated Right)	-10.3190
RR IMU X-axis Maximum Acceleration (Rotated Right)	8.9854
RR IMU Y-axis Minimum Acceleration (Rotated Right)	-4.9153
RR IMU Y-axis Maximum Acceleration (Rotated Right)	7.5944

The CIMU Y-axis accelerometer measurements track closely (within 0.2 m/s^2) with the rear IMUs if the IMU rotations are considered (Y axis on CIMU measures in the same plane as the X axis on the rear IMUs and vici versa). The CIMU X-axis accelerometer measurements do not track as closely (within 1.9 m/s^2) with the rear IMUs if the IMU rotations are considered (X axis on CIMU measures in the same plane as the Y axis on the central IMU and vici versa).

The gyroscope readings after rotating the rear IMUs are displayed in Figure 4.88 and Table 4.66.

Figure 4.88 Gyroscope values after rotating rear IMUs by pi/2 (90 degrees)

Table 4.66 Gyrometer measurements from rotating rear IMUs by pi/2(90 degrees)

Measurement	IMU inverted (Radians per second)
CIMU X-axis Minimum Rotation rate (Standard orientation)	-1.1013
CIMU X-axis Maximum Rotation rate (Standard orientation)	1.1731
CIMU Y-axis Minimum Rotation rate (Standard orientation)	-0.9486
CIMU Y-axis Maximum Rotation rate (Standard orientation)	1.1744
RL IMU X-axis Minimum Rotation rate (Rotated Left)	-0.9659
RL IMU X-axis Maximum Rotation rate (Rotated Left)	1.0846
RL IMU Y-axis Minimum Rotation rate (Rotated Left)	-1.1073
RL IMU Y-axis Maximum Rotation rate (Rotated Left)	1.1021
RR IMU X-axis Minimum Rotation rate (Rotated Right)	-1.1720
RR IMU X-axis Maximum Rotation rate (Rotated Right)	0.8675
RR IMU Y-axis Minimum Rotation rate (Rotated Right)	-1.0961
RR IMU Y-axis Maximum Rotation rate (Rotated Right)	1.1088

Both X and Y axis gyroscope measurements of the CIMU track closely to the rear IMUs (within 0.002 radians per second) if the IMU rotations are considered (X axis on the CIMU measures in the same plane as the Y axis on the rear IMUs and vici versa).

This experiment has ascertained that the IMU Y-axis has directional biases of 0.2m/s^2 in accelerometer measurements and the IMU X-axis had a directional accelerometer bias of 1.9m/s^2. The gyrometer bias was 0.002 rps for both axes on both IMUs.

4.9 Summary of results.

This section provides a summary of the examined results.

The Kalman filter is more accurate than the Madgwick filter as the Madgwick filter has a lower limit of a 10Hz sampling frequency and the IMUs were polled at an average of 8.33Hz, but the Kalman filter takes 6.9 times longer to process the same measurements. The Madgwick filter is very sensitive to beta (weighting of accelerometer data) and the sample length, which should be set to a multiple of the number of samples comprising a sequence of movements being converted by the filter. After calibration, an IMU at rest may still exhibit up to 0.03208 degrees of roll variation and 0.0257 of pitch variation, an indication of sensor noise. The Kalman filter calculated the actual resting position of 1.047 ($\pi/3$) radians as 1.0297 radians. These measurements are within the noise floor of both the robot arm and IMU but only just. (0.02 radians is 1.14 degrees). IMU Pitch movement accuracy is within $\pi/90$ radians (2 degrees) and roll sensitivity is $\pi/180$ (1 degree). Z-axis directional bias was measured as 0.012m/s^2 for the accelerometer and 0.027rps for the gyroscope. Directional biases were measured as 0.002rps for both gyroscope axes (X and Y) and 0.2m/s^2 for the Y-axis accelerometer measurements and 1.9m/s2 for the X-axis accelerometer measurements.

The MLP NN was not able to understand the data relationships with either the varied-movement or rotational movements (although interesting the MLKP NN performed better on the more complex rotational movements (-0.8881 on varied movements with 10 layers and 0.4015 on rotational movements with 10 layers for the single CIMU device). The RBF NN performed adequately but was terribly slow (at least four times longer than the MLP NN for the fastest recoded time). The NARXNET NN results were excellent (untrained R values were a minimum of 0.99) and (with three layers used) performed in a similar period to the Kalman filter.

In Chapter 5, analysis of the results is undertaken with conclusions reached in Chapter 6.

Chapter 5

Discussion and Analysis

5.1 Introduction

This chapter discusses the results of the experiments shown in Chapter 4. It begins with a reflection on the performance of the experimental design in Section 5.2, and then goes on to discuss the results of individual experiments in Section 5.3, discusses the research objective in Section 5.5, before concluding with a summary in Section 5.6.

5.2 Experimental Design Analysis

The execution of the experimental design provided the data required for analysis but several areas for improvement were noted. These include improving the sampling rate of the IMUs, reducing the sampling rate of the robot arm, improving robot arm and IMU data alignment, reducing magnetic interference and implementing a multiple IMU process to measure yaw.

The low IMU sampling rate of 8.33 Hz could be improved by removing the requirement for an I^2C multiplexor, and ideally, utilising individual Universal Asynchronous Receiver-Transmitter (UART) interfaces for each IMU. If the selected IMUs do not support UART interfacing, then individual SPI connections should be used, with individual I^2C connections as a last resort.

The robot arm polling program is limited (discovered by experimentation) to 500Hz sample rates to obtain predictable timing periods. The maximum sampling rate of the computer was utilised for all these experiments (approximately 4603Hz) and is likely to have heavily influenced the timing phase differences between the computer timings and the robot arm timings. Future experiments should restrict robot arm polling to a rate that is officially supported by the robot arm and/or switch to a direct connection such as Universal Serial Bus (USB) or a serial port.

Data alignment is a manual process, relying on the experimenter manually aligning the start of movement between the robotic arm measurements and the IMU measurements.
One possibility to increase the accuracy of timing alignments is to provision two time-synchronised computers, one directly connected to the vehicle-processor and the other directly to the robot arm. Using reliable and consistent time-stamped data would decrease the time necessary to manually align readings and it is possible that this can be automated. This would also permit automated timing alignments of the other IMUs to the robotic arm movements.
It is to be noted that this timing concern is only of importance for verification that the system is

performing well and is not required in operation. It is expected that, in production, the processor will be running other tasks such as vehicle control and route-planning, etc, meaning truly predictable timing is not possible without the use of a real-time operating system.

The experiments did not attempt to measure yaw due to two assumptions. The first is that roll, and pitch movements more accurately reflect a bumpy topology than yaw movements (yaw would indicate the vehicle is sliding in some manner, which should be managed by the vehicle guidance system). The other assumption is that magnetometer data would be invalid when in the IMUs were positioned close to motors and the aluminium chassis and the results of the experiments of Section 4.5.3 agree with that assumption. Future investigation is required to determine how close the IMUs need to be to a (usually steel) UGV (rover) chassis and/or robotic arm motors without magnetic effects affecting accurate measurements. To support this investigation, the baseboard should be changed to an all-plastic construction such as acrylonitrile butadiene styrene (ABS) plastic, and mounting equipment (including brackets) should be nylon.

The bias experiments of Sections 4.7 and 4.8 and the calibration results Section 4.1.1 show that there are small bias differences between positive and negative directions on all calibrated axes and large uncalibrated biases and noise levels amongst the IMUs. The X-axis has the highest uncalibrated accelerometer discrepancies (1.9 m/s^2 accelerometer offset for the FR IMU). The Y-axis has the highest gyrometer uncalibrated discrepancies (0.02 rps for the CIMU). When calibrated, the results from Sections 4.7 show that the FL IMU Z-axis gyroscope bias is 0.027rps and the Z-axis accelerometer bias is 0.012m/s^2, depending on orientation. The results from Section 4.8 show that the RR X-axis gyroscope bias is different (0.1 rps), and that the RR accelerometer bias is also large (1.9m/s^2) depending on orientation. These values are statistically significant when considering the accuracy of Euler conversions. Long term gyrometer drift was not considered in this experiment due to time constraints.

A suggested method to obtain yaw values is to use 4 IMUs in two pairs of two. One pair should be mounted horizontally and the other vertically, with each member of the pair mounted in the opposite direction of the other, both vertically and laterally. The purpose of this approach is to also observe if averaging the IMU results will result in directional biases and drift biases cancelling each other.

The robot arm provides predictable angles once calibrated however the robot joint construction and arm length does not easily permit roll and pitch modifications while travelling in forward/reverse directions except for exceedingly small distances. As such, this experiment does not consider roll and pitch movements while under any form of direct forward/reverse motions although it does examine

rotational movements in forward and reverse directions. The robot arm is also only capable of moving at 250mm per second without generating faults and this is not reflective of the higher accelerations that a production vehicle is likely to experience at speed.

Ideally, a track should be designed with accurate angles so that the effects of normal forward motion while the vehicle is undergoing pitch and roll movements can be analysed. The time constraints of this experiment did not permit the construction of such a track that is long enough to give a good set of data useful for training purposes. The measurement apparatus will need to include a means to verify vehicle angles (strobe-camera/pressure sensors, etc) because the RC vehicle contains oil-filled suspension and wishbone suspension springs so determining the actual position and rotation of the vehicle chassis will be challenging as it can't be based upon ground-angles alone. A track design of this configuration was not in scope, however future projects should investigate this style of experiment to verify vehicle chassis movement under pitch and roll conditions while travelling forwards, or at least, use a robotic arm that is capable of simultaneous roll, pitch, yaw and forward motions (the forward motion should contain both static and accelerating speeds).

5.3 Filter results analysis.

Caruso et al. stated in their work that there is no perfect filter for Euler conversion and these results certainly indicate the truth of this (Caruso et al., 2021).

The results sections show that using a Madgwick complimentary filter with gradient-descent optimisation to fuse gyroscope and accelerometer data to predict roll, pitch and yaw values is 95.3% accurate (0.9948 measured against 1.047 actual and 0.7487 measured against 0.7854 actual) when the timing of the datasets are predictable (simple rolling and pitching movements) but, on varied movements, the filter would need to be "reset" every so often to match the pattern of the data to prevent the accumulation of errors. This is especially significant when the vehicle is required to change its speed when responding to local conditions. While this issue can be partially explained by the inconsistencies of the alignment of the IMU and robot arm data, in operation, these timing inconsistencies are likely to be present due to processor scheduling. It is recommended that a Madgwick filter be implemented only if the sets of movements can be predicted/determined so the gradient descent algorithm does not over or under fit. This finding matches the results of Caruso (Caruso et al., 2021) where they outline that parameter selection is critical for filter-based sensor fusion approaches.

As such, based on these results, it is not recommended that a Madgwick implementation be used for Euler angle conversion for a rural rover implementation. The sample rate of the IMUs is lower than the recommended parameters for this filter, so if the IMU sample rate can be increased to at least 15 Hz, it is recommended that this filter be evaluated again before implementation.

Because of the low performance of this filter, it is not used as a comparison with the NN models.

The Kalman filter approximates the robot arm control movement slightly more accurately than the Madgwick and does not appear reactive to the length of the data samples fed into it but overshoots considerably (up to 0.4 radians (4 degrees), ignoring random spikes) on changes of angle (Figure 4.19) and takes at least 6.9 times the processing time compared to the Madgwick filter (1.8195 seconds versus 0.2626 seconds on varied movements) but this timing will vary depending on the processor used. Reducing the overshooting behaviour will certainly alter responsiveness to change (this is a classic filter dynamic) and optimal values would need to be adjusted to the vehicle's speed.

Of the two filters, the Kalman is preferable for a long-range rover to avoid the sampling size conditions of the Madgwick filter, despite the additional processing overhead and overshooting complications. These findings explain the popularity of the filter, despite its additional processing time compared to the simpler Madgwick filter. As such, the Kalman filter values are used for comparison with the results from the neural network experiments. In production, the output of the Kalan filter will require smoothing with a digital low-pas filter.

Both the Madgwick and MATLAB Kalman-based filters use magnetometer data for heading correction only, and as yaw was not a desired output, the invalid magnetometer data discovered when analysing the poor neural network results (Section 4.5.3) did not affect the filter Euler roll and pitch output values. The magnetometer readings indicate that the IMUs are registering the stronger robot arm motor magnetic fields instead of the earth's magnetic field. Other researchers have noted that the use of magnetometer data is limited in the real world to specific use cases (Caruso et al., 2021; Harindranath & Arora, 2018; Taylor et al., 2017).

5.4 Neural Network results analysis.

With more complex movements, the differences between the Madgwick and Kalman filters and the NNs becomes noticeable. While the Madgwick and Kalman filters can determine the Euler angles in a broad sense, the MLP NN (with any reasonable number of layers) struggled to understand the relationships. As the MSE values were good, this usually indicates overtraining, which is where the NN specialises in fitting the training data to the detriment of understanding the overall relationships and is normally indicative of a model's incompatibility with the data.

A radial basis filter (RBF) neural network with a spread of one used 400 neurons to provide slightly worse results compared to the Kalman filter, but also with a significant time penalty of 8.6869 seconds, four times longer than the Kalman filter or five-layer NARX NN model (Section 4.5.4). As such, the RBF NN model is not advised for this workload.

Selecting the MATLAB NARXNET NARX RNN model gave excellent training and test results but incurred a higher processor timing period compared to the Kalman filter with the default setting of 10 layers. (3.137 seconds vs 2.1066 seconds for the Kalman filter). With 5 layers, the NARXNET results took 2.4845 seconds against the Kalman filter's 2.1066 seconds which is well within experimental error tolerances. With a three-layer NARXNET model, the NN took 1.5774 seconds compared to the Kalman filter's 1.8195 seconds.

The NARX RNN is the best choice of NN model for Euler conversion, but computing resources need to be considered. The Raspberry Pi may need to be upgraded to a more powerful processor that supports Tensorflow operations or another NN toolset, which is a significant cost in both dollars and power usage. An evaluation of a NARX implementation on an Internet of Things (IoT) level processor should be undertaken in a future work.

5.5 Research Objectives analysis.

The addition of extra layers and/or adding two and then four IMU units to the neural network inputs did give slight improvements but these are not statistically significant. There are several reasons for this result.

1. The additional IMU data is not adding much more information than the central IMU, as the neural network will already have determined the relationship of the central IMU position to the desired results (including noise and drift measurements), and any additional IMU data is confusing the system with extraneous data and noise. If the IMUs were mounted in different reference frames, or if the frame was not completely rigid, then it is likely that additional IMUs would give useful information, but these approaches were not explored.
2. Despite comparing the rear IMU set to the front IMU sets, the spacing between them may be too small to obtain measurable differences in performance.

If using a NARX NN model, it is recommended to set the number of layers to three to obtain an effective balance between low processing time and conversion accuracy. To use such an NN, the processor would need to be upgraded to support Tensorflow, which is both a financial and power consumption cost. It is recommended that such processors be evaluated for effectiveness in future research.

5.6 Summary

This thesis has answered the research objectives and produced the following findings, listed in bullet form, below.

- Research Objective One is to determine whether multiple inertial measurement units (IMUs) in conjunction with a neural network can improve image stabilisation of a camera on an RC vehicle, compared with a single IMU. The NARXNET NN models untrained data R value for a single IMU was 0.9902. This increased 0.9935 when using the Front Three IMUs and increased further to 0.9953 with all five IMUs being utilised, so technically that is a yes, but these figures are within experimental error tolerances and are not statistically significant.

- Research Objective Two is to determine the least number of inertial measurement units required to provide a significant measurable improvement. The least number of IMUs to produce a satisfactory output is one however, after obtaining the bias and noise results from Sections 4.1-4.3. 4.5 and 4.7 to 4.8, it is recommended that two sets of a pair of IMUs be implemented. One set should be mounted horizontally and the other vertically, to determine if yaw can be measured without the use of a magnetometer. Each IMU in the pair should be mounted inversely to the other, both vertically and laterally. i.e., One should appear upside down compared to the other with X axes reversed. This method is to determine whether IMU biases and offsets can cancel each other out and to investigate the long effect of gyroscope drift on the neural network results.

- Research Objective Three is to determine the least number of layers a neural network might need to provide accurate roll and pitch data from IMU inputs. The least number of layers for an accurate functional neural network to calculate Euler angles from IMU inputs is three, if using a NARXNET neural network model.

- The Madgwick filter is sensitive to the beta value (weighting factor of accelerometer input into the complimentary filter) and the length of the data being sampled, with respect to the pattern of that data. As such, it is not recommended for use with a rural rover workload.

- The multilevel perceptron (MLP) and Radial Basis Function (RBF) neural network models are not suitable for a rural rover workload. The nonlinear, autoregressive external input (NARX) recurrent neural network (RNN) model performs extremely well, even with as few as three hidden layers. The timing of the three-layer NARX NN model was less than the Kalman filter timing, indicating that, if the processor can support such a model, it should be preferred.

- The Kalman filter is an accurate and efficient filter and should be implemented where the processor specification prevents the use of a NARX neural network model.

- Robotic arm motor magnetic fields will swamp earth magnetic field measurements.
  If magnetometer measurements are required, chassis construction and mounting parts should be non-magnetic (no paramagnetic components) and nearby motor shielding may be required.

- A more flexible robotic arm or a specific track layout should be used to obtain real world accelerations and the effects of these on measurements.

- If using a robotic arm to create test angles, NTP should be implemented to synchronise time between machines to reduce the workload of manual data alignment.

- The chassis board is too small to effectively measure differences in IMU placement. At least a 1/5 scale chassis should be utilised. The actual size will need to be determined by evaluation.

- The sample rate for the IMU needs to be at least 15Hz and the sample rate for the robot arm should be reduced to below 500Hz to mitigate both timing issues and inaccurate measurements.

In Chapter 6, conclusions and future recommendations are discussed, with Appendices listed in Chapter 7 and references contained in Chapter 8.

Chapter 6. Conclusion and Future Works

The research objectives have been answered through this experiment.

Research Objective One was to determine whether multiple inertial measurement units (IMUs) in conjunction with a neural network can improve image stabilisation of a camera on an RC vehicle, compared with a single IMU. The experiment Sections 4.4 and 4.6 demonstrate that multiple IMUs offer little statistically significant benefit compared to a single IMU when all IMUs are in the same (small) plane of reference. The NARXNET NN model's untrained data R value for a single IMU was 0.9902. This increased 0.9935 when using the Front Three IMUs and increased further to 0.9953 with all five IMUs being utilised but these figures are within experimental error tolerances.

Research Objective Two was to determine the least number of inertial measurement units required to provide a significant measurable improvement. The experiment has proven that one IMU is sufficient to provide accurate roll and pitch data but suggests, because of the significant levels of noise, bias and offset values that IMUs possess (Sections 4.1 to 4.3 and 4.7 to 4.8), that future work investigate two sets of paired IMUs to investigate if this arrangement would provide bias and offset cancellation, including long term gyroscope drift.

Research Objective Three was to determine the least number of layers a neural network might need to provide accurate roll and pitch data from IMU inputs. For a NARX implementation, a minimum of 3 layers is required and this incurred a similar processing time penalty to an industry standard Kalman filter implementation. (1.5774 seconds NARXNET processing time compared to the Kalman filter's 1.8195 seconds processing time).

While answering the research questions, the thesis also outlines that although the Madgwick filter is the simplest and fastest of all filters evaluated, the experiments have shown it has limitations based on the size of movement patterns (effective sample rate) and the beta values, and the Madgwick filter does not handle low-frequency sample rates as well as the Kalman filter.

The Kalman filter is slightly more accurate than the Madgwick filter, and could be made more accurate with some optimisation of process noise and training trajectory values, but has a longer processing time than the NARXNET three-layer neural network. It is a "middle of the road" approach, and it is clear why this filter is popular in industry, as a Tensorflow-capable IoT processor would not be required to implement this filter, saving money and power consumption. If a selected processor can run the NARX NN model, then the NARX NN model should give superior results, but this claim

needs further investigation.

6.1 Future work.

The experiments undertaken have answered the research questions however improvements and other avenues of research could be implemented.

Future work in this area should consider using two sets of a pair of IMUs where one set is mounted horizontally and the other vertically, to determine if yaw can be measured without the use of a magnetometer. Each IMU in the pair should be mounted inversely to the other, both vertically and laterally. i.e., One should appear upside down compared to the other with X axes reversed. Evaluation should be undertaken to also investigate if biases and offsets would cancel each other.

A track with known angles should be built with a suitable means of detecting chassis movement by visual means, since any real-world vehicle will involve shock absorbers and suspension systems, making determinations of chassis angles by topology alone, difficult. If such a track is not available, a robotic arm capable of both higher speed and the ability to move in a straight line in a forward direction for at least 1m should be implemented to mimic a subset of real-world conditions. It is envisaged that the vehicle/robot speed would also need to be an input to the neural network system.

Future research should be undertaken on a larger vehicle (the size of which needs to be determined primarily by battery capacity, at the present time), to determine what chassis mounting methods best suits the use of magnetometer data.

The IMU data-collection process can be improved by increasing the effective sample rate by using individual serial UART ports per IMU on the chassis computer (Individual SPI ports should be utilised if serial ports are not available) instead of the I^2C bus.

The robotic arm sampling rate needs to be decreased to 500Hz or lower to meet the maximum requirements of the Robotic arm TCP/IP interface.

Robotic arm and IMU synchronisation were based off central IMU timings but individual IMUs could be synchronised to the closest robotic arm timing in an equivalent manner to the configured central IMU process if NTP time synchronisation was implemented to configure both the chassis computer and the data-collection laptop with identical time settings. If a track is used to provide ground truth, a robotic arm is not required, but it is envisaged that the financial and time cost of this will mean that a robotic arm is likely to be utilised instead. Synchronisation can be slightly improved by ensuring

that robotic arm measurements are collected from a dedicated network port, directly connected to the robotic arm.

Future work should investigate the use of a NARX NN model instead of a Kalman filter on a suitable IoT processor.

6.2 Summary

This thesis investigates if a neural network using multiple IMUs would provide more accurate Euler conversions than with just a single IMU. A secondary objective was to examine the minimum number of IMUs that would be required to provide an improvement. A tertiary objective was to determine the minimum layer configuration a neural network requires to provide an improvement.
A robotic arm was used to provide ground truth for aboard comprising of five IMUs, with one mounted centrally at the front of the board and one at each corner (See Figure 3.1)
The calibrated IMUs were compared to discover noise and bias offsets and then pre-programmed robotic arm movements were used to compare (in MATLAB) a Madgwick complementary filter, an indirect Kalman filter and three neural network models (MLP, RBF and NARX).
The single central IMU, the front three IMUs and all IMU measurements were compared to answer the research questions.
It was found that the addition of more IMUs offers no statistically significant benefit to using a single IMU and recommends a three-layer NARX neural network be implemented for this workload. Future research should investigate if two sets of two pairs of IMUs can be configured to provide bias and offset cancellation.

Chapter 7

Appendices

7.1 Appendix 1. Code

All code can be obtained from https://github.com/BratNZ/Thesis.

The names and functions of the various scripts in the github repository are outlined below in Table 7.1.

Table 7.1 Script Names and Functions

Script Name	Script Function
Imudata.py	Raspberry Pi script to capture IMU data and apply calibration offsets before saving to file.
Calibrate.py	Raspberry Pi script that calibrates gyroscope and accelerometers on each IMU.
GetRobotData.py	Ubuntu Windows Services for Linux container script that polls robot arm for position status.
MATLAB_Stationary.txt	List of MATLAB commands to analyse data of stationary vehicle for sensor noise readings
MATLAB_Roll.txt	List of MATLAB commands to analyse data for rolling movement sensitivity
MATLAB_Pitch.txt	List of MATLAB commands to analyse data for pitching movement sensitivity
MATLAB_Varied.txt	List of MATLAB commands to analyse data for varied movements
MATLAB_Magnetometer.txt	List of MATLAB commands to analyse data for varied movements, assisted by magnetometer
MATLAB_Rotation	List of MATLAB commands to analyse data for varied movements while under rotation
MATLAB_Z Axis	List of MATLAB commands to analyse data for varied movements with FL IMU facing down. This detects biases in the Z axis.
MATLAB_XY Axis	List of MATLAB commands to analyse data for varied movements with FL IMU facing down with each rear IMU facing 90 degrees away from the chassis. This detects biases in the X and Y axes.

IMU data capture timings involve multiplexor switching, IMU polling and general script timings. Multiplexer switching time took 515 nanoseconds at the start of the movement set down to 41 milliseconds after 30 seconds of operation. After 1 minute of operation, the multiplexing switching

time was measured at 47 milliseconds. The time taken to read acceleration data was 2202 milliseconds and the time taken to read gyroscope data was 1659 milliseconds with a total time to measure each IMU of 2400 milliseconds. Between each measurement, it took 71 milliseconds (excluding multiplexor switching time) to begin processing the next set of measurements.

The robot arm is sampled at approximately 6403 readings per second which equates to a time of 0.15625 milliseconds per reading.

7.2 Appendix 2. Calibration results.

Table 7.2 IMU Error Coefficients produced from initial calibration (rounded to 8 decimal places)

IMU	Gyro X	Gyro Y	Gyro Z	Accel X	Accel Y	Accel Z
RR	-0.02998955	0.01596670	0.00715132	0.00078995	-0.01058559	-0.02925765
FR	-0.03219639	0.00790274	-7.2477796985e-05	-0.00159294	0.01550039	-0.01817877
CM	0.023254980	0.00284742	-0.00660733	0.01240663	0.0155684	-0.03045218
FL	0.008131502	-0.00708547	0.00312161	-0.00665329	-0.01244544	-0.01997862
RL	-0.00307724	0.00518083	-0.00114486	0.0027521	0.0128008	0.00237755

Figure 7.1 IMU Calibration graphs for central IMU

Figure 7.2 IMU Calibration graphs for Front Left IMU

Figure 7.3 IMU Calibration graphs for Front Right IMU

Figure 7.4 IMU Calibration graphs for Rear Left IMU

Figure 7.5 IMU Calibration graphs for Right Rear IMU

7.3 Appendix 3. Equipment Specifications.

The Raspberry Pi computer selected to capture IMU data is a Raspberry Pi 4b with 4Gb of RAM (Raspberry, n.d.) using Raspbian version 11 (Bullseye) operating system, running on a generic 16Gb MicroSD card, using the ext4 filesystem (mounted with no atime).

Python 3.9.2 is used for data capture on the Raspberry Pi. Additional python modules to be installed are sparkfun_qwiic (i^2c support), sparkfun-qwiic-tca9548a (to drive the multiplexer), board (to simplify addressing) and adafruit-circuitpython-icm20x (to communicate with the ICM-20948 IMUs). Finally, SciPy (apt install python3-scipiy), NumPy and Matplotlib pip libraries are installed as these are used in the calibration script. The maximum I^2C sampling rate of 1Mbs is selected.

All IMUs are comprised of Sparkfun IMU-20948 chipsets (Sparkfun, n.d.). The gyroscope is set to a range of ±250rps, the accelerometer is set to a range of ±2g, and the magnetometer has a fixed range of ±4900µT.

A Sparkfun TCM9548A I^2C multiplexor (required as ICM-20948 IMUs only have two configurable I^2C addresses) connects all IMUs to the Raspberry Pi's I^2C bus on the default I^2C pins of GPIO2 and 3. Ports 0,3,4,5 and 7 are used.

To power the equipment, a 65W, 30Ah USB battery pack is used to power the Raspberry Pi, which powers the sensors and multiplexor.

A Universal Robotics UR5 robotic arm provides the "ground truth" of correct angles that is used to

verify the filter and neural network results. Position status is obtained over TCP/IP calls using a Python ut_rtde module running under an Ubuntu Windows Services for Linux container on a laptop.

The laptop used is an HP Elitebook 850 G8 laptop with 16 Gb of RAM and an i7-1165G7 Quad-core CPU running at 2.8GHz. The operating system installed is Windows 10 Pro 22H2, build 10945.3448, running the Windows Experience Feature Pack version 1000.19044.1000.0. There was some difficulty in installing the ut_rtde robot arm software for Python (wheel dependency issues) on the laptop so an Ubuntu Services for Linux container was run on the Laptop and this container was used to launch the GetRobotData.py program (Appendix 1) which connects to the robot arm via TCP/IP and polls the robot arm for position information. The laptop connects to the Raspberry Pi via ssh to provide an interface to run the imudata.py script.

To process the IMU data, MATLAB 2023a (64bit build 9.14.0.2206163) is implemented on an HP Z230 workstation (quad-core Xeon E3-1270v3@3.50GHz CPU with 32Gb of DDR3 RAM and 1Tb Samsung 860QV0 SSD) running Windows 10 Pro 22H2 for data processing and analysis.

The Madgwick filter selected is the freely available MATLAB implementation provided by the author of the filter. The “Initial release” version of this code was used (dated 28/09/2011). (Madgwick, 2009). An initial Beta gain (ratio of accelerometer to gyro data used) is set to the recommended value of 0.5. The accelerometer values are normalised to the gyroscope values internally within the algorithm. There are separate methods within the code. The first, UpdateIMU, is used for accelerometer and gyroscope data only, and the second, Update, also adds support for magnetometer data. MATLAB wrapper scripts were created to call these functions.

The Kalman filter selected is either the MATLAB-based “imufilter” or the “ahrsfilter” (used with magnetometer readings) with default settings unless specified otherwise. MATLAB wrapper scripts were created to call either function as appropriate.

7.4 Appendix 4. IMU-29048 IMU specifications

Digital Output	I2C SPI
VDD Power Supply / V	1.71 to 3.6
VDDIO I/O Power Supply / V	1.71 to 3.6
Gyro FSR	±250/500/1000/2000
Gyro Sensitivity Error	±1.5%
Gyro Rate Noise	0.015dps/√Hz
Accel FSR	±2/4/8/16
Accel Sensitivity Error	±0.5%
Accel Noise	230µg/√Hz
Compass FSR	±4900µT
Pressure Sensor Relative Accuracy	
Pressure Sensor Noise	

Chapter 8. References

Ab Wahab, M. N., Nefti-Meziani, S., & Atyabi, A. (2015). A comprehensive review of swarm optimization algorithms. *PLoS ONE*, *10*(5). https://doi.org/10.1371/JOURNAL.PONE.0122827

Adafruit. (2022). *Slamtec RPLIDAR A1 - 360 Laser Range Scanner : ID 4010 : $99.95 : Adafruit Industries, Unique & fun DIY electronics and kits*. https://www.adafruit.com/product/4010

Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia-Segui, J., & Watteyne, T. (2017). Understanding the Limits of LoRaWAN. *IEEE Communications Magazine*, *55*(9), 34–40. https://doi.org/10.1109/MCOM.2017.1600613

Ahmed, S., Qiu, B., Ahmad, F., Kong, C. W., & Xin, H. (2021). A State-of-the-Art Analysis of Obstacle Avoidance Methods from the Perspective of an Agricultural Sprayer UAV's Operation Scenario. *Agronomy 2021, Vol. 11, Page 1069*, *11*(6), 1069. https://doi.org/10.3390/AGRONOMY11061069

Akagawa, K. (1996). *US5994699A - Thermal camera for infrared imaging - Google Patents*. https://patents.google.com/patent/US5994699A/en

Al-Sarawi, S., Anbar, M., Alieyan, K., & Alzubaidi, M. (2017). Internet of Things (IoT) communication protocols: Review. *ICIT 2017 - 8th International Conference on Information Technology, Proceedings*, 685–690. https://doi.org/10.1109/ICITECH.2017.8079928

Andreev, S., Galinina, O., Pyattaev, A., Gerasimenko, M., Tirronen, T., Torsner, J., Sachs, J., Dohler, M., & Koucheryavy, Y. (2015). Understanding the IoT connectivity landscape: A contemporary M2M radio technology roadmap. *IEEE Communications Magazine*, *53*(9), 32–40. https://doi.org/10.1109/MCOM.2015.7263370

Android Authority. (n.d.-a). *Image-Stablization-GIF.gif (404×296)*. Retrieved August 17, 2023, from https://www.androidauthority.com/wp-content/uploads/2020/02/Image-Stablization-GIF.gif

Android Authority. (n.d.-b). *Image-Stablization-Perspective.gif (350×261)*. Retrieved August 17, 2023, from https://www.androidauthority.com/wp-content/uploads/2020/02/Image-Stablization-Perspective.gif

Aparna, R., Ruchitha, H. T. S., Pranavi, N., & Keskar, A. G. (2020). IMU based Tracking of a Person using Nonlinear Autoregressive Exogenous(NARX) Algorithm in GPS-denied Areas. *2020 1st IEEE International Conference on Measurement, Instrumentation, Control and Automation, ICMICA*

2020. https://doi.org/10.1109/ICMICA48462.2020.9242732

Arsalan, M., Umair, A., & Kumar Verma, V. (2018). Dash7: Performance. In *IOSR Journal of Electronics and Communication Engineering (IOSRJECE)* (Vol. 2, Issue 5). www.iosrjournals.orgwww.iosrjournals.org

Augustin, A., Yi, J., Clausen, T., & Townsley, W. M. (2016). A study of Lora: Long range & low power networks for the internet of things. *Sensors (Switzerland)*, *16*(9). https://doi.org/10.3390/s16091466

Auysakul, J., Xu, H., & Pooneeth, V. (2018). A Hybrid Motion Estimation for Video Stabilization Based on an IMU Sensor. *Sensors 2018, Vol. 18, Page 2708*, *18*(8), 2708. https://doi.org/10.3390/S18082708

Auysakul, J., Xu, H., & Pooneeth, V. (2019). Video stabilization with a dual system based on an IMU sensor for the mobile robot. *Advances in Intelligent Systems and Computing*, *856*, 618–626. https://doi.org/10.1007/978-3-030-00214-5_78/COVER

Aviation Authority of New Zealand, C. (2015). *Published by the Civil Aviation Authority of New Zealand Unmanned Aircraft Operator Certification*. www.caa.govt.nz

Bailey, D. W., Trotter, M. G., Knight, C. W., & Thomas, M. G. (2018). Use of GPS tracking collars and accelerometers for rangeland livestock production research. *American Society of Animal Science*, 81–88. https://doi.org/10.1093/tas/txx006

Baronti, P Pillai, P Chook, V.W.C Chessa, S Gotta, A Hu, F. (2007). Wireless sensor networks: a survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communications*, *30*(7), 1655–1695.

Battiato, S., Gallo, G., Puglisi, G., & Scellato, S. (2007). SIFT features tracking for video stabilization. *Proceedings - 14th International Conference on Image Analysis and Processing, ICIAP 2007*, 825–830. https://doi.org/10.1109/ICIAP.2007.4362878

Beef & Lamb NZ. (2021). *45Th Edition Farm Facts*. www.beeflambnz.com

Beef and Lamb NZ. (2017). *Guide to New Zealand Cattle Farming*.

Beliveau, A., Spencer, G. T., Thomas, K. A., & Roberson, S. L. (1999). Evaluation of MEMS capacitive accelerometers. *IEEE Design & Test of Computers*, *16*(4), 48–56. https://doi.org/10.1109/54.808209

Bernini, N., Bertozzi, M., Castangia, L., Patander, M., & Sabbatelli, M. (2014). Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey. *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, 873–878. https://doi.org/10.1109/ITSC.2014.6957799

Blanche, J., Chirayil Nandakumar, S., Mitchell, D., Harper, S., Groves, K., West, A., Lennox, B., Watson, S., Flynn, D., & Yamamoto, I. (n.d.). *Millimeter-Wave Sensing for Avoidance of High-Risk Ground Conditions for Mobile Robots*.

Borges De Sousa, J., & Andrade Gonçalves, • G. (2010). *Unmanned vehicles for environmental data collection*. https://doi.org/10.1007/s10098-010-0313-5

Byeon, M., & Yoon, S. W. (2020). Analysis of Automotive Lidar Sensor Model Considering Scattering Effects in Regional Rain Environments. *IEEE Access*, *8*, 102669–102679. https://doi.org/10.1109/ACCESS.2020.2996366

Caruso, M., Sabatini, A. M., Laidig, D., Seel, T., Knaflitz, M., Croce, U. Della, & Cereatti, A. (2021). Analysis of the accuracy of ten algorithms for orientation estimation using inertial and magnetic sensing under optimal conditions: One size does not fit all. *Sensors*, *21*(7). https://doi.org/10.3390/S21072543

Castano, R., Manduchi, R., & Fox, J. (2014). *Classification experiments on real-world texture*. Jpl.Nasa.Gov. https://trs.jpl.nasa.gov/handle/2014/41475

Chang, S., Zhong, Y., Quan, Z., Hong, Y., Zeng, J., & Du, D. (2016). A real-time object tracking and image stabilization system for photographing in vibration environment using OpenTLD algorithm. *Proceedings of IEEE Workshop on Advanced Robotics and Its Social Impacts, ARSO*, *2016-Novem*, 141–145. https://doi.org/10.1109/ARSO.2016.7736271

Chao, H., Gu, Y., Gross, J., Guo, G., Fravolini, M. L., & Napolitano, M. R. (2013). A comparative study of optical flow and traditional sensors in UAV navigation. *Proceedings of the American Control Conference*, 3858–3863. https://doi.org/10.1109/ACC.2013.6580428

Chaudhary, S., Zakhami, S., & Roy, S. D. (2019). Visual Feedback based Trajectory Planning to Pick an Object and Manipulation using Deep learning. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3352593.3352616

Chemhengcharoen, P., Nilsumrit, P., Pongpetrarat, P., & Phanomchoeng, G. (2019). *Development of a Prototype of Autonomous Vehicle for Agriculture Applications*. https://doi.org/10.1145/3330180.3330191

Cheong, M. K., Bahiki, M. R., & Azrad, S. (2016). Development of collision avoidance system for useful UAV applications using image sensors with laser transmitter. *IOP Conference Series: Materials Science and Engineering*, *152*(1), 012026. https://doi.org/10.1088/1757-899X/152/1/012026

Cheong, P. S., Bergs, J., Hawinkel, C., & Famaey, J. (2017). Comparison of LoRaWAN classes and their power consumption. *2017 IEEE Symposium on Communications and Vehicular Technology, SCVT 2017*, *2017-December*, 1–6. https://doi.org/10.1109/SCVT.2017.8240313

Cho, Y., Han, J., Kim, J., Lee, P., & Park, S. B. (2019). Experimental validation of a velocity obstacle based collision avoidance algorithm for unmanned surface vehicles. *IFAC-PapersOnLine*, *52*(21), 329–334. https://doi.org/10.1016/j.ifacol.2019.12.328

Choit, J., Ahnt, S., & Chungt, W. K. (2005). Robust sonar feature detection for the slam of mobile robot. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2083–2088. https://doi.org/10.1109/IROS.2005.1545284

Crane, C. D., Armstrong, D. G., Touchton, R., Galluzzo, T., Solanki, S., Lee, J., Kent, D., Ahmed, M., Montane, R., Ridgeway, S., Velat, S., Garcia, G., Griffis, M., Gray, S., Washburn, J., & Routson, G. (2006). Field report: Team CIMAR's NaviGATOR: An unmanned ground vehicle for the 2005 DARPA Grand Challenge. *Journal of Field Robotics*, *23*(8), 599–623. https://doi.org/10.1002/ROB.20127

Dai, R., Stein, R. B., Andrews, B. J., James, K. B., & Wieler, M. (1996). Application of tilt sensors in functional electrical stimulation. *IEEE Transactions on Rehabilitation Engineering*, *4*(2), 63–72. https://doi.org/10.1109/86.506403

Damlamian, A., & Jaffard, S. (2019). Wavelet methods in mathematical analysis and engineering. *Wavelet Methods in Mathematical Analysis and Engineering*, 1–178. https://doi.org/10.1142/7899

DASH7 Alliance – An open specification. (n.d.). Retrieved January 14, 2021, from https://dash7-alliance.org/

De Sanctis, M., Cianca, E., Araniti, G., Bisio, I., & Prasad, R. (2016). Satellite communications supporting internet of remote things. *IEEE Internet of Things Journal*, *3*(1), 113–123. https://doi.org/10.1109/JIOT.2015.2487046

De Simone, M. C., Rivera, Z. B., & Guida, D. (2018a). Obstacle Avoidance System for Unmanned Ground Vehicles by Using Ultrasonic Sensors. *Machines 2018, Vol. 6, Page 18*, *6*(2), 18. https://doi.org/10.3390/MACHINES6020018

De Simone, M. C., Rivera, Z. B., & Guida, D. (2018b). Obstacle Avoidance System for Unmanned Ground Vehicles by Using Ultrasonic Sensors. *Machines 2018, Vol. 6, Page 18*, *6*(2), 18. https://doi.org/10.3390/MACHINES6020018

Deruyck, M., Vereecken, W., Tanghe, E., Joseph, W., Pickavet, M., Martens, L., & Demeester, P. (2010). Comparison of power consumption of mobile WiMAX, HSPA and LTE access networks. *2010 9th Conference of Telecommunication, Media and Internet, CTTE 2010*. https://doi.org/10.1109/CTTE.2010.5557715

Druzhkov, P. N., & Kustikova, V. D. (2016). A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis 2016 26:1*, *26*(1), 9–15. https://doi.org/10.1134/S1054661816010065

Duroc, Y. (2022). From Identification to Sensing: RFID Is One of the Key Technologies in the IoT Field. *Sensors 2022, Vol. 22, Page 7523*, *22*(19), 7523. https://doi.org/10.3390/S22197523

Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3876 LNCS*, 265–284. https://doi.org/10.1007/11681878_14

Euston, M., Coote, P., Mahony, R., Kim, J., & Hamel, T. (2008). A complementary filter for attitude estimation of a fixed-wing UAV. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 340–345. https://doi.org/10.1109/IROS.2008.4650766

Fan, B., Li, Q., & Liu, T. (2017). How Magnetic Disturbance Influences the Attitude and Heading in Magnetic and Inertial Sensor-Based Orientation Estimation. *Sensors 2018, Vol. 18, Page 76*, *18*(1), 76. https://doi.org/10.3390/S18010076

Fixed link licence | Radio Spectrum Management New Zealand. (n.d.). Retrieved December 7, 2020, from https://www.rsm.govt.nz/licensing/licences-you-must-pay-for/fixed-location-licences/fixed-link-licence/

Foix, S., Alenyà, G., & Torras, C. (2011). Lock-in time-of-flight (ToF) cameras: A survey. *IEEE Sensors Journal*, *11*(9), 1917–1926. https://doi.org/10.1109/JSEN.2010.2101060

Formsma, O., Dijkshoorn, N., Van Noort, S., & Visser, A. (2010). *Realistic Simulation of Laser Range Finder Behavior in a Smoky Environment*. http://www.science.uva.nl/research/isla

Fossum, E. R., & Hondongwa, D. B. (2014). A review of the pinned photodiode for CCD and CMOS

image sensors. *IEEE Journal of the Electron Devices Society*, *2*(3), 33–43. https://doi.org/10.1109/JEDS.2014.2306412

Fpatrocini. (n.d.). *NY fish-eye Free Photo Download | FreeImages*. Retrieved October 3, 2023, from https://www.freeimages.com/photo/ny-fish-eye-1447949

Gaddam, S. C., & Rai, M. K. (2018). A comparative study on various LPWAN and cellular communication technologies for IoT based smart applications. *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research, ICETIETR 2018*. https://doi.org/10.1109/ICETIETR.2018.8529060

Ganesan, P., & Sajiv, G. (2018). A comprehensive study of edge detection for image processing applications. *Proceedings of 2017 International Conference on Innovations in Information, Embedded and Communication Systems, ICIIECS 2017*, *2018-Janua*, 53–54. https://doi.org/10.1109/ICIIECS.2017.8275968

Gao, H., Liu, J., Wei, Q., & Yu, Y. (2016). Mobile Robot Obstacle Avoidance Algorithms Based on Information Fusion of Vision and Sonar Space robot View project Reconfigurable Modular Robot View project Mobile Robot Obstacle Avoidance Algorithms Based on Information Fusion of Vision and Sonar. *Article in International Journal of Future Generation Communication and Networking*, *9*(8), 111–120. https://doi.org/10.14257/ijfgcn.2016.9.8.11

Ghaffari, M., Alhaj Ali, S. M., Murthy, V., Liao, X., Gaylor, J., & Hall, E. L. (2004). Design of an unmanned ground vehicle, bearcat III, theory and practice. *Journal of Robotic Systems*, *21*(9), 471–480. https://doi.org/10.1002/ROB.20027

Gheorghiu, R., & Iordache, V. (2018). Use of Energy Efficient Sensor Networks to Enhance Dynamic Data Gathering Systems: A Comparative Study between Bluetooth and ZigBee. *Sensors*, *18*(6), 1801. https://doi.org/10.3390/s18061801

Gilmore, M. S., Castaño, R., Mann, T., Anderson, R. C., Mjolsness, E. D., Manduchi, R., & Saunders, R. S. (n.d.). *Strategies for autonomous rovers at Mars*.

Gomez, C., Oller, J., & Paradells, J. (2012). Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors (Switzerland)*, *12*(9), 11734–11753. https://doi.org/10.3390/s120911734

Goodin, C., Carruth, D., Doude, M., & Hudson, C. (2019). Predicting the influence of rain on LIDAR in ADAS. *Electronics (Switzerland)*, *8*(1). https://doi.org/10.3390/ELECTRONICS8010089

Hall, C. (2021). *Future batteries, coming soon: Charge in seconds, last months a*. https://www.pocket-lint.com/gadgets/news/130380-future-batteries-coming-soon-charge-in-seconds-last-months-and-power-over-the-air

Haq, I., Anwar, S., Shah, K., Khan, M. T., & Shah, S. A. (2015). Fuzzy Logic Based Edge Detection in Smooth and Noisy Clinical Images. *PLOS ONE*, *10*(9), e0138712. https://doi.org/10.1371/JOURNAL.PONE.0138712

Harindranath, A., & Arora, M. (2018). MEMS IMU sensor orientation algorithms-comparison in a simulation environment. *2018 International Conference on Networking, Embedded and Wireless Systems, ICNEWS 2018 - Proceedings*. https://doi.org/10.1109/ICNEWS.2018.8904029

Hautì, N., Tarel, J.-P., Lavenant, J., Aubert, D., Hautì Ere (b, N., Tarel, J.-P., Lavenant, · J, & Aubert, · D. (2006). Automatic fog detection and estimation of visibility distance through use of an onboard camera. *Machine Vision and Applications 2005 17:1*, *17*(1), 8–20. https://doi.org/10.1007/S00138-005-0011-1

Haxhibeqiri, J., De Poorter, E., Moerman, I., & Hoebeke, J. (2018a). A survey of LoRaWAN for IoT: From technology to application. In *Sensors (Switzerland)* (Vol. 18, Issue 11). MDPI AG. https://doi.org/10.3390/s18113995

Haxhibeqiri, J., De Poorter, E., Moerman, I., & Hoebeke, J. (2018b). A Survey of LoRaWAN for IoT: From Technology to Application. *Sensors 2018, Vol. 18, Page 3995*, *18*(11), 3995. https://doi.org/10.3390/S18113995

Higgins, W. T. (1975). A Comparison of Complementary and Kalman Filtering. *IEEE Transactions on Aerospace and Electronic Systems*, *AES-11*(3), 321–325. https://doi.org/10.1109/TAES.1975.308081

Hoffman, B. D., Baumgartner, E. T., Huntsberger, T. L., & Schenker, P. S. (1999). Improved Rover State Estimation in Challenging Terrain. *Autonomous Robots 1999 6:2*, *6*(2), 113–130. https://doi.org/10.1023/A:1008879310128

Hoffmann, J., Jüngel, M., & Lötzsch, M. (2004). *LNAI 3276 - A Vision Based System for Goal-Directed Obstacle Avoidance*. http://www.aiboteamhumboldt.com

Hornik, K., Stinchcombe, M., & White, H. (1989). Sci-Hub | Multilayer feedforward networks are universal approximators. Neural Networks, 2(5), 359–366 | 10.1016/0893-6080(89)90020-8. *Nerual Networks*, *2*, 359–366. https://sci-hub.se/10.1016/0893-6080(89)90020-8

Hsu, S. C., Liang, S. F., & Lin, C. T. (2005). A robust digital image stabilization technique based on inverse triangle method and background detection. *IEEE Transactions on Consumer Electronics*, *51*(2), 335–345. https://doi.org/10.1109/TCE.2005.1467968

Huang, J., Jiang, S., & Lu, X. (2001). Rain backscattering properties and effects on the radar performance at MM wave band. *International Journal of Infrared and Millimeter Waves*, *22*(6), 917–922. https://doi.org/10.1023/A:1014922632540

Huang, S., Qin, H., Zhou -, Z., Liu, Z., Liu, X., Duan, G., -, A., Li, X., Liu, W., Pan, Y., Wang, Q., Meng, Z., & Liu, H. (2020). Review on Application of Binocular Vision Technology in Field Obstacle Detection You may also like Vision measurement for flat parts based on local line-angle contour segmentation Precise pose and radius estimation of circular target based on binocular vi. *IOP Conference Series: Materials Science and Engineering*. https://doi.org/10.1088/1757-899X/806/1/012025

Irsyad Bin Sahalan, M. (2018). *DEVELOPMENT OF OBSTACLE AVOIDANCE TECHNIQUE FOR MULTI AGENT AUTONOMOUS SURFACE VESSEL USING OPTIMAL RECIPROCAL COLLISION AVOIDANCE*.

Jawad, H. M., Nordin, R., Gharghan, S. K., Jawad, A. M., & Ismail, M. (2017). Energy-efficient wireless sensor networks for precision agriculture: A review. In *Sensors (Switzerland)* (Vol. 17, Issue 8). MDPI AG. https://doi.org/10.3390/s17081781

Jia, X., Feng, Q., Fan, T., & Lei, Q. (2012). RFID technology and its applications in Internet of Things (IoT). *2012 2nd International Conference on Consumer Electronics, Communications and Networks, CECNet 2012 - Proceedings*, 1282–1285. https://doi.org/10.1109/CECNET.2012.6201508

Jiang, X., Zong, Y., Liu, Z., Wang, X., & Chen, Z. (2010). Research on combinational image stabilization technology based on MEMS IMU. *Optoelectronic Imaging and Multimedia Technology*, *7850*, 785002. https://doi.org/10.1117/12.868889

Jiménez, F., Naranjo, J. E., Gómez, O., & Anaya, J. J. (2014). Vehicle tracking for an evasive manoeuvres assistant using low-cost ultrasonic sensors. *Sensors (Switzerland)*, *14*(12), 22689–22705. https://doi.org/10.3390/S141222689

Johnstone-Wallace, D. B., & Kennedy, K. (1944). Grazing management practices and their relationship to the behaviour and grazing habits of cattle. *The Journal of Agricultural Science*, *34*(4), 190–197. https://doi.org/10.1017/S0021859600023649

Kapoor, R., Ramasamy, S., Gardi, A., Van Schyndel, R., & Sabatini, R. (2018). Acoustic sensors for air

and surface navigation applications. *Sensors (Switzerland)*, *18*(2). https://doi.org/10.3390/S18020499

Karpenko, A., Jacobs, D., Baek, J., & Levoy, M. (2011). *Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes*.

Karuppuswamy, J. (2000). *Detection and Avoidance of Simulated Potholes in Autonomous Vehicles in an Unstructured Environment*.

Katiyar, S., & Dutta, A. (n.d.). PSO Based Path Planning and Dynamic Obstacle Avoidance in CG Space of a 10 DOF Rover; PSO Based Path Planning and Dynamic Obstacle Avoidance in CG Space of a 10 DOF Rover. *Advances in Robotics - 5th International Conference of The Robotics Society*. https://doi.org/10.1145/3478586

Katiyar, S., & Dutta, A. (2019). Path Planning and Obstacle Avoidance in CG Space of a 10 DOF Rover using RRT. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3352593.3352615

Kawai, S., Takeuchi, K., Shibata, K., & Horita, Y. (2014). A Smart Method to Distinguish Road Surface Conditions at Night-time using a Car-Mounted Camera. *IEEJ Transactions on Electronics, Information and Systems*, *134*(6), 878–884. https://doi.org/10.1541/IEEJEISS.134.878

Khan, M. N., & Ahmed, M. M. (2021). Weather and surface condition detection based on road-side webcams: Application of pre-trained Convolutional Neural Network. *International Journal of Transportation Science and Technology*. https://doi.org/10.1016/J.IJTST.2021.06.003

KISHORE, P. S. V., P. SURESH, K., SMRUTI, D., & RAMESH, K. (2018). SOLAR POWERED OBSTACLE AVOIDING LAWN MOWER. *I-Manager's Journal on Electrical Engineering*, *12*(2), 1. https://doi.org/10.26634/JEE.12.2.14169

Kleeman, L., & Kuc, R. (2008). Sonar Sensing. *Springer Handbook of Robotics*, *January 2008*. https://doi.org/10.1007/978-3-540-30301-5

Koenig, S., Likhachev, M., & Furcy, D. (2004). Lifelong Planning A∗. *Artificial Intelligence*, *155*(1–2), 93–146. https://doi.org/10.1016/J.ARTINT.2003.12.001

Krotkov, E., Simmons, R., Cozman, F., & Koenig, S. (n.d.). *Safeguarded Teleoperation for Lunar Rovers: From Human Factors to Field Trials*.

Kusuma Arbawa, Y., Utaminingrum, F., Setiawan, E., sitasi, C., Arbawa, Y. K., Utaminingrum, F., &

Setiawan, E. (2021). Three combination value of extraction features on GLCM for detecting pothole and asphalt road. *Jurnal Teknologi Dan Sistem Komputer*, *9*(1), 64–69. https://doi.org/10.14710/JTSISKOM.2020.13828

Kwaśniewski, K. K., & Gosiewski, Z. (2018). Genetic algorithm for mobile robot route planning with obstacle avoidance. *Acta Mechanica et Automatica*, *12*(2), 151–159. https://doi.org/10.2478/AMA-2018-0024

La Rosa, F., Celvisia Virzì, M., Bonaccorso, F., & Branciforte STMicroelectronics, M. (n.d.). *Optical Image Stabilization (OIS)*.

Landaluce, H., Arjona, L., Perallos, A., Falcone, F., Angulo, I., & Muralter, F. (2020). A Review of IoT Sensing Applications and Challenges Using RFID and Wireless Sensor Networks. *Sensors 2020, Vol. 20, Page 2495*, *20*(9), 2495. https://doi.org/10.3390/S20092495

Lavric, A., & Popa, V. (2018). Performance Evaluation of LoRaWAN Communication Scalability in Large-Scale Wireless Sensor Networks. *Wireless Communications and Mobile Computing*, *2018*. https://doi.org/10.1155/2018/6730719

Lebakula, V., Tang, B., Goodin, C., & Bethel, C. L. (2021). Shape Estimation of Negative Obstacles for Autonomous Navigation. *IEEE International Conference on Intelligent Robots and Systems*, 4525–4531. https://doi.org/10.1109/IROS51168.2021.9636250

Lecun, Y., Muller, U., Ben, J., Cosatto, E., & Flepp, B. (n.d.). *Off-Road Obstacle Avoidance through End-to-End Learning*. Retrieved June 6, 2022, from http://yann.lecun.com

Li, C., Peng, Z., Huang, T. Y., Fan, T., Wang, F. K., Horng, T. S., Munoz-Ferreras, J. M., Gomez-Garcia, R., Ran, L., & Lin, J. (2017). A Review on Recent Progress of Portable Short-Range Noncontact Microwave Radar Systems. *IEEE Transactions on Microwave Theory and Techniques*, *65*(5), 1692–1706. https://doi.org/10.1109/TMTT.2017.2650911

Li, W., Tan, M., Wang, L., & Wang, Q. (2019). A cubic spline method combing improved particle swarm optimization for robot path planning in dynamic uncertain environment. *International Journal of Advanced Robotic Systems*, *16*(6). https://doi.org/10.1177/1729881419891661

Ljubičić, R., Strelnikova, D., Perks, M. T., Eltner, A., Peña-Haro, S., Pizarro, A., Fortunato, S., Sasso, D., Scherling, U., Vuono, P., & Manfreda, S. (2021). A comparison of tools and techniques for stabilising UAS imagery for surface flow observations. *Hydrology and Earth System Sciences*, *Discussion*. https://doi.org/10.5194/hess-2021-112

Lu, L., Yunda, A., Carrio, A., & Campoy, P. (2020). Robust autonomous flight in cluttered environment using a depth sensor. *International Journal of Micro Air Vehicles*, *12*. https://doi.org/10.1177/1756829320924528

Lv, J., Qu, C., Du, S., Zhao, X., Yin, P., Zhao, N., Qu, S., Lv, J., Qu, C., Du, S., Zhao, X., Yin, P., Zhao, N., & Qu, S. (2021). Research on obstacle avoidance algorithm for unmanned ground vehicle based on multi-sensor information fusion. *Mathematical Biosciences and Engineering 2021 2:1022*, *18*(2), 1022–1039. https://doi.org/10.3934/MBE.2021055

Lwowski, J., Sun, L., Mexquitic-Saavedra, R., Sharma, R., & Pack, D. (2014). A Reactive bearing angle only obstacle avoidance technique for unmanned ground vehicles. *International Conference of Control, Dynamic Systems, and Robotics*. https://doi.org/10.11159/JACR.2014.004

Madgwick, S. O. H. (2009). *Madgwick Algorithm*. https://x-io.co.uk/downloads/madgwick_algorithm_matlab.zip

Madgwick, S. O. H. (2010). *An efficient orientation filter for inertial and inertial/magnetic sensor arrays | Resourcium*. https://resourcium.org/resource/efficient-orientation-filter-inertial-and-inertialmagnetic-sensor-arrays

Madgwick, S. O. H., Harrison, A. J. L., Sharkey, P. M., Vaidyanathan, R., & Harwin, W. S. (2013). Measuring motion with kinematically redundant accelerometer arrays: Theory, simulation and implementation. *Mechatronics*, *23*(5), 518–529. https://doi.org/10.1016/J.MECHATRONICS.2013.04.003

Madgwick, S. O. H., Harrison, A. J. L., & Vaidyanathan, R. (2011). Estimation of IMU and MARG orientation using a gradient descent algorithm. *IEEE International Conference on Rehabilitation Robotics*, 1–7. https://doi.org/10.1109/ICORR.2011.5975346

Madgwick, S. O. H., Wilson, S., Turk, R., Burridge, J., Kapatos, C., & Vaidyanathan, R. (2020). An Extended Complementary Filter for Full-Body MARG Orientation Estimation. *IEEE/ASME TRANSACTIONS ON MECHATRONICS*, *25*(4), 2054–2064. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9103115

Mahony, R., Hamel, T., & Pflimlin, J. M. (2005). Complementary filter design on the special orthogonal group SO(3). *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, *2005*, 1477–1484. https://doi.org/10.1109/CDC.2005.1582367

Mahony, R., Hamel, T., & Pflimlin, J. M. (2008). Nonlinear complementary filters on the special

orthogonal group. *IEEE Transactions on Automatic Control*, *53*(5), 1203–1218. https://doi.org/10.1109/TAC.2008.923738

Mai, Y., Zhao, H., & Guo, S. (2012). The analysis of image stabilization technology based on small-UAV airborne video. *Proceedings - 2012 International Conference on Computer Science and Electronics Engineering, ICCSEE 2012*, *3*, 586–589. https://doi.org/10.1109/ICCSEE.2012.77

Manderson, A., & Hunt, C. (2013). INTRODUCING THE AGRI-ROVER: AN AUTONOMOUS ON-THE-GO SENSING ROVER FOR SCIENCE AND FARMING. *Accurate and Efficient Use of Nutrients on Farms. Occasional Report No. 26*.

Manduchi, R., Castano, A., Talukder, A., & Matthies, L. (2005). Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation. *Autonomous Robots 2005 18:1*, *18*(1), 81–102. https://doi.org/10.1023/B:AURO.0000047286.62481.1D

Mannar, S., Thummalapeta, M., Saksena, S. K., & Omkar, S. N. (2018). *Vision-based Control for Aerial Obstacle Avoidance in Forest Environments*. *51*(1), 480–485. https://doi.org/10.1016/J.IFACOL.2018.05.081

Marins, J. L., Yun, X., Bachmann, E. R., McGhee, R. B., & Zyda, M. J. (2001). An extended Kalman filter for quaternion-based orientation estimation using MARG sensors. *IEEE International Conference on Intelligent Robots and Systems*, *4*, 2003–2011. https://doi.org/10.1109/IROS.2001.976367

Martina, N., Hicksona, R., Draganovab, I., Horneb, D., Kenyona, P., & Morris, S. (2015). *Walking distance and energy expenditure of beef cows grazing on hill country in winter*. Proceedings of the New Zealand Society of Animal Production. http://www.sciquest.org.nz/elibrary/download/142531/BRIEF+COMMUNICATION%3A+Walking+distance+and+energy+expenditure+of+beef+cows+grazing+on+hill+country+in+winter?

Masehian, E., & Katebi, Y. (2014). Sensor-based motion planning of wheeled mobile robots in unknown dynamic environments. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *74*(3–4), 893–914. https://doi.org/10.1007/S10846-013-9837-3

Matlab. (2023). *Matlab PolyFit function*. https://au.mathworks.com/help/matlab/ref/polyfit.html

Matthies, L., & Rankin, A. (2003). Negative Obstacle Detection by Thermal Signature. *IEEE International Conference on Intelligent Robots and Systems*, *1*, 906–913. https://doi.org/10.1109/IROS.2003.1250744

Matthies, Larry, Kelly, A., Litwin, T., & Tharp, G. (1995). Obstacle detection for unmanned ground

vehicles: A progress report. *Intelligent Vehicles Symposium, Proceedings*, 66–71. https://doi.org/10.1007/978-1-4471-1021-7_52

Michels, J. (2005). High Speed Obstacle Avoidance using Monocolar Vision and Reinforcement Learning. *Machine Learning*, 1–8.

Mingkhwan, E., & Khawsuk, W. (2017). Digital image stabilization technique for fixed camera on small size drone. *Proceedings - ACDT 2017: 3rd Asian Conference on Defence Technology: Advance Research Collaboration on Defence Technology*, 12–19. https://doi.org/10.1109/ACDT.2017.7886149

Moeller, R. (2020). *GPS-Guided Autonomous Robot with Obstacle Avoidance and Path Optimization Agricultural Robot-Potato Virus Y View project GPS-Guided Autonomous Robot with Obstacle Avoidance and Path Optimization*. https://www.researchgate.net/publication/342916625

Mohammed, A. S., Amamou, A., Ayevide, F. K., Kelouwani, S., Agbossou, K., & Zioui, N. (2020). The Perception System of Intelligent Ground Vehicles in All Weather Conditions: A Systematic Literature Review. *Sensors 2020, Vol. 20, Page 6532*, *20*(22), 6532. https://doi.org/10.3390/S20226532

Mohanty, P. K., & Dewang, H. S. (2021). A smart path planner for wheeled mobile robots using adaptive particle swarm optimization. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *43*(2). https://doi.org/10.1007/S40430-021-02827-7

Moravec, H. P. (n.d.). *Rover Visual Obstacle Avoidance*.

Morimoto, C., & Chellappa, R. (1998). Evaluation of image stabilization algorithms. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, *5*, 2789–2792. https://doi.org/10.1109/ICASSP.1998.678102

Mubarak, M. N. (2013). *Degree Programme in Machine Automation Mubarak, Muhammad Nauman: Outdoor Obstacle Detection using Ultrasonic sensors for an autonomous vehicle ensuring safe operations*.

Mundt, M., Johnson, W. R., Potthast, W., Markert, B., Mian, A., & Alderson, J. (2021). A Comparison of Three Neural Network Approaches for Estimating Joint Angles and Moments from Inertial Measurement Units. *Sensors 2021, Vol. 21, Page 4535*, *21*(13), 4535. https://doi.org/10.3390/S21134535

Nadimi, E. S., Jørgensen, R. N., Blanes-Vidal, V., & Christensen, S. (2012). Monitoring and classifying

animal behavior using ZigBee-based mobile ad hoc wireless sensor networks and artificial neural networks. *Computers and Electronics in Agriculture*, *82*, 44–54. https://doi.org/10.1016/j.compag.2011.12.008

New Zealand IoT Alliance. (2019). *IoT Spectrum in New Zealand Spectrum available in New Zealand for IoT Systems*. http://www.iot.org.au/wp/wp-content/uploads/2016/12/IoTSpectrumFactSheet.pdf

Nguyen-Huu, P.-N., & Titus, J. (2009). *Reliability and Failure in Unmanned Ground Vehicle (UGV)*. http://grrc.engin.umich.edu/

Nguyen Viet, C., & Marshall, I. (n.d.). *VISION-BASED OBSTACLE AVOIDANCE FOR A SMALL, LOW-COST ROBOT*.

Noori, K. K., Kumar Mishra, M., & Abdullah, Z. (n.d.). *VISION-BASED OBSTACLE AVOIDANCE ON QUADCOPTER*. Retrieved June 4, 2022, from www.tjprc.org

Norouzian, F., Marchetti, E., Hoare, E., Gashinova, M., Constantinou, C., Gardner, P., & Cherniakov, M. (2019). Experimental study on low-THz automotive radar signal attenuation during snowfall. *IET Radar, Sonar and Navigation*, *13*(9), 1421–1427. https://doi.org/10.1049/IET-RSN.2018.5644

NZ Government. (2018). *National Animal Identification and Tracing (NAIT) programme | NZ Government*. https://www.mpi.govt.nz/animals/national-animal-identification-tracing-nait-programme/

Odeh, S., & Faqeh, R. (2009). *Vision-Based Obstacle Avoidance of Mobile Robot UsingQuantized Spatial Model Formal Analysis of Attestation in Trusted Execution Environments View project*. https://doi.org/10.3844/ajeassp.2009.611.619

Odelga, M., Kochanek, N., & Bulthoff, H. H. (2017). Efficient real-time video stabilization for UAVs using only IMU data. *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems, RED-UAS 2017*, 210–215. https://doi.org/10.1109/RED-UAS.2017.8101668

Ofir, N., Galun, M., Nadler, B., & Basri, R. (n.d.). *Fast Detection of Curved Edges at Low SNR*.

OneNZ. (2023a). *OneNZ and Starlink*. https://one.nz/why-choose-us/spacex/

OneNZ. (2023b). *OneNZ Network coverage*. https://one.nz/network/coverage/

Parhi, D. R., & Mohanty, P. K. (2016). IWO-based adaptive neuro-fuzzy controller for mobile robot

navigation in cluttered environments. *International Journal of Advanced Manufacturing Technology*, *83*(9–12), 1607–1625. https://doi.org/10.1007/S00170-015-7512-5

Petterson, T. C. (2020). *Development of an electric vehicle for autonomous use on a New Zealand dairy farm*.

Qureshi, U., & Golnaraghi, F. (2017). An Algorithm for the In-Field Calibration of a MEMS IMU. *IEEE Sensors Journal*, *17*(22), 7479–7486. https://doi.org/10.1109/JSEN.2017.2751572

Rajavarshini, R., Shruthi, S., Mahanth, P., Kumar, B. C., & Suyampulingam, A. (2021). Comparative analysis of image processing techniques for obstacle avoidance and path deduction. *Journal of Physics, Conference Series*, *2070*(1), 1–10. https://doi.org/10.1088/1742-6596/2070/1/012121

Ramseyer, A., Boissy, A., Thierry, B., & Dumont, B. (2009). Individual and social determinants of spontaneous group movements in cattle and sheep. *Animal*, *3*(9), 1319–1326. https://doi.org/10.1017/S1751731109004790

Raspberry. (n.d.). *Buy a Raspberry Pi 4 Model B – Raspberry Pi*. Retrieved August 12, 2023, from https://www.raspberrypi.com/products/raspberry-pi-4-model-b/?variant=raspberry-pi-4-model-b-4gb

Ren, K., Karlsson, J., Liuska, M., Hartikainen, M., Hansen, I., & Jørgensen, G. H. M. (2020). A sensor-fusion-system for tracking sheep location and behaviour. *International Journal of Distributed Sensor Networks*, *16*(5). https://doi.org/10.1177/1550147720921776

Rodriguez F., S. A., Frémont, V., & Bonnifait, P. (2008). Extrinsic calibration between a multi-layer lidar and a camera. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 214–219. https://doi.org/10.1109/MFI.2008.4648067

Ropert-Coudert, Y., & Wilson, R. (2005). Trends and perspectives in animal-attached remote sensing. *Front Ecol Environ*, *3*, 437–444. https://doi.org/10.1890/1540-9295(2005)003[0437:tapiar]2.0.co

Rosique, F., Navarro, P. J., Fernández, C., & Padilla, A. (2019). A systematic review of perception system and simulators for autonomous vehicles research. *Sensors (Switzerland)*, *19*(3). https://doi.org/10.3390/S19030648

Russell, G., & Fausett, L. V. (1996). Comparison of function approximation with sigmoid and radial basis function networks. *Proceedings of SPIE*, *2760*(1), 61–72. https://doi.org/10.1117/12.235903

Ryu, Y. G., Roh, H. C., & Chung, M. J. (2010). Video stabilization for robot eye using IMU-aided feature tracker. *ICCAS 2010 - International Conference on Control, Automation and Systems*, 1875–1878. https://doi.org/10.1109/ICCAS.2010.5670177

Sabatini, A. M. (2006). Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing. *IEEE Transactions on Biomedical Engineering*, *53*(7), 1346–1356. https://doi.org/10.1109/TBME.2006.875664

Samarasinghe, S. (2006). *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition*. Auerbach Publications. https://www.amazon.com/Neural-Networks-Applied-Sciences-Engineering-ebook/dp/B008GXJR9S

Schäfer, B. H., Proetzsch, M., & Berns, K. (2005). *Stereo-Vision-Based Obstacle Avoidance in Rough Outdoor Terrain*. www.deere.com

Schneider, R., & Wenger, J. (2003). High resolution radar for automobile applications. *Advances in Radio Science*, *1*, 105–111. https://doi.org/10.5194/ARS-1-105-2003

Schwertfeger, S., Birk, A., & Bülow, H. (2011). Using iFMI spectral registration for video stabilization and motion detection by an Unmanned Aerial Vehicle (UAV). *9th IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2011*, 61–67. https://doi.org/10.1109/SSRR.2011.6106770

Sethi, P., & Sarangi, S. R. (2017). Internet of Things: Architectures, Protocols, and Applications. *Journal of Electrical and Computer Engineering*, *2017*, 1–25. https://doi.org/10.1155/2017/9324035

Shanmuga Sundaram, B. (2016). A quantitative analysis of 802.11ah wireless standard. *International Journal of Latest Research in Engineering and Technology*, *2*.

Sharma, P. S., & Chitaliya, N. G. (2007). Obstacle Avoidance Using Stereo Vision: A Survey. *International Journal of Innovative Research in Computer and Communication Engineering (An ISO*, *3297*(1). https://doi.org/10.15680/ijircce.2015.0301005

Sharma, S. U., & Shah, D. J. (2017). A Practical Animal Detection and Collision Avoidance System Using Computer Vision Technique. *IEEE Access*, *5*, 347–358. https://doi.org/10.1109/ACCESS.2016.2642981

Shen, H., Pan, Q., Cheng, Y., & Yu, Y. (2009). Fast video stabilization algorithm for UAV. *Proceedings - 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009*,

4, 542–546. https://doi.org/10.1109/ICICISYS.2009.5357609

Shene, T. N., Sridharan, K., & Sudha, N. (2016). Real-Time SURF-Based Video Stabilization System for an FPGA-Driven Mobile Robot. *IEEE Transactions on Industrial Electronics*, *63*(8), 5012–5021. https://doi.org/10.1109/TIE.2016.2551684

Shing, L. K., Lwin, K. Z., Sein, K. K., Chan, Y., & Thar, W. W. (2008). *Real-Time Obstacle Avoidance and Fuzzy Dynamic Steering Control for an Unmanned Ground Vehicle*. 978–983.

Short Range Devices GURL | Radio Spectrum Management New Zealand. (n.d.). Retrieved December 10, 2020, from https://www.rsm.govt.nz/licensing/frequencies-for-anyone/short-range-devices-gurl

Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots, Second Edition | The MIT Press*. https://mitpress.mit.edu/books/introduction-autonomous-mobile-robots-second-edition

SigFox. (2023a). *Coverage - Sigfox 0G Technology*. https://www.sigfox.com/coverage/

SigFox. (2023b). *What is Sigfox? - Sigfox 0G Technology*. https://www.sigfox.com/what-is-sigfox/

Simmons, R., Henriksen, L., Chrisman, L., & Whelan, G. (n.d.). *Obstacle Avoidance and Safeguarding for a Lunar Rover*.

Šljivo, A., Kerkhove, D., Tian, L., Famaey, J., Munteanu, A., Moerman, I., Hoebeke, J., & De Poorter, E. (2018). Performance evaluation of IEEE 802.11ah networks with high-throughput bidirectional traffic. *Sensors (Switzerland)*, *18*(2), 1–28. https://doi.org/10.3390/s18020325

Spark. (2021). *Spark Network Coverage*.

Spark NZ. (2023). *Spark NZ network coverage*. https://www.spark.co.nz/shop/mobile/network.html

Sparkfun. (n.d.). *SparkFun 9DoF IMU Breakout - ICM-20948 (Qwiic) - SEN-15335 - SparkFun Electronics*. Retrieved August 12, 2023, from https://www.sparkfun.com/products/15335

Spenko, M., Kuroda, Y., Dubowsky, S., & Iagnemma, K. (2006). Hazard avoidance for high-speed mobile robots in rough terrain. *Journal of Field Robotics*, *23*(5), 311–331. https://doi.org/10.1002/ROB.20118

Sreenivasulu, D., & Deka, P. C. (2011). A comparative study on RBF and NARX based methods for forecasting of groundwater level. *International Journal of Earth Sciences and Engineering*, *4*,

743–756. https://www.researchgate.net/publication/286298837_A_comparative_study_on_RBF_and_NARX_based_methods_for_forecasting_of_groundwater_level

Starlink. (2022). *Starlink satellite coverage*. https://satellitemap.space

Starlink. (2023). *Everything you need to know about Starlink in New Zealand*. https://starlink.co.nz/starlink-new-zealand/

Sugihara, R., & Gupta, R. K. (2011). Path Planning of Data Mules in Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, *8*(1), 27. https://doi.org/10.1145/1993042.1993043

Suh, Y. S. (2019). Laser Sensors for Displacement, Distance and Position. *Sensors (Basel, Switzerland)*, *19*(8). https://doi.org/10.3390/S19081924

Sulaimon Alli, K., Oluwafemi Onibonoje, M., Oluwole, A. S., Adegoke Ogunlade, M., Mmonyi, A. C., Ayamolowo, O., & Olushola Dada, S. (2018). *DEVELOPMENT OF AN ARDUINO-BASED OBSTACLE AVOIDANCE ROBOTIC SYSTEM FOR AN UNMANNED VEHICLE*. *13*(3). www.arpnjournals.com

Sun, J., Rehg, J. M., & Bobick, A. (n.d.). *Learning for Ground Robot Navigation with Autonomous Data Collection*.

Swain, D., Gregg, D., Bishop-Hurley, G., Trotter, M., & Petty, S. (2013). On property benefits of Precision Livestock Management. In *Meat & Livestock Australia* (Vol. 364, Issue July).

Tang, P., Zhang, R., Liu, D., Huang, L., Liu, G., & Deng, T. (2015). Local reactive obstacle avoidance approach for high-speed unmanned surface vehicle. *Ocean Engineering*, *106*, 128–140. https://doi.org/10.1016/J.OCEANENG.2015.06.055

Taylor, L., Miller, E., & Kaufman, K. R. (2017). Static and dynamic validation of inertial measurement units. *Gait & Posture*, *57*, 80–84. https://doi.org/10.1016/J.GAITPOST.2017.05.026

Team, E. (2019). *Bluetooth 5 range put to the test*. https://www.eedesignit.com/bluetooth-5-range-put-to-the-test/

Thillainayagi, R., & Senthil Kumar, K. (2017). Video stabilization technique for thermal infrared Aerial surveillance. *Proceedings of 2016 Online International Conference on Green Engineering and Technologies, IC-GET 2016*. https://doi.org/10.1109/GET.2016.7916630

Tian, L., Santi, S., Seferagić, A., Lan, J., & Famaey, J. (2021). Wi-Fi HaLow for the Internet of Things: An up-to-date survey on IEEE 802.11ah research. *Journal of Network and Computer Applications*,

182, 103036. https://doi.org/10.1016/J.JNCA.2021.103036

Tran, T.-Y., Houre, E., & Clarke, N. (2015). *Wading depth estimation for a vehicle*. United States Patent. https://www.freepatentsonline.com/9026310.pdf

van Hecke, K., de Croon, G., van der Maaten, L., Hennes, D., & Izzo, D. (2018). Persistent self-supervised learning: From stereo to monocular vision for obstacle avoidance: *Https://Doi.Org/10.1177/1756829318756355*, *10*(2), 186–206. https://doi.org/10.1177/1756829318756355

Waitzman, D. (1990). *RFC 1149*. https://datatracker.ietf.org/doc/html/rfc1149

Wallace, A. M., Halimi, A., & Buller, G. S. (2020). Full Waveform LiDAR for Adverse Weather Conditions. *IEEE Transactions on Vehicular Technology*, *69*(7), 7064–7077. https://doi.org/10.1109/TVT.2020.2989148

Wang, J., Song, Q., Jiang, Z., & Zhou, Z. (2016). A novel InSAR based off-road positive and negative obstacle detection technique for unmanned ground vehicle. *International Geoscience and Remote Sensing Symposium (IGARSS)*, *2016-November*, 1174–1177. https://doi.org/10.1109/IGARSS.2016.7729297

Wang, L., Zhao, H., Guo, S., Mai, Y., & Liu, S. (2012). The adaptive compensation algorithm for small UAV image stabilization. *2012 IEEE International Geoscience and Remote Sensing Symposium*, 4391–4394. https://doi.org/10.1109/IGARSS.2012.6350400

Wang, S., Wang, L., He, X., & Cao, Y. (2021). A Monocular Vision Obstacle Avoidance Method Applied to Indoor Tracking Robot. *Drones 2021, Vol. 5, Page 105*, *5*(4), 105. https://doi.org/10.3390/DRONES5040105

Wangchuk, K., Wangdi, J., & Mindu, M. (2017). Comparison and reliability of techniques to estimate live cattle body weight. *Http://Www.Tandfonline.Com/Action/JournalInformation?Show=aimsScope&journalCode=taar20#.VsXoziCLRhE*, *46*(1), 349–352. https://doi.org/10.1080/09712119.2017.1302876

Weber, D., Guhmann, C., & Seel, T. (2020). Neural networks versus conventional filters for inertial-sensor-based attitude estimation. *Proceedings of 2020 23rd International Conference on Information Fusion, FUSION 2020*. https://doi.org/10.23919/FUSION45008.2020.9190634

What is LoRaWAN® | LoRa Alliance®. (n.d.). Retrieved November 18, 2020, from https://lora-alliance.org/resource-hub/what-lorawanr

Williams, L. R., Fox, D. R., Bishop-Hurley, G. J., & Swain, D. L. (2019). Use of radio frequency identification (RFID) technology to record grazing beef cattle water point use. *Computers and Electronics in Agriculture*, *156*, 193–202. https://doi.org/10.1016/j.compag.2018.11.025

Wiriyaprasat, K., & Ruchanurucks, M. (2015). Realtime VDO stabilizer for small UAVs using a modified homography method. *Proceedings 2015 International Conference on Science and Technology, TICST 2015*, 40–43. https://doi.org/10.1109/TICST.2015.7369337

Wong, D. C., Bui, K., Nguyen, L. H., Smith, G., & Ton, T. T. (2003). Integration of differential global positioning system with ultrawideband synthetic aperture radar for forward imaging. *Https://Doi.Org/10.1117/12.488581*, *5083*, 529–538. https://doi.org/10.1117/12.488581

X-IO. (2008). *GitHub - xioTechnologies/Fusion*. https://github.com/xioTechnologies/Fusion

Xu, Z., Sun, C., Ji, T., Manton, J. H., & Shieh, W. (2019). Computational complexity comparison of feedforward/radial basis function/recurrent neural network-based equalizer for a 50-Gb/s PAM4 direct-detection optical link. *Optics Express*, *27*(25), 36953. https://doi.org/10.1364/OE.27.036953

Yamauchi, B. (2006). Autonomous urban reconnaissance using man-portable UGVs. *Https://Doi.Org/10.1117/12.660435*, *6230*, 261–271. https://doi.org/10.1117/12.660435

Yamauchi, B. (2008). All-weather perception for small autonomous UGVs. *Https://Doi.Org/10.1117/12.776792*, *6962*, 28–37. https://doi.org/10.1117/12.776792

Yan, W. Y., Shaker, A., & El-Ashmawy, N. (2015). Urban land cover classification using airborne LiDAR data: A review. *Remote Sensing of Environment*, *158*, 295–310. https://doi.org/10.1016/J.RSE.2014.11.001

Yean, S., Lee, B. S., Yeo, C. K., Vun, C. H., & Oh, H. L. (2018). Smartphone Orientation Estimation Algorithm Combining Kalman Filter with Gradient Descent. *IEEE Journal of Biomedical and Health Informatics*, *22*(5), 1421–1433. https://doi.org/10.1109/JBHI.2017.2780879

Yibin, L., Caihong, L., & Zijian, Z. (2006). Q-learning based method of adaptive path planning for mobile robot. *Proceedings of IEEE ICIA 2006 - 2006 IEEE International Conference on Information Acquisition*, 983–987. https://doi.org/10.1109/ICIA.2006.305871

Yoo, T. S., Hong, S. K., Yoon, H. M., & Park, S. (2011). Gain-Scheduled Complementary Filter Design for a MEMS Based Attitude and Heading Reference System. *Sensors 2011, Vol. 11, Pages 3816-3830*, *11*(4), 3816–3830. https://doi.org/10.3390/S110403816

Yuan, Q., Asadi, E., Lu, Q., Yang, G., & Chen, I. M. (2019). Uncertainty-Based IMU Orientation Tracking Algorithm for Dynamic Motions. *IEEE/ASME Transactions on Mechatronics*, *24*(2), 872–882. https://doi.org/10.1109/TMECH.2019.2892069

Zeng, Y., Yan, E., Li, C., & Li, Y. (2008). Application of multivariable time series based on RBF neural network in prediction of landslide displacement. *Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS 2008*, 2707–2712. https://doi.org/10.1109/ICYCS.2008.163

Zhang, J., Huang, M., Jin, X., & Li, X. (2017). A real-time Chinese traffic sign detection algorithm based on modified YOLOv2. *Algorithms*, *10*(4). https://doi.org/10.3390/A10040127

Zhang, M., Zhang, M., Chen, Y., & Li, M. (2021). *IMU Data Processing For Inertial Aided Navigation: A Recurrent Neural Network Based Approach*.

Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE Multimedia*, *19*(2), 4–10. https://doi.org/10.1109/MMUL.2012.24

Zhong, Z., Wang, Z., Lin, L., Liang, H., & Xu, F. (2020). Robust Negative Obstacle Detection in Off-Road Environments Using Multiple LiDARs. *2020 6th International Conference on Control, Automation and Robotics, ICCAR 2020*, 700–705. https://doi.org/10.1109/ICCAR49639.2020.9108058

Zhou, L., Li, Z., & Kaess, M. (2018). Automatic Extrinsic Calibration of a Camera and a 3D LiDAR Using Line and Plane Correspondences. *IEEE International Conference on Intelligent Robots and Systems*, 5562–5569. https://doi.org/10.1109/IROS.2018.8593660

Zhu, J., Li, C., & Xu, J. (2016). Digital Image Stabilization for Cameras on Moving Platform. *Proceedings - 2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2015*, 255–258. https://doi.org/10.1109/IIH-MSP.2015.23

Zhu, W. H., & Lamarche, T. (2007). Velocity estimation by using position and acceleration sensors. *IEEE Transactions on Industrial Electronics*, *54*(5), 2706–2715. https://doi.org/10.1109/TIE.2007.899936

Zogopoulos, E. (2021). *Battery Technology: A New Era Emerging | Energy Industry Review*. Energy Industry Review. https://energyindustryreview.com/analysis/battery-technology-a-new-era-emerging/