

ROBUST, TIME-CRITICAL, EVIDENCE-BASED
ADAPTIVE DATA FUSION

by

MOHAMMAD AMIN JAVADI

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2020

Copyright © by Mohammad Amin Javadi 2020

All Rights Reserved



ACKNOWLEDGEMENTS

I would like to thank my supervisor, professor Dr. Brian L. Huff for constantly motivating, encouraging me, and for his genuine advice throughout the course of my PhD studies. I am grateful to have my family members support and patience especially my beloved wife. I am also thankful of the advisement from Dr. Jay Rosenberger and Dr. Bill Corley who served as my committee members.

March 02, 2020

ABSTRACT

ROBUST, TIME-CRITICAL, EVIDENCE-BASED ADAPTIVE DATA FUSION

Mohammad Amin Javadi, PhD

The University of Texas at Arlington, 2020

Supervising Professor: Brian L. Huff

Sensors have become inevitable part of many studies and working areas ranging from navigation, transportation and medical applications. A sensor can help a user in a variety of situations including dangerous, inaccessible, time and money-consuming circumstances. Applying multiple sensors simultaneously allows for improving the accuracy of measurement estimates for system states. As an example, a part of this study uses a GPS sensor to increase the accuracy of the position estimation obtained by an IMU in an indoor environment. The same GPS device with position outputs can also be studied to provide a new measuring dimension such as velocity. This way of sensors helping each other is covered in this study to achieve more accurate and robust estimates compared to when they contribute separately.

This study is carried out to develop a new way of achieving better estimates of system states. An attempt to indicate the applicability and robustness is also provided for navigation purposes. This work is founded on evidence-based theory wherein pieces of evidence or facts are used to cross-check the outputs given by sources. It also applies the concept of meta-sensing in which proportional weights are assigned to the data from each

sensor based on how close the data from each sensor is to the others. This causes the estimates to be more robust and reliable in a real-time environment. The obtained outcomes show more reliable decision-making under the defined scenarios.

Table of Contents

ACKNOWLEDGEMENTS	III
ABSTRACT	IV
LIST OF ILLUSTRATIONS	VIII
LIST OF TABLES	XII
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 LITERATURE REVIEW	4
2.1 Uncertainty and Data Fusion Methodologies	5
2.2 Dealing With Noise and State Estimation	8
2.3 Contribution Notes.....	14
CHAPTER 3 METHODOLOGY	15
3.1 Model Explanation.....	17
3.2 Sensor / Data Source Analysis and Modeling.....	19
3.2.1 Sensor / Data Source Selection and Applications.....	21
3.2.2 Sensor / Data Source Error Mode Independence and Data Correlation	22
3.2.3 Equivalence	23
3.3 Sensor Fusion Model Selection.....	29
3.3.1 Analysis To Direct The Data Fusion Activities.....	31
3.4 The Computing Infrastructure Middleware and its Necessity.....	34
3.4.1 Data Processing Architecture and Structure	36
CHAPTER 4 DATA ANALYSIS AND RESULTS	39
4.1 Software and Hardware Configurations	39
4.2 Evidence-Based Data Fusion.....	41
4.3 Real-Time Data Collection In ROS	42
4.4 Results and Comparisons – Experiment 1.....	44
4.4.1 Easting Position.....	44
4.4.2 Northing Position	47
4.4.3 Heading Angle	50

4.4.4	Velocity	54
4.5	Results and Comparisons – Experiment 2.....	57
4.5.1	Scenario 0 - Average.....	60
4.5.2	Scenario 1 – Fusion Algorithm Added.....	66
4.5.3	Scenario 2 – Grouping The Same-Class Sensors	76
4.5.4	Scenario 3 – System Model Added	85
4.5.5	Scenario 4 – System Model’s Weight Removed.....	94
4.6	Performance Overview – Average and Fusion Scenarios	109
CHAPTER 5	CONCLUSIONS AND FUTURE WORK.....	119
5.1	Concluding Remarks	119
5.2	Future Work.....	120
APPENDIX	122
	Difference Between Two Consecutive Data	122
	Fusion Algorithm – Easting Displacement	122
	Fusion Algorithm – Northing Displacement.....	123
	Fusion Algorithm – Heading Angle	124
	Fusion Algorithm – Velocity.....	126
	IMU Acceleration Filter (Zeroing down)	128
	Determining the Velocity Based on Heading Angle for IMUs (Northing)	128
	Determining the Velocity Based on Heading Angle for IMUs (Easting).....	129
	Lidar MatchScan	130
	GPS Magnetic Field Conversion (1)	130
	GPS Magnetic Field Conversion (2)	131
REFERENCES	136
BIOGRAPHICAL INFORMATION	140

LIST OF ILLUSTRATIONS

Figure 3-1 General system representation	18
Figure 3-2 System representation of the current research	18
Figure 3-3 GPS block in Simulink	24
Figure 3-4 IMU block in Simulink	26
Figure 3-5 Lidar block in Simulink.....	27
Figure 3-6 GPS Magnetic Field block in Simulink.....	28
Figure 3-7 GNC block in Simulink.....	29
Figure 3-8 Delaying Matlab function and a delay block.....	30
Figure 3-9 An example on the fusion algorithm	32
Figure 3-10 ROS communication algorithm (Clearpath Robotics 2014)	34
Figure 3-11 Nodes and topic relationship (Martinez Romero 2014).....	35
Figure 3-12 ROS nodes and topics applied in the research.....	36
Figure 4-1 Sensor calibration process in Mission Planner software.....	40
Figure 4-2 Pixhawk (PX4).....	41
Figure 4-3 U-Blox GPS sensor	41
Figure 4-4 ROS start-up screen in a terminal on Ubuntu	42
Figure 4-5 MAVROS package start-up screen in a terminal on Ubuntu.....	43
Figure 4-6 A representation of a subscriber block.....	43
Figure 4-7 Experiment 1 run	44
Figure 4-8 Truth and raw values of all sensors for Easting (Y) axis – normal conditions.	45
Figure 4-9 Fused, truth, & raw (averaged) values of Easting (Y) axis – normal conditions	45
Figure 4-10 Truth and raw values of all sensors for Easting (Y) axis – IMU2 noisy.....	46

Figure 4-11 Fused, truth, & raw (averaged) values of Easting (Y) axis – IMU2 noisy.....	47
Figure 4-12 Truth & raw values of all sensors for Northing (X) axis – normal conditions.	48
Figure 4-13 Fused, truth, & raw (averaged) values of Northing (X) axis – normal conditions	48
Figure 4-14 Truth and raw values of all sensors for Northing (X) axis – IMU2 noisy	49
Figure 4-15 Fused, truth, & raw (averaged) values of Northing (X) axis – IMU2 noisy	50
Figure 4-16 Truth and raw values of all sensors for Heading Angle – normal conditions	51
Figure 4-17 Fused, truth, & raw (averaged) values of Heading Angle – normal conditions	52
Figure 4-18 Truth and raw values of all sensors for Heading Angle – GPS (latlon converted) noisy	53
Figure 4-19 Fused, truth, & raw (averaged) values of Heading Angle – GPS (latlon converted) noisy	53
Figure 4-20 Truth and raw values of all sensors for Velocity – normal conditions	54
Figure 4-21 Fused, truth, & raw (averaged) values of Velocity – normal conditions	55
Figure 4-22 Truth and raw values of all sensors for Velocity – IMU2 noisy	56
Figure 4-23 Fused, truth, & raw (averaged) values of Velocity– IMU2 noisy	56
Figure 4-24 Experiment 2 run	58
Figure 4-25 Easting displacement – Scenario 0	60
Figure 4-26 Scaled Easting displacement – Scenario 0	61
Figure 4-27 Northing displacement – Scenario 0	61
Figure 4-28 Heading angle – Scenario 0	62
Figure 4-29 Scaled Heading angle – Scenario 0	63
Figure 4-30 Velocity – Scenario 0	63
Figure 4-31 Scaled Velocity – Scenario 0	64

Figure 4-32 Animated traversed path – Scenario 0	65
Figure 4-33 Easting weights – Scenario 1	66
Figure 4-34 Easting displacement – Scenario 1	67
Figure 4-35 Scaled Easting displacement – Scenario 1	67
Figure 4-36 Northing weights – Scenario 1	68
Figure 4-37 Northing displacement – Scenario 1	69
Figure 4-38 Heading weights – Scenario 1	70
Figure 4-39 Heading angle – Scenario 1	71
Figure 4-40 Velocity weights – Scenario 1	72
Figure 4-41 Velocity – Scenario 1	73
Figure 4-42 Scaled Velocity – Scenario 1	73
Figure 4-43 Animated traversed path – Scenario 1	75
Figure 4-44 Easting weights – Scenario 2	76
Figure 4-45 Easting displacement – Scenario 2	77
Figure 4-46 Scaled Easting displacement – Scenario 2	77
Figure 4-47 Northing weights – Scenario 2	78
Figure 4-48 Northing displacement – Scenario 2	79
Figure 4-49 Heading weights – Scenario 2	80
Figure 4-50 Heading angle – Scenario 2	81
Figure 4-51 Velocity weights – Scenario 2	82
Figure 4-52 Velocity – Scenario 2	83
Figure 4-53 Animated traversed path – Scenario 2	84
Figure 4-54 Easting weights – Scenario 3	85
Figure 4-55 Easting displacement – Scenario 3	86
Figure 4-56 Scaled Easting displacement – Scenario 3	86

Figure 4-57 Northing weights – Scenario 3	87
Figure 4-58 Northing displacement – Scenario 3	88
Figure 4-59 Heading weights – Scenario 3.....	89
Figure 4-60 Heading angle – Scenario 3	90
Figure 4-61 Velocity weights – Scenario 3	91
Figure 4-62 Velocity – Scenario 3.....	92
Figure 4-63 Animated traversed path – Scenario 3.....	93
Figure 4-64 Easting weights – Scenario 4	94
Figure 4-65 Easting displacement – Scenario 4	95
Figure 4-66 Scaled Easting displacement – Scenario 4.....	95
Figure 4-67 Northing weights – Scenario 4	96
Figure 4-68 Northing displacement – Scenario 4	97
Figure 4-69 Heading weights – Scenario 4.....	98
Figure 4-70 Heading angle – Scenario 4	99
Figure 4-71 Velocity weights – Scenario 4	100
Figure 4-72 Velocity – Scenario 4.....	100
Figure 4-73 Animated traversed path – Scenario 4.....	101
Figure 4-74 Easting fusion results	114
Figure 4-75 Easting fusion results – Scaled	115
Figure 4-76 Northing fusion results.....	116
Figure 4-77 Heading fusion results.....	117
Figure 4-78 Velocity fusion results.....	118

LIST OF TABLES

Table 3-1 Research sensors and deliverables	21
Table 3-2 4-noded graph experimental results	33
Table 4-1 Justification of Velocity in Scenario 1	74
Table 4-2 Easting displacement for all scenarios	110
Table 4-3 Northing displacement for all scenarios.....	111
Table 4-4 Heading angle for all scenarios	112
Table 4-5 Velocity for all scenarios	113
Table A-1 Easting weights for all scenarios.....	132
Table A-2 Northing weights for all scenarios	133
Table A-3 Heading angle weights for all scenarios.....	134
Table A-4 Velocity weights for all scenarios	135

CHAPTER 1

INTRODUCTION

Many types of systems must consume streams of incoming data to make real-time automated control decisions. Automated stock trading applications, aircraft auto pilots, or automated driver assist applications such as auto braking, or automated parallel parking are all potential examples of this class of system. Data streaming can happen from one or multiple sources of data. While using one source of data may not represent all characteristics of a system, applying multiple sources of data can increase the accuracy and robustness of the results. As an instance, a specialist may decide on the existence and growth trend of a brain tumor by considering the evidence from multiple sources of data such as Magnetic Resonance Imaging (MRI), physical disorder monitoring, etc.

The appropriateness and quality of the control decisions made are directly dependent on the quality, and reliability of the data input as well as the capabilities and flexibility of the algorithms that process that data. Continuing advances in embedded data processing speeds and the availability of low-cost sensor technologies are expanding the potential for autonomous control. The ready availability of low-cost embedded sensors, however, does not ensure the availability of good system state information. Embedded controllers typically rely on one, or a few, sensor input streams to determine a system state. If the sensor data stream is interrupted or corrupted, the resulting control actions maybe incorrect. This fact indicates the need for further research into the creation of a more robust, adaptive sensor fusion techniques which perform well with faulty or noisy sensors and/or disruptive environmental conditions.

The objective of this research is to present a real-time solution to the problem of how a system can determine its fused estimate dynamically for a system state variable from a

stream of sensor readings or data input sources. It is assumed that the input data streams cannot reliably provide input values at a fixed rate. This drives the proposed algorithm to examine the validity of incoming data from each sensor by using the values from other sensors and pieces of evidence to ensure that the system state estimator is robust to corrupted inputs. The applied approach explores anomalies in the incoming data by comparing each source of data with other sources and pieces of evidence, and assigns weights to each source to specify the fused estimate for each time step throughout the system operation. A variety of achievable data types such as acceleration, velocity, displacement etc. are also generated through mathematical functions for proper data analysis. This development is designed to identify added sources of verification to increase the reliability of the outputted data which will be discussed in the beginning of Chapter 3.

This research presents a strategy for using multiple input data streams to generate system state estimates. This strategy utilizes underlying beliefs about the behavior of the system and knowledge of the potential failure modes and causes associated with the various data streams. The proposed technique is described as evidence-based because it attempts to use this knowledge of system and sensor behavior to filter the input data streams prior to their use by the control system. Rather than directly feeding in sensor stream inputs, this technique proposes the generation of “fused estimates” of critical system states and relying on these estimates as inputs into the control system. Applying the models and filtering strategies to fuse multiples of sources (sensors) enables a decision maker to have more reliable outputs.

In an attempt to better illustrate this strategy, a simple skid-steer autonomous ground vehicle (AGV) is used as an example. It is assumed that to adequately control an AGV it is required to know its position and orientation relative to a navigation frame whose origin is arbitrarily defined as the vehicle’s location when it is parked in its charging station. This

vehicle will be commanded to various locations about its operating environment through a series of X (Northing) and Y (Easting) Cartesian coordinate positions. To simplify this problem, it is assumed that the AGV operates over level ground so that changes in altitude Z (Positive Down) can be ignored. To support waypoint navigation, it is assumed that the AGV is equipped with a low-cost control system and sensor suite that provides GPS, IMU (Inertial Measurement Unit), digital compass, and Lidar (Light Detection And Ranging) data streams.

In the current study, the vehicle which contains four sensors: GPS, IMU, digital compass, and Lidar, will be able to fuse different data under distinct conditions one of which happens when an external magnetic field generates disturbing magnetic force which impacts the magnetometers mounted on the vehicle and causes them to drift.

Due to the existence of environmental or noise factors, data from different sensors are fused using relative weights in such a way that when one or more sensors (sources) fail to transmit the data, the fusion algorithm still specifies navigation directions satisfactorily.

The contribution of this research can be expressed in terms of the following:

- Improved stability of the system state estimation under noisy situations through a novel evidence-based fusion algorithm,
- Evidence-based noise filtration,
- Dynamic weight-assignment data fusion based on graph theory,
- Working models without characterizing sensor error distributions,

CHAPTER 2

LITERATURE REVIEW

Researchers continue to explore advancements in automated systems using sensors in order to obtain more robust outputs or estimates by applying more than one sensor or source of data (Xiao and Qin 2018). According to (Wieland and Marcus Wallenburg 2012), robustness is considered as a proactive strategy capable of resisting change without adapting its initial stable configuration. The concept of robustness in this research refers to an attempt to decrease the variation of data in the fused outputs. Especially, when one or more sensors produce distorted data, the system is required to generate a robust output using the data from the sensors that are believed to provide undistorted data based on pieces of evidence. As a clearer representation of robustness related to this research, three positioning sensors are considered on a vehicle collecting 20 displacement values each second. In each time step ($1/20^{\text{th}}$ of a second), three values are obtained and compared with each other. Three values of 0.08m, 0.1m, and 0.4 from all these sensors are assumed. A maximum speed of 32.19k/h or 0.447 meters per 20^{th} of a second is also considered for the vehicle. All the sensors reported values within the range that the vehicle could travel; However, value from the third sensor is distant from the other two sensors. The researcher concluded that the value from the third sensor should be given less weight when averaging all three values to demonstrate the mean of displacement in a particular time step. In essence, the instability in the third sensor's data compared to the data from the other two sensors caused the system to remove or de-weight the distorted data such that the system could reach a stable configuration. This example conveys the algorithm robustness to outlying data, which reacts in terms of the weight assigned to each sensor in every time step.

Unreliability in sensor input data can be due to sensor failures or environmental effects that reduce the accuracy of the decision in the system. More reliable and less complicated algorithms would improve the decision-making process. Researchers may devise their own methodology or use a combination of methods to handle the reliability of data transferred to the system from different sensors. Fusion strategies and methodologies have been broadly discussed by many researchers and will be addressed in the next subsections. This study tries to explain these approaches in each part, address the resolutions suggested by the researchers and their drawbacks, and present strategies and contributions to overcome those gaps.

2.1 Uncertainty and Data Fusion Methodologies

Researchers use multiple sensors to reach more precise results as compared to use of a single sensor which can fail easily under a certain condition. For example, GPS signals may be blocked in a covered area and transmit no signal or false signals. Multiple sensors, however, can produce less failure rates when set up correctly. They may cover the failure caused by a sensor in a group of sensors. Uncertainty is the fundamental concept that has caused many researchers to provide more evidence for the purpose of improved robustness of system state estimates by either adding more sources of information or obtaining further facts from the existing sources through more analysis.

After collecting the data, it needs to be fused and analyzed for which many fusion methodologies and algorithms have been proposed in a variety of situations. Despite the fact that outputs generated by these heterogeneous sources (sources with different natures such as a GPS and an IMU) are different they all can be considered for data fusion to provide similar outputs using novel approaches depending on the research application

such as in speech recognition (Safavi and Mporas 2017), (Ding and Lin 2017), (Du et al. 2015) (Mitra et al. 2016), and medicine (Chromy and Klima 2017) (Zheng et al. 2017).

Current study uses weights as proportional values of each sensor in every time period in which it is fusing the data. A similar strategy has also been used by (Xiao and Qin 2018). What this research proposes, however, applies a piece of evidence such as the kinematics of the system or the data from other sensors. In other words, this work introduces the evidence and each sensor votes as a basis for fusion. That way, the fusion strategy in one time step may differ from other time steps.

All the above-mentioned efforts are to handle uncertainty as much as possible which is still an open issue. For example, in wireless sensor network applications the collected data are often imprecise and uncertain (Xiao and Qin 2018). Several mathematical approaches have been discussed by researchers to deal with uncertainty out of which the most important are briefly explained as follows:

(1) Bayes theory, which applies the probability distribution of the measured data and a conditional probability or likelihood to represent an estimate/output. Bayes fusion strategy applies Bayes` rule to combine successive measurements of the state of a system from a single source, which involves obtaining a new estimate for the target state given the previous old estimate (Mishra 2013) with examples on reconstructing sparse wireless signals (Ji et al. 2016), obtaining spatial maps from conflicting information (Bogaert and Gengler 2018), dynamic object tracking (Markovic and Petrovic 2014), multi-point gas detection (Hou et al. 2016), and power transformer condition evaluation (Li et al. 2018). This theory involves a prior knowledge and concept of conditional probability. This study, however, does not apply prior knowledge in order to avoid unexpected bias for estimation.

(2) Dempster-Shafer (DS) Theory, that was first proposed by Dempster (Dempster 1967) in his combination rule of belief functions and consequently developed by Shafer

(Shafer 1976) which does not need strict prior conditions as Bayesian theory of probability and includes great advantages of stating the uncertainty directly and deriving new evidence using a combination of other sources of evidence. This theory allows for belief functions to base degrees of belief for questions of interest even if degrees of belief do not have any mathematical properties. The fundamentals to DS theory include (a): The idea of obtaining degrees of belief for one question from subjective probabilities for a “related” question, (b) When degrees of belief are based on independent items of evidence, DS theory allows for them to be combined. Dempster-Shafer theory, however, carries a shortcoming when highly conflicting pieces of evidence are involved. Therefore, trials were carried out by researchers through several methods by either modifying Dempster’s combination rule or pre-processing the sources of evidence as referenced and discussed by (Xiao 2019), and also addressed as the decision-making process improvement in the fields of localization (Elkin et al. 2017), (Kasebzadeh et al. 2014), (Li et al. 2015a), medical sciences (Li et al. 2015b), (Wang et al. 2016), (Moraru et al. 2018), (Gui et al. 2017), (Mulyani et al. 2016), and fault diagnosis (Hui et al. 2017b), (Tang et al. 2017), (Jiang et al. 2016), (Zhou et al. 2018), (Jiang et al. 2017), (Xiao 2017), (Khazaei et al. 2016), (Lv et al. 2012), (Hui et al. 2016), (Yao et al. 2017), and (Hui et al. 2017a). Weighting strategy in the current study, which will be discussed in the Chapter 3, presents a solution to combine all data from different sources regardless of the nature of the sources.

(3) Fuzzy Logic, which is a methodology applied where data obtained from different sources are not precise, tries to deal with uncertainty in a reasonable computation framework. This theory extends the Boolean set theory and the related two-valued logic (Stover et al. 1996) proposed by (Zadeh 1965) who extended for those binary numbers to any value between 0 and 1, continuously spread. Instances of the fuzzy logic include enhanced accuracy of sensors using fuzzy logic data fusion (Al-Sharman et al. 2018), (Jian

et al. 2011), better data clustering (Majumder and Pratihar 2018), and medicine (Ahmadi et al. 2018) and (Mehranfar et al. 2017). Fuzzy logic also uses membership functions as a basis for transforming binary values, referred to as crisp values, into values within a range. It may create issues to find suitable membership values. Moreover, the transformation process and selecting the appropriate membership function, such as triangular, trapezoidal, and Gaussian (Jian et al. 2011) may be inconvenient.

Researchers also tried to obtain fuzzy values for each sensor in every time period; however, this may ignore the real-time decision making process (Seiti and Hafezalkotob 2018). As an example, in a study carried out by (Xiao and Qin 2018), they generated initial data 100 times but in a variation range of $[-0.1, 0.1]$ as their experimental settings; however, producing data and obtaining the desired outcome may be different than actually implementing the algorithm within real-time and dynamic framework. This means that how sensors would behave may not match lab-made experiments. Therefore, obtaining values in real-time and at every time period is presented as a novelty to be addressed in this research.

2.2 Dealing With Noise and State Estimation

Noise is considered as undesired irregular fluctuations which accompany the actual signal. All signals include degrees of noise depending on the sensor type and environmental factors. Therefore, avoiding or reducing noise has always been a considerable concern since real data can be totally disturbed in the presence of excessive noise. The need to eliminate noise has led researchers to estimate the outputs of a model, usually known as state. System states may sometimes be unavailable due to physical constraints, technological restrictions, expensive cost for measuring (Hu et al. 2016), or be incorrect because of the environmental noise/effects. About motion and position sensors

such as GPS and IMU (including gyroscope and accelerometer), researchers apply state estimation/filtering methods to obtain robust outputs to tackle the presence of noise or missing data. This includes a variety of applications such as localization, velocity estimation, navigation, etc.

The Kalman filter (KF), invented by (Kalman 1960), is a very common filter to estimate a future state. Known as an appropriate linear estimator (Selin 1964), having a good performance with linear systems and the most famous including its various representations (Zaidner and Shapiro 2016), KF has been the most impressive approach among others. But due to drawbacks in various situations such as increased complexity of a model and the necessity for rapid change response, harder parameters calibration, linearity of the system, Gaussian probability distribution function, and prior knowledge of system measurement noise covariance (Q) and update measurement noise covariance (R) (Nada et al. 2018), etc., researchers on data fusion and analytics are persuaded to look for new methodologies as will be discussed further.

As an instance, (Lin et al. 2016) used KF to analyze trajectory data and segmentation. However, if one attempts to use multiple sources of data for a single input of KF, such as the easting position of the AGV example in this study, another strategy for fusion is still needed.

A more recent study by (Wang et al. 2019) showed the use of GPS improved by KF for outdoor environment and combination of IMU and magnetometer while indoor for better positioning estimation. They created a novel approach to estimate the heading angle using two adjacent data points. They declared less distortion in magnetic-interfered environments. However, their research is based on a few assumptions which may not always be true. For example, two adjacent data points obtained from magnetometer are considered the same under high-frequency data-collection conditions.

Another study by (Mishra 2013) introduced sliding windows using quadratic optimization to estimate the location and track the movement of a mobile robot platform wherein high variance and bias in the GPS data exists. Although their research included only GPS receivers and their contribution could handle levels of noise regarding one sensor type, the current study includes diverse sensors, resulting in the use of a novel approach toward noise reduction. A similar study to this work is done by (Kim and Lee 2016) to increase the positioning accuracy using GPS and IMU in which they used regular KF and added fuzzy IF-THEN rules to moderate the drawbacks known for KF such as the impossibility to gain very accurate system and measurement models for a real environment in a tunnel or under an overpass. Even worse for the GPS receiver, it carries low frequency and sensor errors. To overcome this, they innovated an algorithm to control the Q and R matrices.

Another variation of KF is the Extended Kalman Filter (EKF). EKF is an extended format of KF which can be applied to nonlinear systems (Julier and Uhlmann 2004). It also provides a growth trend so that measurements from other sensors such as optical flow sensors and laser range finders can be added as further improvements in accuracy and robustness. In a recent study done by (Sun et al. 2019), positioning a ground vehicle for restaurant environment was established using odometry, IMU, and magnetometer, combined with EKF. They intended to show the use of low-cost sensors and reduction in labor costs. Similar to many other researches in this area their study suffers from Gaussian noise measurement assumption for IMU. It also ignores external interference.

(Kim and Lee 2016) applied EFK as a core method for the positioning estimation but later question that by resorting to the fact that EKF is dependent on how accurate the system and measurement models are, which is the main drawback of EKF.

(Kim and Lee 2016) also used GPS and IMU inputs to the KF. In addition, their contribution is to control the system and measurement noise covariance by incorporating a fading factor, the GPS sensor bias, and the degree of reliability of GPS signal. A fading factor is built to apply a factor matrix to the predicted covariance matrix to deliberately increase the variance of the predicted state vector. The fact that they use two GPS devices raises the issue of having the same failure modes of the same sensors. This is well shown in Chapter 4 where the first experiment applies the same IMUs but the second experiment handles sensors of the same type as groups. Additionally, they are still feeding in the sensor output to the KF. Although it improves that, the question is now what if the sensor itself generates noisy or missing data? Then the whole algorithm is prone to fail or provide weak performance. The algorithm in this research, however, compares the same input, for instance position, from different sensors such as GPS, IMU and Lidar, and provides the fused estimate on a voting basis which will be reviewed in Chapter 3.2.3. In a close study to this work done by (Munguía 2014), Inertial Navigation System (INS) is used in a framework of EKF with the aid of GPS device for better estimates. The INS is a widely used dead reckoning system (navigation using a previously determined position rather than by using landmarks or electronic navigation methods) which fuses sensory information taken from inertial sensors (accelerometers) and rotational sensors (gyroscopes) in order to continuously estimate the position and the orientation of the body (vehicle, in the concept of the current study).

Regarding the IMU noise, researchers may use other methods. For instance, (Pan et al. 2016) applied sensor calibration, digital-analog conversion and signal smoothing on the raw signals to reduce the IMU noise in a trajectory reconstruction concept. They also applied a low-pass filter based on the Exponential Weighted Moving Average (EWMA) to smoothen the IMU noise. Therefore, each scholar may use their own simple, heuristic or

more complicated filtering method to extract the actual signal from the noise efficiently. In the current study, initial noise from the IMU sensor and the error concluded from conversions are handled using pieces of evidence and the concept of equivalence to increase the reliability of the filtering process. Equivalence will be discussed in Chapter 3.

Regarding the Lidar distance and intensity accuracy and noise, different methods are used as discussed in (Xu et al. 2019). They address laser echo signal calculations and more reliable de-noising methods (Chang et al. 2018), particularly when the detection range is large. This study uses a high-performance, indoor device and does not involve large area of detection. Therefore, Lidar noise is not a consideration.

Most of the sensor fusion methods found in this literature review assume that sensor noise or bias follows a Gaussian or constant distribution. Gaussian noise refers to the statistical noise having the probability density function equal to that of the normal distribution (Barbu 2013). As an instance, (Guo et al. 2017) applies this assumption for a target-tracking model using several sensors. Even KF holds this assumption. Therefore, this motivated the current study to introduce an algorithm not dependent on a sensor error distribution type.

Other variations of KF such as Adaptive Kalman Filter (AKF) and Unscented Kalman Filter (UKF) are also used by researchers. In an attempt to integrate INS and GPS for navigation purposes, (Mohamed and Schwarz 1999) showed a 50% improvement in estimating the navigation parameters by implementing an AKF when compared to that of a conventional KF. However, they mention drawbacks for AKF such as having a more complex algorithm leading to an additional estimation block. Also, it still needs tuning either Q or R or both. Afterwards, by initially representing the drawbacks of EKF, (Wan and van der Merwe 2000) introduced an improvement to UKF which was first invented by (Julier and Uhlmann 1997). According to (Wan and van der Merwe 2000), the state distribution in

EKF is approximated by a Gaussian random variable, then propagated analytically through the first-order linearization of the nonlinear system. This may produce large errors in the true posterior mean and covariance of the transformed Gaussian random variable, which may lead to sub-optimal performance and sometimes divergence of the filter. The UKF, however, addresses this problem by using a deterministic sampling approach. It also uses a Gaussian random variable to approximate the state distribution but is now represented using a minimal set of carefully chosen sample points. These sample points completely capture the true mean and covariance of the Gaussian random variable, and when propagated through the true nonlinear system, captures the posterior mean and covariance accurately to the 3rd order (Taylor series expansion) for nonlinearity. The EKF, in contrast, only achieves first-order accuracy. The computational complexity of the UKF is the same order as that of the EKF.

Particle Filter (PF) is another well-known estimation technique with the capability to be used for nonlinear non-Gaussian models (Munguía 2014). PF is the recursive computations of Monte Carlo-based statistical signal processing (Gustafsson et al. 2002) which uses a set of samples (referred to as particles) to represent the posterior distribution of stochastic processes given noisy/partial observations. It uses the concept of sequential importance sampling and the Bayesian theory which approximates relevant distributions with random measures composed of particles (samples from the space of the unknowns) and their pertinent weights (Djuric et al. 2003). PF can be an alternative to EKF. It approximates continuous distributions using discrete random measures, which are composed of weighted particles, where the particles are samples of the unknown states from the state space, and the particle weights are probability masses computed through Bayes theory. In the implementation of particle filtering, importance sampling plays a crucial role and since the procedure is designed for sequential use, the method is also

called sequential importance sampling. The advantage of particle filtering over other methods is in that the exploited approximation does not involve linearization around current estimates but rather approximations in the representation of the desired distributions by discrete random measures. The key idea on the PF in comparison to EKF is finding an approximate solution using a complex model rather than an exact solution using a simplified model (Hsiao et al. 2005). On the other hand, a recognized problem with the particle filter is that its performance reduces rapidly when the dimension of the state space increases (Thrun 2002). In addition, it is essential to have a reliable way to detect divergence and to restart the filter. Due to the aforementioned complexities, this research avoids using this filter.

2.3 Contribution Notes

According to the above discussion, the methodologies of fusion strategies and appropriate modeling are still of considerable interest but distant from achieving the competence in analyzing different data concurrently (Fung et al. 2017). Therefore, the directions of this study are established toward an inclusive way of fusing data with the least possible rate of failures and better estimates throughout the operations. This is achievable through a novel algorithm for fusing the data using weight assignment. Preliminary work was implemented to utilize homogeneous sensors to include diverse operational scenarios in which system unpredictability in terms of linearity or non-linearity does not affect the fusion algorithm remarkably. System non-linearity refers to the system in which outputs are not merely a sum of inputs due to feedback or multiplicative effects between the components (Boeing 2016). A secondary experiment is also handled using heterogeneous sensors to reduce the bias caused by the same sensors.

CHAPTER 3

METHODOLOGY

In this chapter the concepts of evidence, sensor error independence, output equivalence, adaptive state estimation, and meta-sensing will be discussed. The ways these concepts contribute to the implementation of the proposed data fusion techniques will also be presented. The example of an AGV is used to present these concepts.

Evidence – The term evidence is accepted in this research to address facts or information indicating whether a belief or proposition is true or valid (Oxford University Press 2019). Within the proposed data fusion strategy, attempts are made to discover facts or information about the system being controlled and the incoming data streams that increase the belief in the accuracy and validity of the data used as control system inputs. System knowledge that can be used to support the creation of evidence can include knowledge of a system's kinematics and dynamics, verified performance constraints such as velocity and acceleration limits, knowledge of control inputs such as commanded speed and heading commands, and the underlying laws of physics such as the fact that a vehicle at rest cannot instantaneously move below the surface of the ground. This work also attempts to exploit the fact that the comparison of data generated from different sensor data streams can be used as evidence to increase or decrease the belief that the sensor data reflects the true state of the system.

Sensor Error Independence – This work is premised on the belief that system state sensors frequently generate data that does not reflect the true state of the system and these errors in sensor output are due to the characteristics of the sensors deployed and the system operating environment. Simple examples include: (1) The output from a digital

compass which can be biased by the presence of near-by ferrous objects or magnets, (2) IMU outputs that exhibit unbounded cumulative output error due to the integration of scale factor and bias errors, (3) consumer-grade GPS receivers that demonstrate position errors of approximately 5 to 10 meters due to the effects of atmospheric conditions, multipath error, orbit errors, and line of site obstructions. Sensor input data stream accuracy and repeatability is negatively affected by many causes of error. The good news is that different types of sensors generally have different causes of error. The concept of sensor error independence attempts to utilize errors in the output data streams of a specific class of sensor that are not present in the output of different types of sensors. System state estimates, such as position, orientation, velocity, and acceleration in the case of an AGV, can be derived from multiple types of sensors. Due to the fact that the cause of error in one type of sensor might not be present in a different sensor, it may be expected that errors in one sensor output stream will not be present in the data generated by a different sensor. To better illustrate this, in the AGV example mentioned earlier, a GPS and an IMU device have been used. Accelerometer readings in the IMU would change as far as the bumps which could impact the displacement gained in this way. However, this is not an error that affects displacement readings generated from GPS sensor. Therefore, the cause of error in an accelerometer would not cause any error to the GPS device in this real example.

By comparing the instantaneous change in sensor outputs simultaneously from multiple classes of sensors, it is possible to observe when the information generated by one type of sensor deviates from the information generated from the majority of the other sensor types. These uncollaborated shifts in sensor output may provide evidence that the value shift is caused by data errors generated by the sensor rather than an actual change in the state of the system being observed. This evidence can then be used to direct

algorithms to systematically reduce the influence of the sensor-induced data errors in the system state estimations.

Output Equivalence – This concept explores whether sensors can detect faulty data when one sensor is distorted. It is one of the definitions applied in this research, which is, converting sensor data to obtain new outputs from the same sensor. This helps the algorithm to detect faulty data. As an instance, GPS position data may be converted to velocity and acceleration estimates by taking derivatives. While acceleration, velocity, or displacement from IMU can fail under a circumstance, similar values from GPS do not since GPS values have different failure modes.

Adaptive State Estimation – The word state refers to an output such as displacement. When a state is estimated it means that various evidence or facts have been utilized to get a robust estimation. A robust estimation signifies that the estimated output is less prone to failure as compared to when no evidence or fact is used for estimation. What sensors or which facts to use under different situations also defines the concept of adaptive state estimation, indicating that it may involve different pieces of evidence at different times.

Meta-Sensing – This innovative concept refers to the fact that a sensor output is checked with the same output of another sensor to examine how other sensors agree with an output of a particular sensor. Accordingly, each sensor will be given a weight based on how close the output of that particular sensor is to other sensors. This weight can be different in various time steps.

3.1 Model Explanation

The model used for this research applies inputs to the system, functions as the heart of the system, mechanisms as the control factors, and the outputs which are considered

as the product. To better demonstrate the mentioned relationship, the system representation used in this study is shown next.

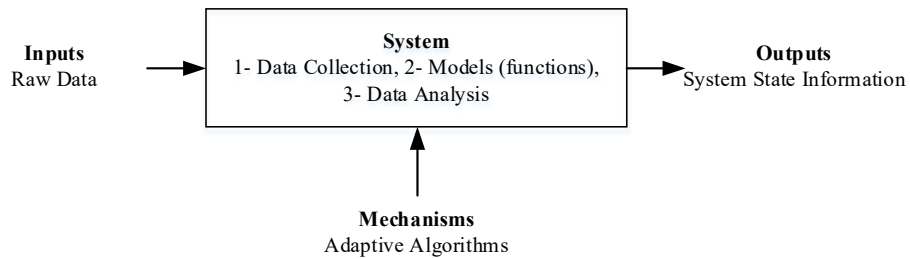


Figure 3-1 General system representation

Sensors transmit their data (Inputs) into the system and the algorithms subscribe to their required data. Other sensors` relative or converted data and the evidence are also added to support the decision-making process referred to as “Mechanisms”. Using the math, codes and modeling tools, the data is analyzed in the “System”. As a conclusion, the fused outputs provide better estimates (Outputs).

Applying the sensors and the structure of this work concludes the following system representation specific to the current study.

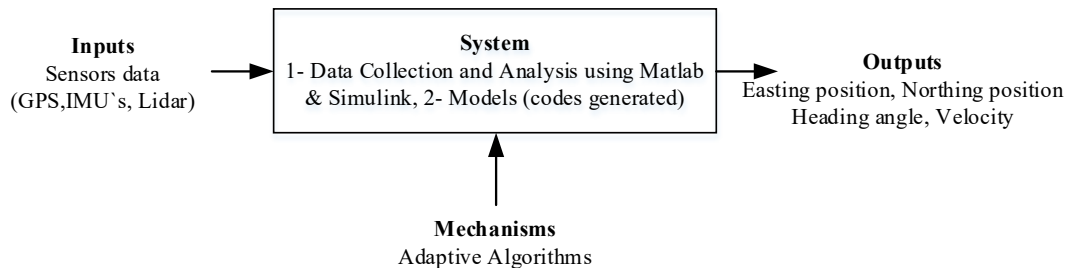


Figure 3-2 System representation of the current research

3.2 Sensor / Data Source Analysis and Modeling

Data source analysis is inspired by the concepts of fusion strategies and state estimation methods discussed in Chapter 2. Sensor data value and system kinematics determine the ratio of contribution (weight) for each sensor and all weights sum to 1 for a model in a specific time step.

Data from all sensors are obtained and go through the process. The modeling is performed based on evidence comparisons from different sources using the custom algorithms, standard functions, and libraries within the Matlab and Simulink development environments. The evidence-based sensor fusion algorithms are designed to detect shifts in sensor data values generated by anomalies within the system's operational environment. There are a significant number of alternate sensor inputs that remain uninfluenced by the anomaly. These unaffected alternate sensor input streams are used to serve as two purposes. They can be used to provide evidence that the primary sensor values may be corrupted, which can be used by the fusion algorithm to systematically reduce the influence of the suspect primary sensor's input stream on the system state estimation process. The uninfluenced secondary sensor input streams can also provide information used to generate state estimation values until the influence of the operational or environmental anomaly disappear and the primary sensor's input stream can again be trusted. These anomalies may include special circumstances such as a covered area or the presence of magnetic field distortions where a particular sensor is blocked from receiving signals.

The application of this research in navigation requires the use of specific input data streams which includes: position, heading angle, and velocity of a vehicle. These inputs

can be provided by using positioning and orientation sensors such as GPS, IMU, digital compass, and Lidar which are described as follows:

GPS – This device has several useful outputs out of which this research applies latitude, longitude, and altitude. Other information including velocity and acceleration may also be extracted using Simulink models.

IMU – It is a unit containing accelerometers (an electromechanical device by which the gravity, vibrations, and movement in one, two, or three axes can be sensed) and gyroscopes (a device which measures rotational motion) to calculate orientation, position and velocity.

Digital compass – This device uses a three-axis magnetometer to measure the absolute orientation of the vehicle with respect to the geomagnetic north. It is useful when determining the heading of the vehicle.

Lidar – This device includes a spinning motor to scan the surroundings and reports the ranges to the detected objects. Accuracy and width of the scanning area and range may not be the same on different devices.

The next table displays the sensors' deliverables (related to this work) in a tabular format.

Table 3-1 Research sensors and deliverables

Sensor	Native Deliverable	Derivatives
GPS	1- Longitude and latitude 2- Altitude	1- Position 2- Velocity 3- Acceleration 4- Heading
IMU	1- Angular rate 2- Acceleration (Body-frame)	1- Orientation 2- Heading 3- Acceleration (Inertial frame) 4- Velocity 5- Position
Digital Compass	1- Magnetic field values	1- Orientation 2- Heading
Lidar	1- Range	1- X and Y coordinates 2- Heading 3- Velocity

The data generated from the sensors are continuously monitored to detect distortion. This occurs to verify how trustable the data from each sensor is for the subsequent data fusion process. Relative weights, which sum to 1, are assigned to each contributing sensor using the concept of graph theory. This process is accomplished simultaneously for all possible pairs of outputs.

Each sensor's data is received in the ROS environment and published to Matlab for fusion purposes. The Robot Operating System (ROS) is a flexible platform which runs on the Ubuntu operating system. It uses reliable and independent procedures to manage a variety of sensors' outputs. That will be explained more in section 3.4.

The AGV test platform used to support this research was equipped with GPS, IMU, digital compass, and Lidar sensors. A Pixhawk autopilot system supplied the GPS, IMU and digital compass sensors. Additional information about the Pixhawk unit is provided in Section 4.1.

3.2.1 Sensor / Data Source Selection and Applications

All sensors contribute to all models as long as they are consistent with other sensors. According to various situations, different combinations of sensors may be required to

handle or correct the fusion algorithm. The point is that when applying different sensors, all may provide a particular output either directly or indirectly through conversions, which after calculations they all can provide the same output with a high correlation. Each sensor might have a unique cause(s) of error but a combination of different sensors become robust against a variety of errors because when one sensor fails due to a specific error, other sensors do not fail since that specific error does not cause them to fail as they have different failure modes.

Data that this research needs as the inputs for a navigation system includes: easting and northing positions (relative to true north), heading angle, and velocity. Parts of these inputs are provided directly while others require computations/conversions to yield the expected type of data.

3.2.2 Sensor / Data Source Error Mode Independence and Data Correlation

Retrieving data from sources with different failure modes concludes the advantage of independent functionality. This results in reduced possibility of errors occurring simultaneously. This study does not hold any assumption regarding sensor error distribution as an improvement to the gap mentioned in 2.2. Since each sensor is planned to contribute individually, error for each sensor is accompanied with that sensor. This way, even if the error for each sensor is not Gaussian or constant, the sensors' outputs are being compared to each other and weights are assigned. Therefore, not knowing the behavior of a sensor at a particular time step does not impact the performance of the algorithm. The following will discuss sensor error with regards to the sources used in this research to reveal failure modes that the upcoming algorithms will handle.

GPS – A common failure mode for a GPS receiver occurs when the device loses the lock on satellites or is surrounded by tall buildings which generates multi-path timing errors. This indicates that the signal is not reflected directly but leaps against surfaces to contact the sensor which generates error on the actual signal.

IMU – A common failure mode when using an IMU unit is that the user often produces error due to manual data conversion. The IMU yields almost reliable data by itself; however, when part of the data such as acceleration is integrated to get velocity, the methods available produce errors in each step. Therefore, the errors are accumulated and the errors become larger than the real data. This means that the noise from one time step is transferred to the next and is augmented such that at a point the output includes further noise than the actual signal. Noise should be well handled while converting the IMU's data such that the output contains the least noise possible. This is another contribution which will be discussed in section 3.2.3.

Digital Compass – A common failure mode for a digital compass is when another magnetic source influences the magnetic field generated by the earth pole and output from a digital compass can be easily disturbed when the unit is placed near a magnet or other ferrous object.

Lidar – This device may fail to measure the range or angle when confronted with high light beams or great brightness reflected from glazing surfaces.

3.2.3 *Equivalence*

In order to ensure equivalent outputs, pair-wise data correlation comparisons among all sensors are initially performed through experiments under normal circumstances before the actual run. This examines to what extent each sensor agrees with the others. As

another part of the initial test of equivalence, Matlab graphical tools, namely scopes, are also visually checked to see if they deliver almost the same outputs in a same scenario, i.e., the results from replicated scenarios should be consistent. Through equivalence conversions, sensor data is converted to another useful output that can be compared to the outputs of other sensors with different failure modes.

The following sections discuss the selected sensors, the methods used to obtain the desired output, and how an output from a particular sensor can be mapped and compared to that of another sensor in the model within the Matlab and Simulink environment.

GPS Block – This is another part of the whole model in Simulink exhibited in the next figure.

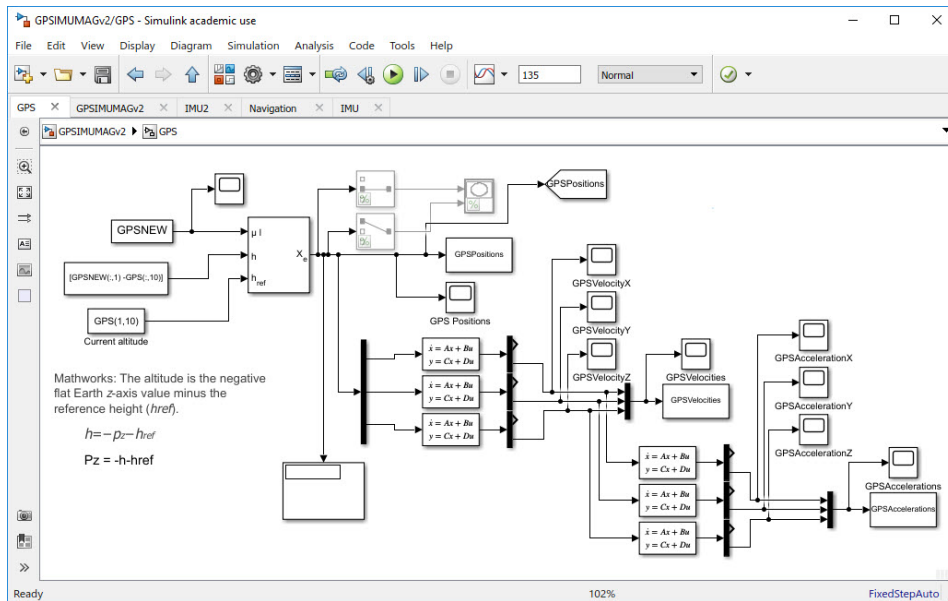


Figure 3-3 GPS block in Simulink

Collected latitude, longitude, and altitude readings from the GPS sensor are converted to positions toward X, Y, and Z axes in a local reference frame. The initial latitude and longitude values are set as the origin of the local reference frame. By using state-

space blocks, position values are converted to velocity and acceleration values. Differentiating the position yields the velocity.

State-space blocks are more accurate alternatives to taking direct derivatives. According to linear systems theory, the state-space model is a model that uses state variables to describe a system by a set of first-order differential or difference equations, rather than by one or more n^{th} -order differential or difference equations and is represented as follows:

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

$$y = Cx + Du$$

Equation 3-1

Where x is the state vector, y is the output vector, u is the input (or control) vector, and A , B , C , and D are parameters demonstrating the state (or system) matrix, input matrix, output matrix, and feedthrough (or feedforward) matrix, respectively. The matrix coefficients must have the following characteristics.

- A must be an n -by- n matrix, where n is the number of states.
- B must be an n -by- m matrix, where m is the number of inputs.
- C must be an r -by- n matrix, where r is the number of outputs.
- D must be an r -by- m matrix.

Experimental trials concluded the following matrices.

$$A = \begin{bmatrix} 0 & 1 \\ -150 & -40 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 40 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{3.5} \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Equation 3-2

Velocity X and Y obtained from this block contribute as follows for the velocity of the vehicle:

$$V = \sqrt{V_x^2 + V_y^2}$$

Equation 3-3

IMU 1 & 2 Blocks – The following is applicable to both IMU1 and IMU2 since they both use the same data processing strategy. Since two IMU sensor units were used in the experiments, two separate blocks were used in the Simulink model which output separately. A representation of one of the IMU blocks is shown below.

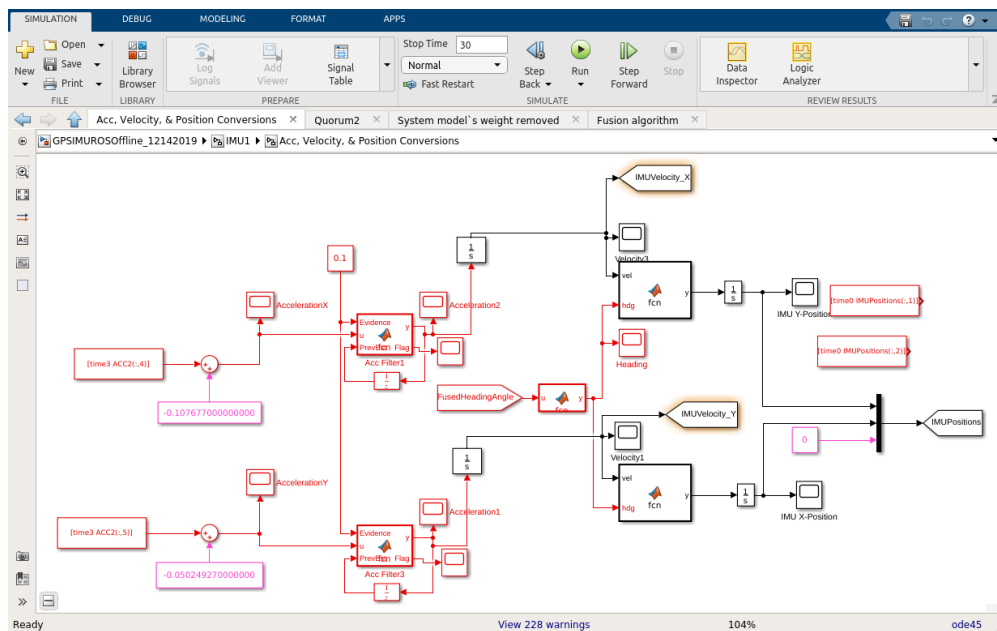


Figure 3-4 IMU block in Simulink

After obtaining IMU's acceleration in all axes they are zeroed down. Maximum acceleration of the vehicle (robot in the second experiment) is 0.1m per 0.1s². Instantaneous acceleration values larger than 0.1m/0.1s² or lower than -0.1m/0.1s² are substituted by their previous value. This filtered data is integrated to obtain velocity.

The velocity and heading sensor inputs determine the position estimates after another integration step. Since the vehicle may have moved any angle other than strictly

east or west, it is necessary to compute the actual distance traversed on the “Easting” and “Northing” axes. As an illustration, if the vehicle is moving towards the north-east at an angle of 20 degrees with regard to true north, then the contribution of its displacement on the easting and northing axes is multiplied by $\cos(20^\circ)$ and $\sin(20^\circ)$, respectively. This block includes the code for a “unit circle” which assigns headings into 8 slices of 45° each to reflect their contributions to displacement on the easting and northing axes. This code is provided in the appendix.

Lidar Block – This block collects the data from Lidar and extracts X and Y positions from scanned ranges.

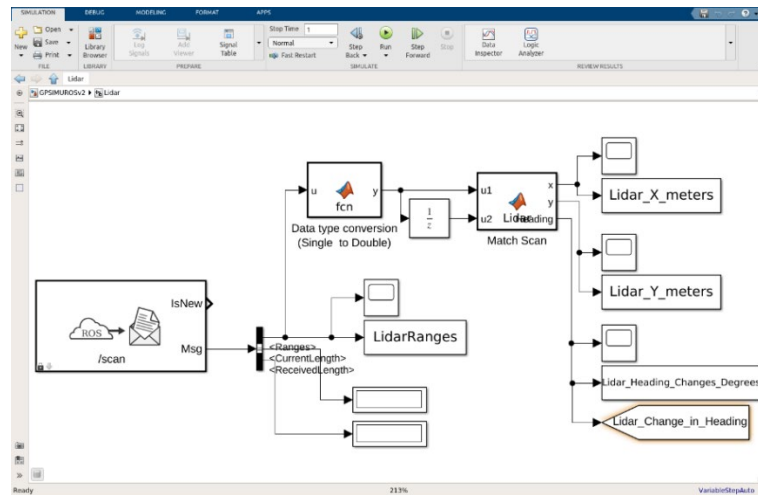


Figure 3-5 Lidar block in Simulink

Since the data provided by the Lidar is a “single data type” a simple Matlab function is used for conversion ($y = \text{double}(\text{lidar_data})$).

Using a “unit delay” block to delay the data one time step, the current data and the data from one time step before are inputted to the next Matlab function called “Match Scan”. The Match Scan function matches two scans of data and outputs the difference of two

scans in terms of change in X(meters), Y(meters), and angle (initially in radians but converted to degrees for use in GNC). The corresponding code is provided in the Appendix.

A “transfer function” is used for Lidar to achieve the velocity as a derivative of positions. Transfer function is another way of taking derivative. The following transfer function, obtained through experimental trials, was used for Lidar.

$$V = \frac{200}{5s^2 + 4s + 2}$$

Equation 3-4

GPS Magnetic Field Block – This is another part of the model in Simulink, exhibited in the next figure.

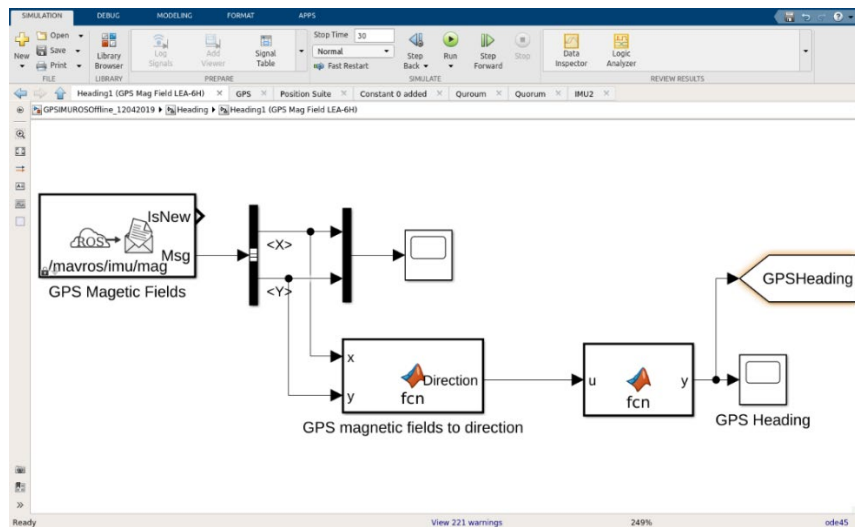


Figure 3-6 GPS Magnetic Field block in Simulink

The first Matlab function covers the magnetic field values to direction. The second Matlab function ensures that all values are within 0 to 360 degrees. All codes are provided in the Appendix.

GNC Block – This block handles all pre-processed outputs coming from the GPS, IMU's, and Lidar blocks and fuses the required data based on the proposed algorithm. An overview of the GNC block is displayed below.

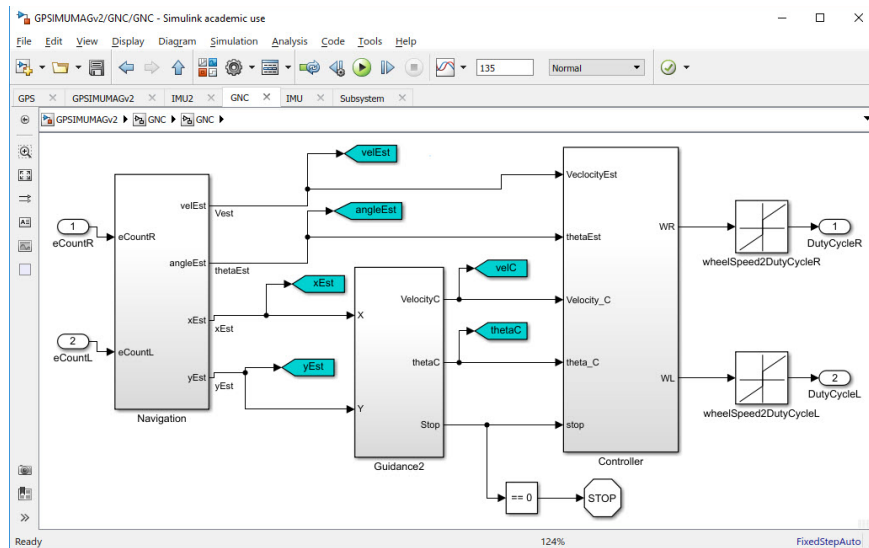


Figure 3-7 GNC block in Simulink

3.3 Sensor Fusion Model Selection

Every two consecutive data values from all sensors are passed through a Matlab function. The first value is the current sensor reading and the second one is the value at one time period before. This is executed using a “delay block”. A delay block delays a signal for as many time periods as the user needs. These two data are differentiated to output the difference of change in value, called delta, for a specified time period. A Matlab function including two consecutive data and a delay block are shown next.

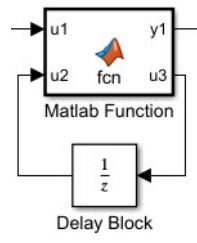


Figure 3-8 Delaying Matlab function and a delay block

The advantage of this approach is that if noisy data causes a sensor to output outlying data, the distance from that particular sensor to other sensors will be large and that sensor will be de-weighted in that specific time step.

For navigation purposes using x and y positions, heading, and velocity, algorithms should be implemented for the pre-processed data fusion in this study. Multi-sensor data fusion which continuously fuses data regardless of their relationships (whether they agree at a particular period of time or not) is also proven to fail in situations, particularly when an interfering factor from the environment distorts the data and causes a noisy incoming data, such that earlier stated where a magnet or a ferrous object deviates an active sensor.

In case of any missing value, a large value such as 1000 is replaced as the value of difference such that algorithm largely de-weights that particular sensor for the time period in which data was missing. Comparisons between the raw outputs, in which all outputs from all sensors are averaged, and the filtered data will be presented in Chapter 4.

According to the above discussions, a decision in each time period is taken collectively. This was earlier referred to as “meta-sensing”. Navigation is handled in hard situations when a sensor transmits no or noisy data by replacing fused estimates through checking other sensors` readings. In addition, this study is dealing with raw data to work

with the worst-case scenario; however, one might be interested in initially filtering out the data on the Pixhawk and feeding them into the algorithms proposed in this research.

3.3.1 Analysis To Direct The Data Fusion Activities

This subsection provides details and an example on how the stated algorithm works. Weights are assigned using the graph theory concept which is explained below.

$$\begin{aligned}
 d_{ij} &= \max(e_i, e_j) - \min(e_i, e_j) & \forall i = 1.2. \dots \quad \forall j = 2.3. \dots \quad i \neq j \\
 D &= \sum_{\substack{i=1.2. \dots \\ j=2.3. \dots}} d_{ij} & \forall i = 1.2. \dots \quad \forall j = 2.3. \dots \quad i \neq j \\
 P_k &= 1 - \frac{d_{ij}}{D} & \forall i = 1.2. \dots \quad \forall j = 2.3. \dots \quad \forall k = 1.2. \dots \quad i \neq j \\
 N_k &= \|P_k\| & \forall k = 1.2. \dots \\
 W_k &= \frac{\sum_{k=1}^{n-1} P_k}{n-1} & \forall k = 1.2. \dots \quad \forall n = 1.2. \dots
 \end{aligned}$$

Equation 3-5

Where e_i and e_j are the delta values for each sensor deducted from each other. Accordingly, d_{ij} would be the magnitude of all possible differences between all deltas. D is the summation of all d_{ij} 's. Considering the sensors as nodes and the deltas edges connecting them based on the graph theory concept, P_k is the magnitude/weight given to a particular edge between nodes i and j . Since P_k is the possibility/weight and the summation of all P_k 's should be 1, then N_k is defined as the normalized value of any P_k . Finally, W_k denotes the weight assigned to each node based on the number of connected edges to a particular node. This method allows each node to get $1/(n-1)^{\text{th}}$ weight of each adjacent edge given there are n nodes in the graph. The above concept is better illustrated in the following example.

In a network of 3 sensors, each sensor acts as a node. Assuming that in a particular time step the delta values (differences in displacement) are 2, 3, and 7, the below figure is the graph representation for this network.

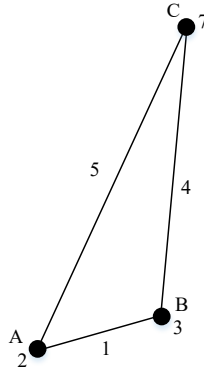


Figure 3-9 An example on the fusion algorithm

The graph ABC includes the vertices A, B, and C as nodes (deltas) and sides AB, AC, and BC as edges. The numbers associated with each node and edge represent the weight of that node and distance between the nodes, respectively. Therefore, we will obtain the values as follow:

$$d_{12(AB)} = \max(2.3) - \min(2.3) = 1$$

$$d_{13(AC)} = \max(2.7) - \min(2.7) = 5$$

$$d_{23(BC)} = \max(3.7) - \min(3.7) = 4$$

$$D = \sum d_{12} + d_{13} + d_{23} = 1 + 5 + 4 = 10$$

$$P_{1(AB)} = 1 - \frac{1}{10} = \frac{9}{10} \quad \cdot \quad P_{2(AC)} = 1 - \frac{5}{10} = \frac{5}{10} \quad \cdot \quad P_{3(BC)} = 1 - \frac{4}{10} = \frac{6}{10}$$

$$N_{1(AB)} = 0.45 \quad \cdot \quad N_{2(AC)} = 0.25 \quad \cdot \quad N_{3(BC)} = 0.3$$

$$W_{1(A)} = \frac{0.45 + 0.25}{2} = 0.35 \quad \cdot \quad W_{2(B)} = \frac{0.45 + 0.3}{2} = 0.375 \quad \cdot \quad W_{3(C)} = \frac{0.25 + 0.3}{2} = 0.275$$

Based on the gained weights for each node (sensor), sensors will get their pertinent weights in that particular time step. Although the simple mean for these sensors is 4, the new mean using this method is 3.75. This means that 7 is given less weight and is recognized as an outlier. The same calculations for the datasets of [2,3,8] and [2,3,9] indicates assigning more weight toward two close numbers (2 and 3) and less weights to the outlier (8 and 9).

Comparisons with more nodes (4 and 5) will be conducted in Chapter 4. The above strategy is applied for velocity dimension where we want to decide how the weights should be distributed over four contributing sensors. The table below shows the final results on four experiments where each of which included four data points from four different sources.

Table 3-2 4-noded graph experimental results

Weight Dataset	w ₁	w ₂	w ₃	w ₄	Dataset Mean	Weighted Mean
[2,3,6,8]	0.2476	0.2571	0.2572	0.2381	4.75	4.7145
[2,3,6,9]	0.25	0.2583	0.25835	0.23335	5	4.9252
[2,5,8,9]	0.23335	0.25835	0.2583	0.25	6	6.0749
[2,4,8,15]	0.25117	0.26046	0.26046	0.22791	7.25	7.0465

As the table shows, when an outlier is farther from the other points, the algorithm gives less weight to that data so that the weighted mean tends to be closer to the other data points.

This algorithm collects the data from all sensors in each time step, dynamically delays, continuously processes them, and yields the output as a single data for a given dimension such as position.

3.4 The Computing Infrastructure Middleware and its Necessity

It is recommended that a cheap, accessible, easy-to-use, non-confounding, extendable, and reliable system be applied that does not result in complicating the algorithms and the processes. Several robot middleware systems such as The Carnegie Mellon Navigation Toolkit (CARMEN), Lightweight Communications and Marshalling (LCM), Microsoft Robotics Studio, and Robot Operating System (ROS) have been introduced for the purpose of code sharing. However, ROS facilitates the simultaneous use of independent modules, such as sensors, as a complex robot control center (Crick et al. 2017). ROS¹ consists of nodes and topics for parallel execution and maintenance of all sensors` freely. The ROS package is installed and run on an Ubuntu platform. The ROS environment is executed by the “roscore” command which is a ROS Master server (node) that saves the names and paths for all the nodes. All other nodes must be registered in this Master node. A node is an agent communicating with the ROS Master and the other nodes via (1) topics (publish/subscribe) using specific message types, (2) services: request/response, like callback functions. The next figure indicates the links between ROS elements.

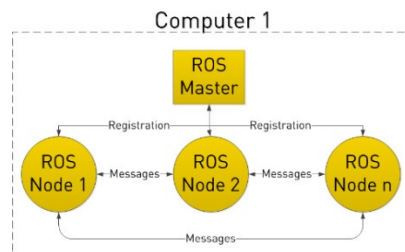


Figure 3-10 ROS communication algorithm (Clearpath Robotics 2014)

¹ A few parts are extracted from materials taught in IE 5379 at UTA in Spring 2018.

Nodes can subscribe to or publish into Topics as shown below.

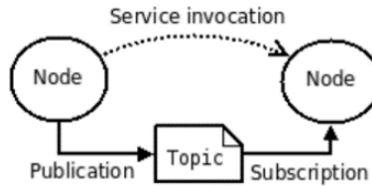


Figure 3-11 Nodes and topic relationship (Martinez Romero 2014)

A robot control system usually comprises many nodes. For example, one node controls a laser range-finder, one node controls the robot's wheel motors, one node performs localization, one node performs path planning, one node provides a graphical view of the system, and so on.

The use of nodes in ROS provides several benefits to the overall system. There is additional fault tolerance as crashes are isolated to individual nodes. Code complexity is reduced in comparison to monolithic systems. Implementation details are also well hidden, even in other programming languages which can easily be substituted.

Topics are named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouple the production of information from their consumption. In general, nodes are not aware of who they are communicating with. Instead, nodes that are interested in data, subscribe to the relevant topic; nodes that generate data, publish to the relevant topic. There can be multiple publishers and subscribers to a topic. The ROS environment also creates asynchronous data distribution which does not interfere with any other transmission by other sensors or during the data processing, more importantly due to the fact that sample rates for all sensors are not the same. This environment will be outlined more in Chapter 4.

3.4.1 Data Processing Architecture and Structure

All the designed algorithms work in numerical format in order to find the difference between two consecutive values. Therefore, if qualitative evidence is to be used along with the other sources, that needs to be digitized to count as a source.

According to the case study used in this research, data is transferred to the Simulink platform as a tool for fusion purposes and storage (if set to store at all) as soon as they are received. Regarding the ROS topics applied, the following provides an overview of what topics the simulation model uses to collect the raw data from sensors.

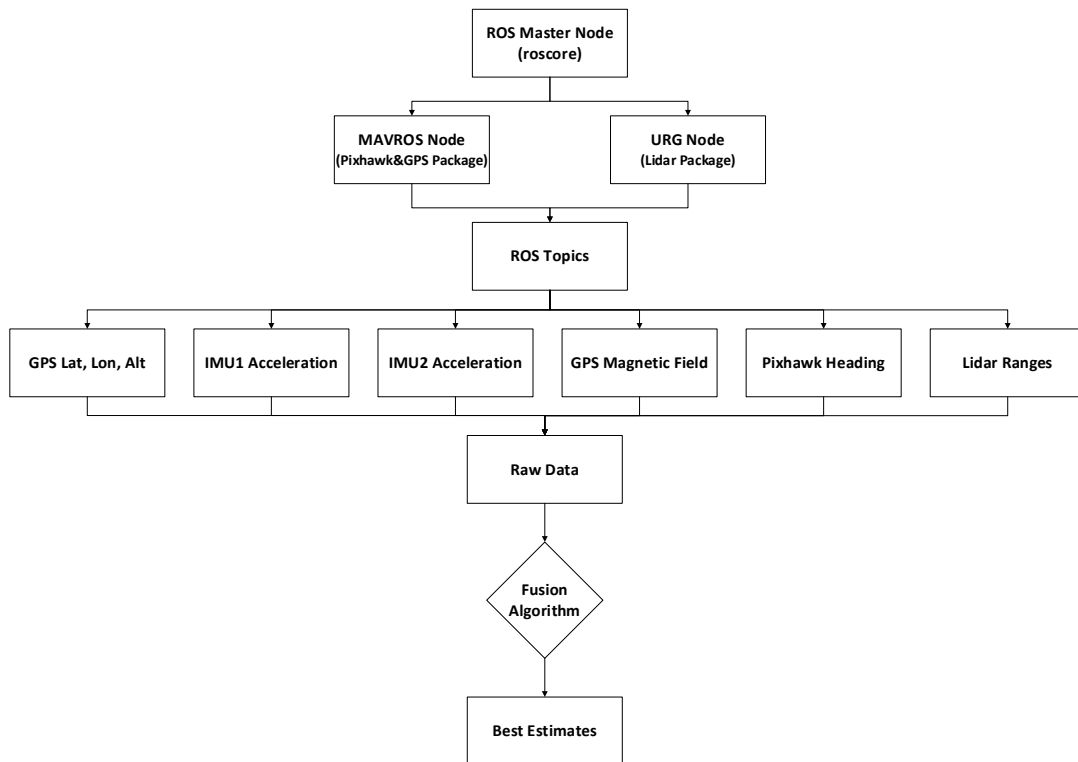


Figure 3-12 ROS nodes and topics applied in the research

ROS Master Node – This is the core part of implementing any other nodes. In other words, this node should be executed first so that other nodes can be configured to finally publish their topics.

MAVROS Node – This node is a package which makes the relationship between the Pixhawk sensors and ROS environment possible.

URG Node – This node provides the access to the data published by Lidar in ROS.

ROS Topics – This includes all topics that are published by executed nodes in ROS. Here, it publishes the topics provided by “MAVROS Node” and “URG Node”.

GPS Lat, Long, Alt – This topic obtains latitude, longitude, and altitude from the GPS sensor. This data is then converted to position values in x, y, and z axes.

IMU1 Acceleration – This topic acquires the acceleration from the first accelerometer. This data is then converted to position values in x, y, and z axes.

IMU2 Acceleration – This topic acquires the acceleration from the second accelerometer. This data is then converted to position values in x, y, and z axes.

GPS Magnetic Field – This topic receives the magnetic fields in x, y, and z axes, and then enables us to convert them to heading angle value.

Pixhawk Heading – This topic obtains the heading angle value from Pixhawk compass using the values from the Pixhawk built-in magnetometer.

Lidar Ranges – This topic provides the ranges to the scanned objects.

Raw Data – This shows the stream of data from the sensors. This sensor data can be subscribed to by any application using Simulink “Subscriber blocks”.

Fusion Algorithms – This implementation block refers to the fusion algorithm used to analyze the data ranging from noise reduction to filtering and evidence-based integration.

CHAPTER 4

DATA ANALYSIS AND RESULTS

This chapter shows the applicability of the previous chapters to better explain the model, its inputs, and outputs. Using measurement sensors, an appropriate software, regular Windows and Ubuntu platforms the experiments are implemented. All of the concepts and terminology explained in Chapter 3 are now combined to show their contribution to analyze the data.

The first section of this chapter introduces the initial experimental setting. The second section shows a new experimental setting in a different environment in which more reliable outputs are drawn compared to the first experiment. The first experiment is described as follows.

4.1 Software and Hardware Configurations

The IMU sensors are calibrated using the Mission Planner software (version 1.3.62) on a Windows platform as shown below to ensure an appropriate accuracy in the area, specifically not to be largely affected by magnetic interference.

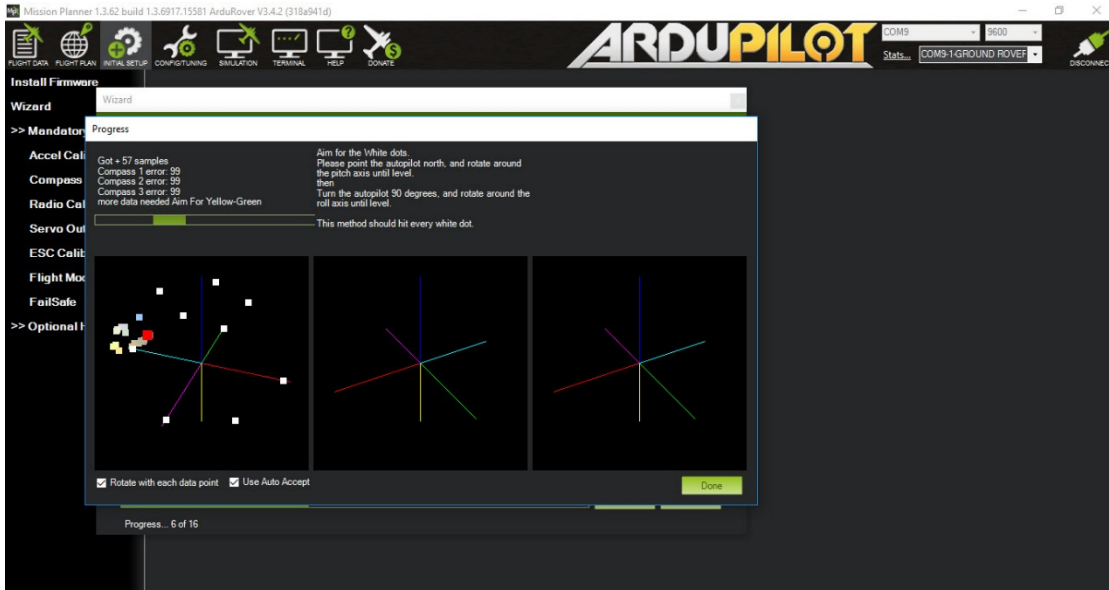


Figure 4-1 Sensor calibration process in Mission Planner software

GPS, IMU1 and IMU2 data are read and subscribed to ROS, then published to Simulink and Matlab. A designed scenario for this research is moving on a predetermined path beside the UTA campus in an open area with a starting point at a latitude of 32.73441 and a longitude of -97.12179. The sky was clear and the GPS sensor could get the required signals to determine the location. The required hardware for this research is configured as described below.

Pixhawk – The Pixhawk was developed by an independent, open-hardware project to provide high-end autopilot hardware to the academic, hobby and industrial communities at low costs and high availability (Meier 2019). The sensors are connected and handled through this board. A Pixhawk contains two IMU's. Each IMU has one gyroscope and one accelerometer (both measuring 3-axes). There is also one magnetometer and one barometer inside the Pixhawk. The next figure displays a Pixhawk.



Figure 4-2 Pixhawk (PX4)

GPS – The U-Blox sensor (LEA-6H-0-002) is used for this study which includes both a GPS output and a magnetometer. This sensor is known for a 2-2.5m horizontal position accuracy, and a 9600 baud rate. Several tests in various fields show consistent results from this sensor.



Figure 4-3 U-Blox GPS sensor

4.2 Evidence-Based Data Fusion

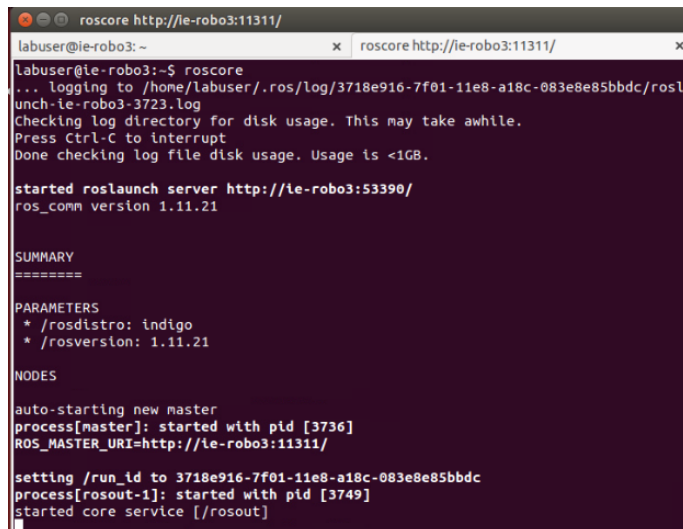
By simply averaging the outputs from all sensors the results can be noisy since one noisy sensor can distort good outputs from the others. Accordingly, different criteria are provided to evaluate the data streams from sensors as follows. Codes are provided in the Appendix.

The deltas are all compared with each other to find out which deltas are closer to each other. This specifies their contribution. Summation of all these weights adds up to 1. At last, each weight is multiplied by its contributing sensor's delta value. This is called the fused estimate at that particular time step.

4.3 Real-Time Data Collection In ROS

A real-time data collection scenario is set up such that IMU2 transmits all noisy data so that the fusion algorithm can represent its capability to ignore that. Regarding the conversions from acceleration to velocity and displacement in IMU's, customized filters are applied. Not properly filtering acceleration values produces drift in the computed velocity and displacement. Accelerometers are usually not able to record very low frequency signals. Drifting is more considerable when integrating acceleration and velocity; therefore, the main issue would be low or very low frequencies while noise is considered as a high frequency signal. Customized filters refer to a block or a structure to filter out a value based on a fact as discussed in section 3.2.3.

Since the GPS sensor, IMUs, and digital compass are a part of the Pixhawk autopilot system, a software package called "MAVROS" was used with ROS to transfer data between the Pixhawk and Simulink. The implementation code and screenshots of both are displayed next.



```
labuser@ie-robo3:~$ roscore
... logging to /home/labuser/.ros/log/3718e916-7f01-11e8-a18c-083e8e85bbdc/ros-l
unch-ie-robo3-3723.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ie-robo3:53390/
ros_comm version 1.11.21

SUMMARY
=====
PARAMETERS
 * /roscpp: indigo
 * /rosversion: 1.11.21

NODES

auto-starting new master
process[master]: started with pid [3736]
ROS_MASTER_URI=http://ie-robo3:11311/

setting /run_id to 3718e916-7f01-11e8-a18c-083e8e85bbdc
process[rosout-1]: started with pid [3749]
started core service [/rosout]
```

Figure 4-4 ROS start-up screen in a terminal on Ubuntu

```

/opt/ros/indigo/share/mavros/launch/px4.launch http://localhost:11311
labuser@ie-robot3:~$ roslaunch mavros px4.launch
... logging to /home/labuser/.ros/log/3718e916-7f01-11e8-a18c-083e8e85bbdc/rosla
unch-ie-robot3-3863.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ie-robot3:51997/

SUMMARY
=====
CLEAR PARAMETERS
* /mavros/

PARAMETERS
* /mavros/cmd/use_comp_id_system_control: False
* /mavros/conn/heartbeat_rate: 1.0
* /mavros/conn/system_time_rate: 1.0
* /mavros/conn/timeout: 10.0
* /mavros/conn/timesync_rate: 10.0
* /mavros/distance_sensor/hrlv_ez4_pub/field_of_view: 0.0
* /mavros/distance_sensor/hrlv_ez4_pub/frame_id: hrlv_ez4_sonar
* /mavros/distance_sensor/hrlv_ez4_pub/id: 0
* /mavros/distance_sensor/hrlv_ez4_pub/orientation: ROLL_180
* /mavros/distance_sensor/hrlv_ez4_pub/send_tf: True
* /mavros/distance_sensor/hrlv_ez4_pub/sensor_position/x: 0.0
* /mavros/distance_sensor/hrlv_ez4_pub/sensor_position/y: 0.0

```

Figure 4-5 MAVROS package start-up screen in a terminal on Ubuntu

To obtain sensor data from the Pixhawk, subscriber blocks are created in each subsystem used in the Simulink model. A subscriber block is a gateway that can obtain sensor data and then the model can use that data. A subscriber block to get GPS coordinates is shown below.

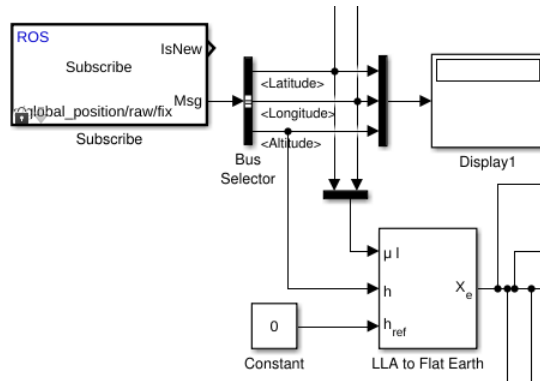


Figure 4-6 A representation of a subscriber block

The subscriber blocks ask for data published by ROS topics. Data from GPS and IMUs are collected for 1.2 mins in a square area as indicated next.

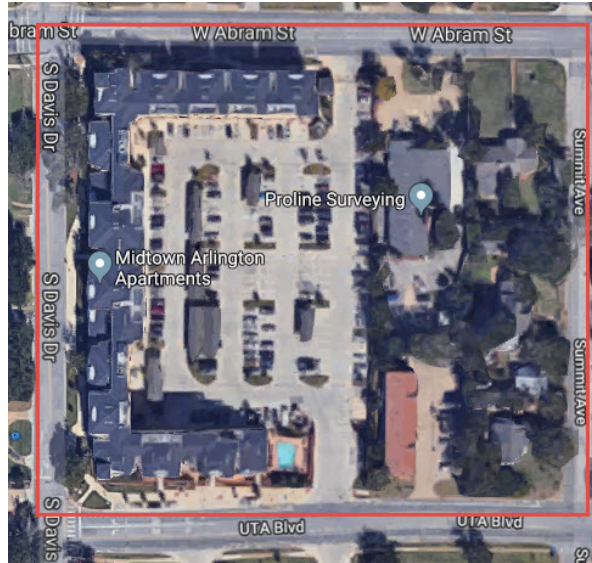


Figure 4-7 Experiment 1 run

The vehicle started from the bottom right corner of the image, at the intersection of UTA Blvd and Summit Ave, and traversed a square path to reach to its start point. Each edge of this area is around 150 m long.

4.4 Results and Comparisons – Experiment 1

This section discusses the results gained from the aforementioned fusion methodology as well as comparisons and comments which are provided as a conclusion. Results for each input of the navigation system including easting and northing positions (relative to true north), heading angle, and velocity are addressed separately. All codes are provided in Appendix.

4.4.1 Easting Position

Raw and truth outputs for the Easting (Y) position under normal conditions are shown in Figure 4-8.

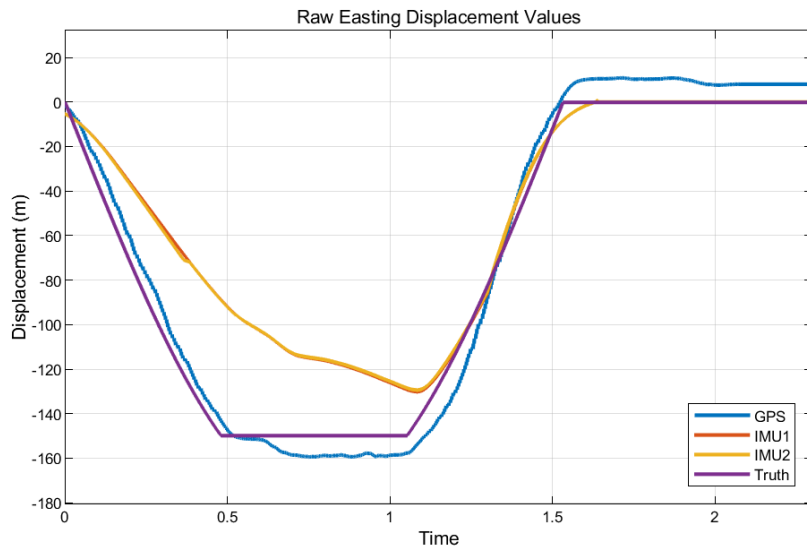


Figure 4-8 Truth and raw values of all sensors for Easting (Y) axis – normal conditions

As shown, the GPS values are closer to the truth and the IMU values are not perfectly reflecting the truth due to noise in IMU's blocks. Figure 4-9 shows the fused values for the easting position.

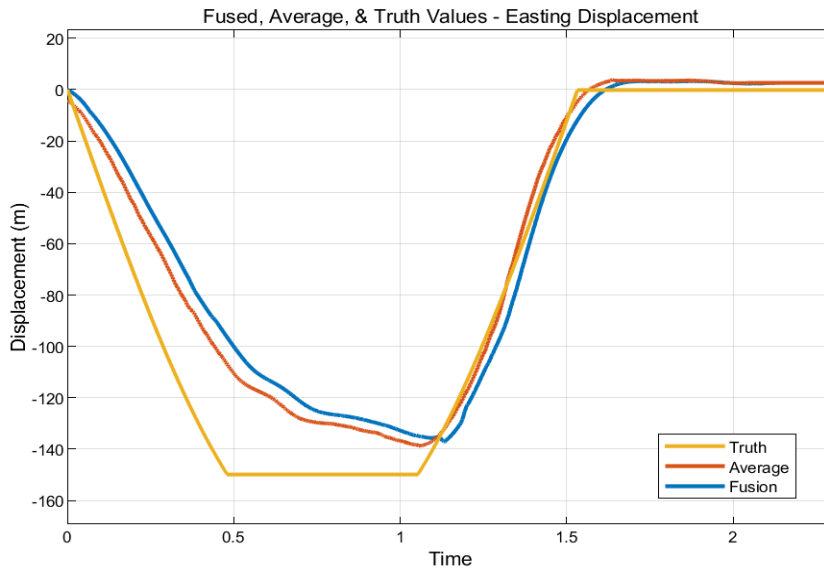


Figure 4-9 Fused, truth, & raw (averaged) values of Easting (Y) axis – normal conditions

Under normal conditions, fusion algorithm works fine and yields close estimates as raw (averaged) values. Subsequent to the discussion under normal conditions, a severe scenario will be addressed under noisy circumstances as follows.

In this scenario the sensor outputs are overshadowed by environmental and destructive factors that cause drift and noise. In the defined scenario, IMU2 was given noisy signals with a mean of 5 meters and a variance of 10 meters as shown in figure 4-10. That distorts the output but was ignored when the fusion methodology was applied. In conclusion, the researcher's results improved the progressed displacement estimates. This included fixing data drifts and the noise of converted data The IMU2 noisy signal is displayed in Figure 4-10.

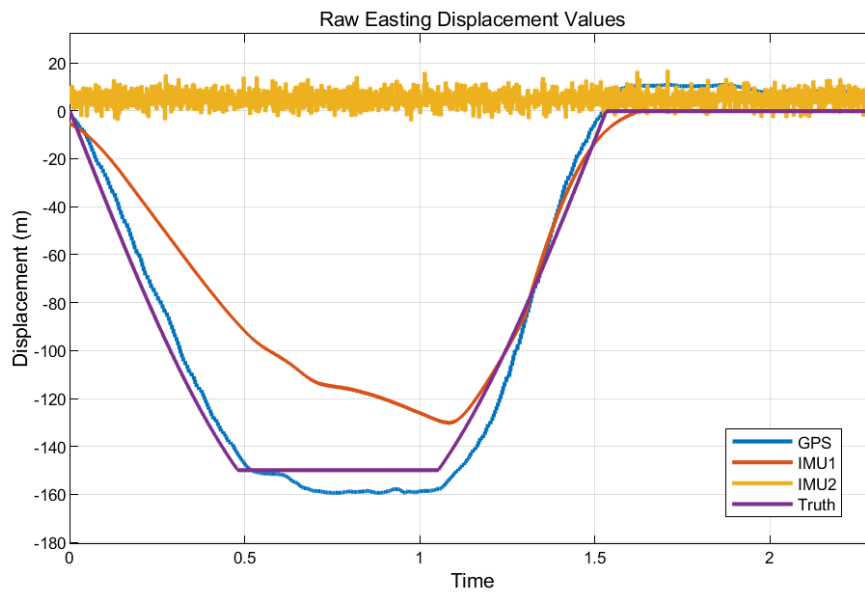


Figure 4-10 Truth and raw values of all sensors for Easting (Y) axis – IMU2 noisy

In figure 4-11, the raw data from all three sensors were averaged to combine all outputs together. It can clearly be seen that the raw data from GPS, IMU1, and the IMU2 noisy data cannot estimate a true output. This indicates that the averaging method failed

to display what occurred to the vehicle. When the fusion methodology was applied, the artificially noisy values from IMU2 were heavily discounted by the algorithm. In conclusion, the proposed data fusion method improved the displacement estimates. This included fixing data drifts and the errors inserted by the IMU2 noisy signals.

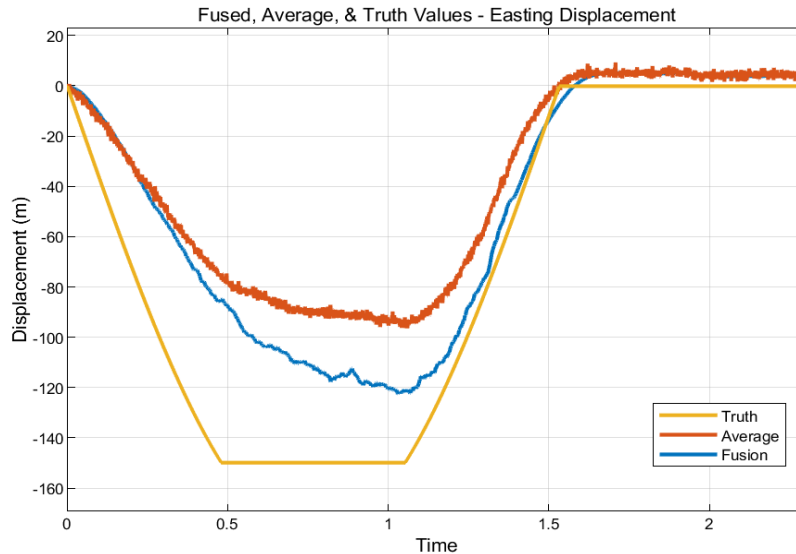


Figure 4-11 Fused, truth, & raw (averaged) values of Easting (Y) axis – IMU2 noisy

It can clearly be seen that the raw data from GPS, IMU1, and IMU2 noisy data cannot estimate a true output. Raw data from all three sensors were averaged to combine all outputs together. However, averaging method failed to display what occurred to the vehicle.

4.4.2 Northing Position

Raw and truth outputs for the Northing (X) position under normal conditions are shown in the next figure.

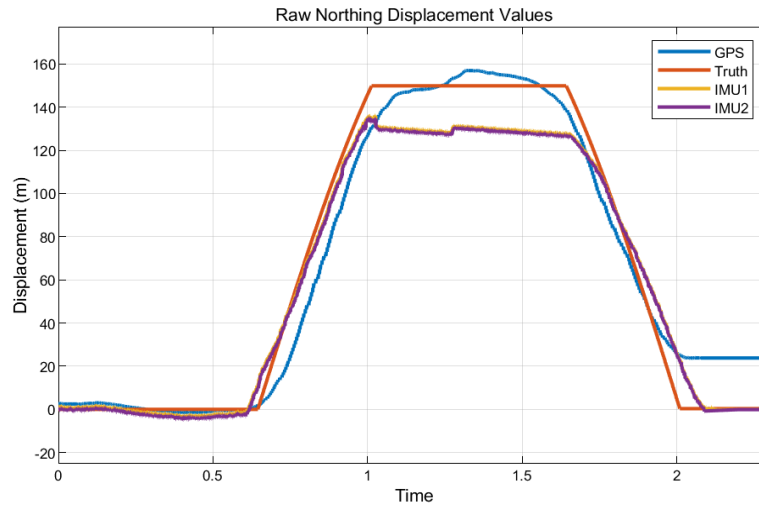


Figure 4-12 Truth & raw values of all sensors for Northing (X) axis – normal conditions

As shown, GPS is closer to the truth and IMU's values do not perfectly reflect the truth values due to noise in the IMU's data processing blocks. The next figure shows the fused values of the northing position.

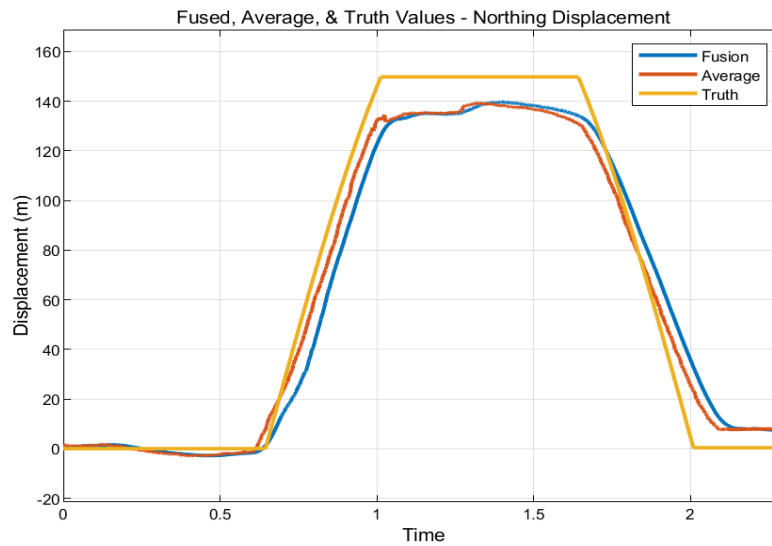


Figure 4-13 Fused, truth, & raw (averaged) values of Northing (X) axis – normal conditions

Under normal conditions, the fusion algorithm works fine and yields similar estimates when compared to the raw (averaged) values. Subsequent to the discussion under normal conditions, a severe scenario will be addressed under noisy circumstances as follows.

In many situations in which sensor outputs are overshadowed by environmental and destructive factors that cause drift or noise, obtaining stable estimates can be better achieved using the algorithm discussed above.

This can be demonstrated by defining a scenario where the IMU2 sensor is again assigned noisy signals with a mean of 5 meters and a variance of 10 meters (figure 4-14).

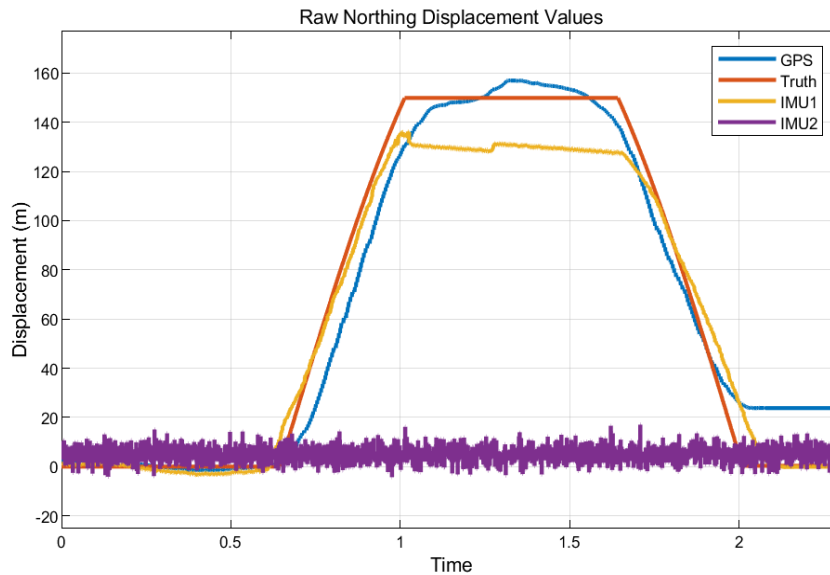


Figure 4-14 Truth and raw values of all sensors for Northing (X) axis – IMU2 noisy

Raw data from all three sensors were averaged to combine all outputs together. However, the averaging method failed to display what occurred to the vehicle. Once again, it can clearly be seen that the raw data from the GPS and IMU1 when combined with the noisy IMU2 data cannot estimate a true output (figure 4-15).

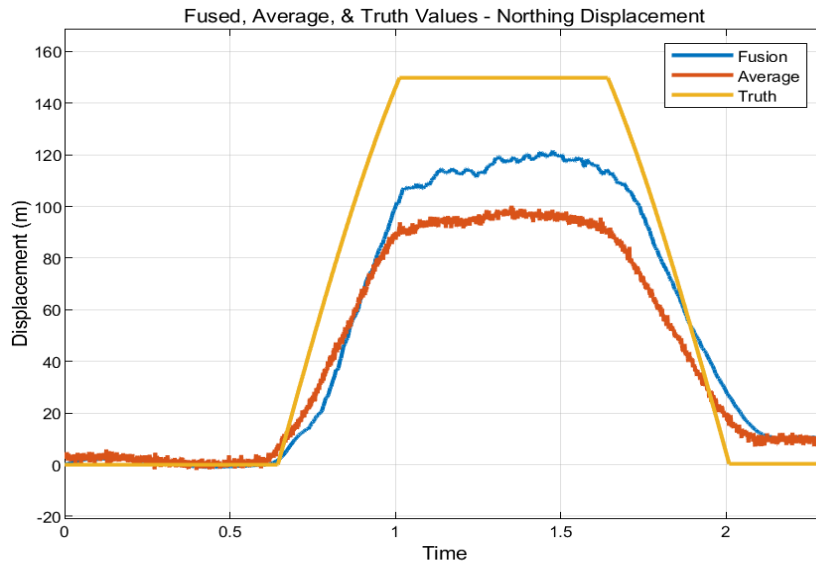


Figure 4-15 Fused, truth, & raw (averaged) values of Northing (X) axis – IMU2 noisy

When the proposed fusion method was applied, the noisy signal values attributed to IMU2 were again heavily discounted. As a result, more robust displacement estimates were obtained. This included adjusting data drifts and the noise of converted data (figure 4-15).

4.4.3 Heading Angle

To calculate the Heading Angle, Pixhawk internal magnetometer, GPS internal magnetometer, and converted latitude and longitude from the GPS have been utilized. Raw and truth outputs for the heading angle under normal conditions are shown in the next figure.

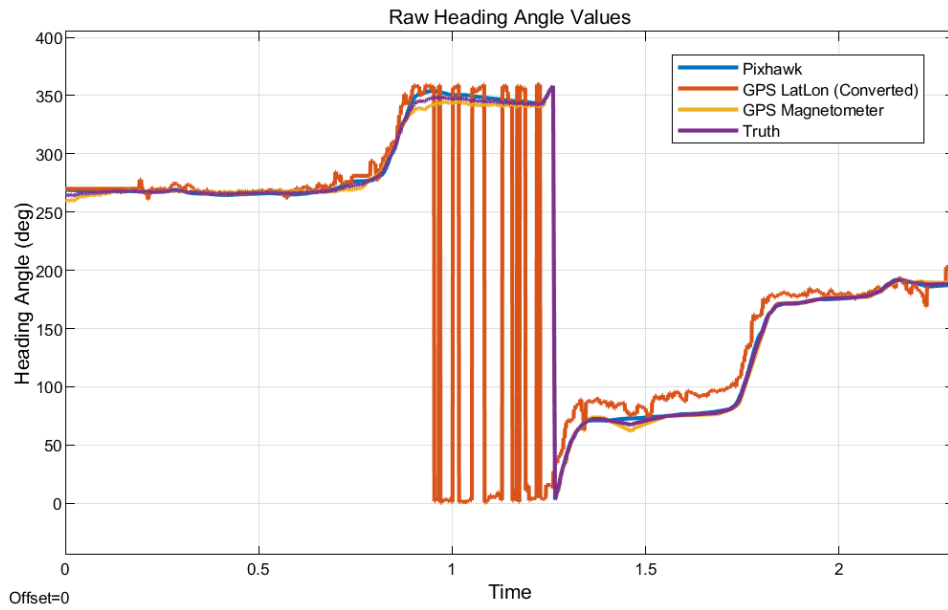


Figure 4-16 Truth and raw values of all sensors for Heading Angle – normal conditions

Under normal conditions, the fusion algorithm generates closer estimates than the raw (averaged) values (figure 4-17). We again simulated the condition where corrupted sensor values were introduced into the system.

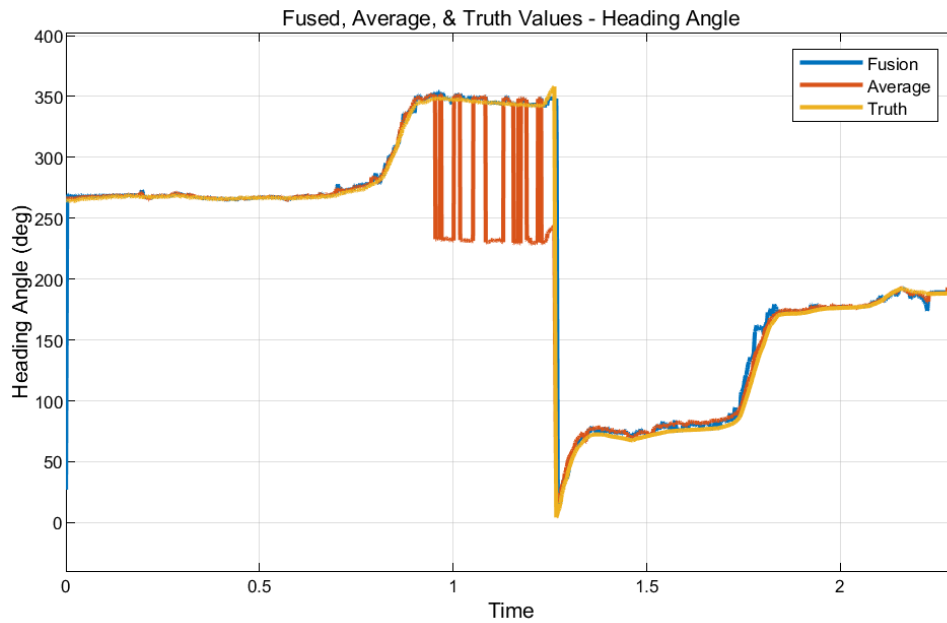


Figure 4-17 Fused, truth, & raw (averaged) values of Heading Angle – normal conditions

In many situations in which sensor outputs are overshadowed by environmental and destructive factors that cause drift or noise, obtaining stable estimates can be better handled using the proposed fusion algorithm. In the defined scenario, Heading Angle from “GPS LatLon” source contained noisy signals with a mean of 5 degrees and a variance of 10 degrees. That could distort the output but was ignored when the fusion methodology was applied. Hence, more robust displacement estimates were achieved. This included adjusting data drifts and the noise of converted data. The GPS noisy signal is displayed in the figure 4-18.

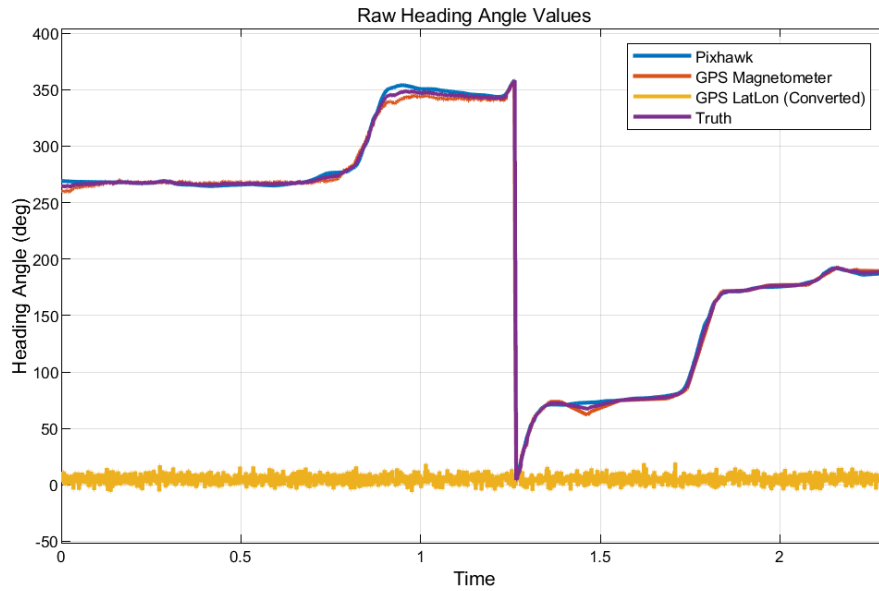


Figure 4-18 Truth and raw values of all sensors for Heading Angle – GPS (latlon converted) noisy

With noisy data from GPS (latlon converted) next figure describes the calculations of the algorithm.

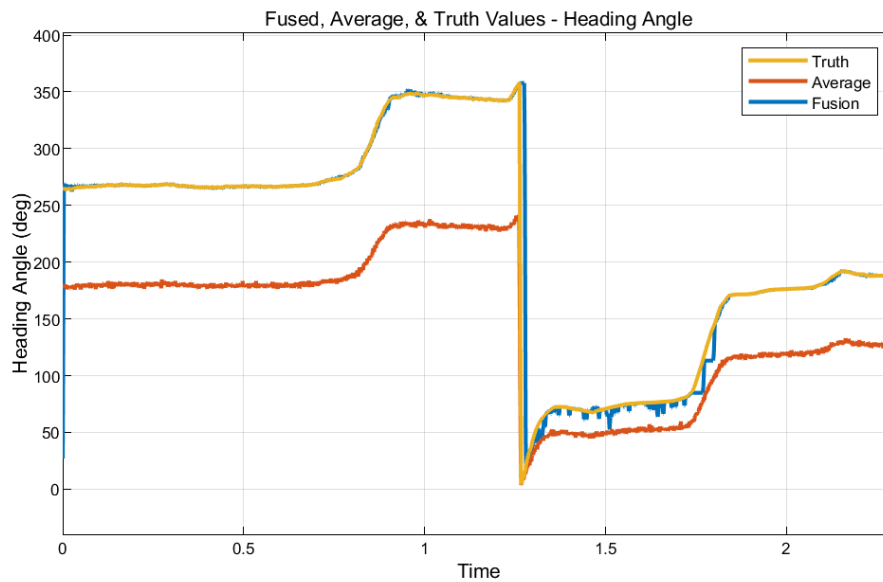


Figure 4-19 Fused, truth, & raw (averaged) values of Heading Angle – GPS (latlon converted) noisy

It can clearly be seen that the raw data from Pixhawk's magnetometer, GPS magnetometer, and GPS (latlon converted) noisy data cannot estimate a true output. Raw data from all three sensors were averaged to combine all outputs together. However, the averaging method failed to display what occurred to the vehicle.

4.4.4 Velocity

The raw and the truth outputs for the Velocity under normal conditions are shown in figure 4-20.

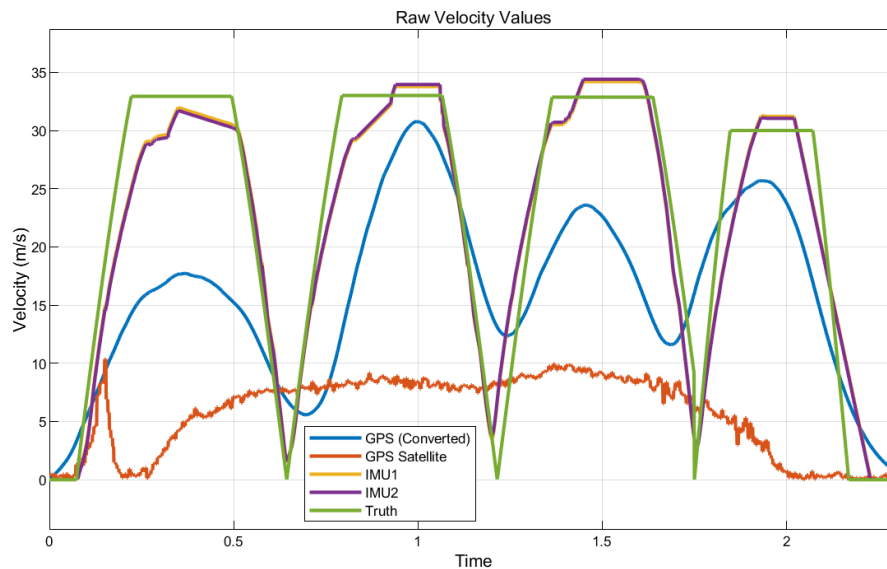


Figure 4-20 Truth and raw values of all sensors for Velocity – normal conditions

As shown, the IMU's values are closer to the truth and the GPS values do not perfectly reflect the truth due to the use of the state-space blocks to take the derivatives of the position values for GPS (converted). The GPS satellite also yielded unreliable data as it was not expected to output reliable values. Figure 4-21 shows the fused values of velocity.

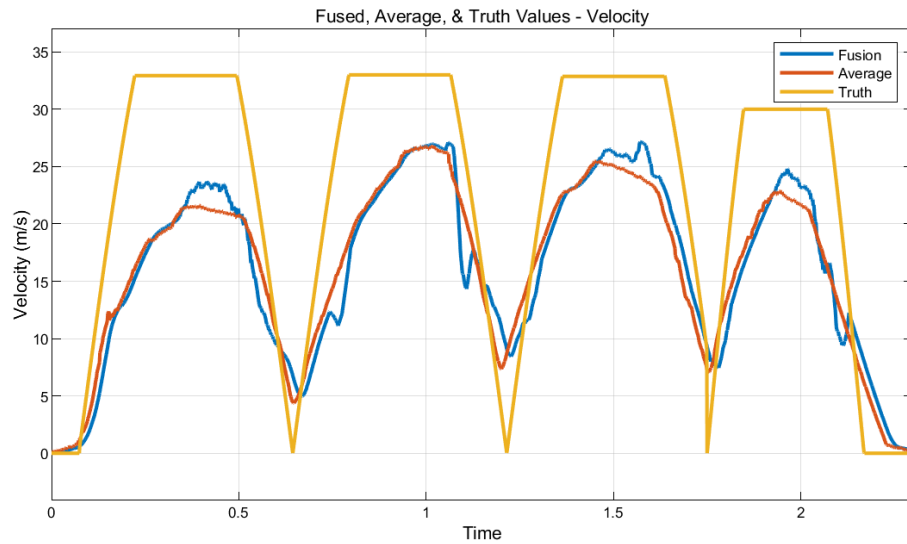


Figure 4-21 Fused, truth, & raw (averaged) values of Velocity – normal conditions

Under normal conditions, the fusion algorithm produces estimates that are similar to the averaged raw values. Subsequent to the discussion under normal conditions, a severe scenario will be described under noisy circumstances as follows.

In many situations in which sensor outputs are overshadowed by environmental and destructive factors that cause drift or noise, obtaining stable estimates can be better handled using the algorithm discussed above. In the defined scenario, IMU2 contained noisy signals with a mean of 5 and a variance of 10 that could distort the output but was ignored when the fusion methodology was applied. Hence, more robust displacement estimates were achieved. This included adjusting for data drifts and the noise of converted data. The IMU2 noisy signal is displayed in the next figure.

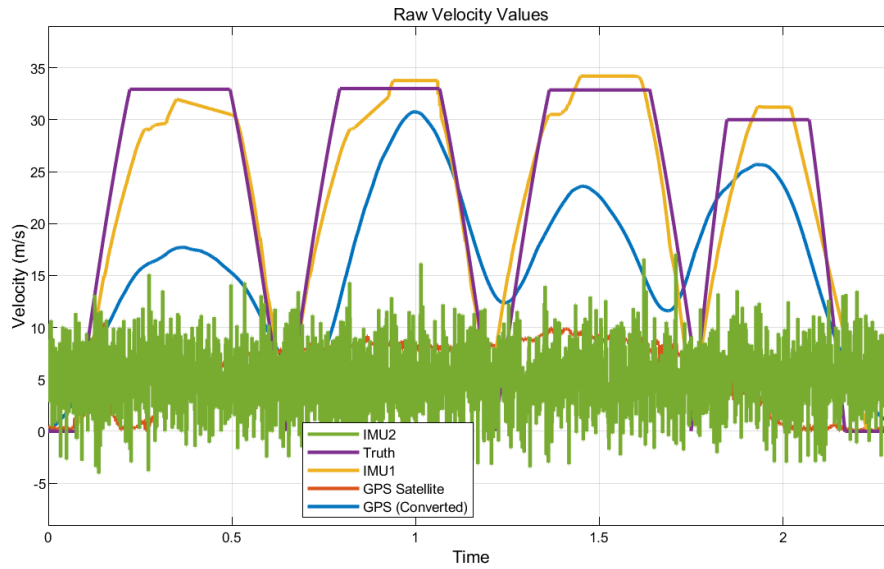


Figure 4-22 Truth and raw values of all sensors for Velocity – IMU2 noisy
 With noisy data from IMU2 the next figure shows how the algorithm works.

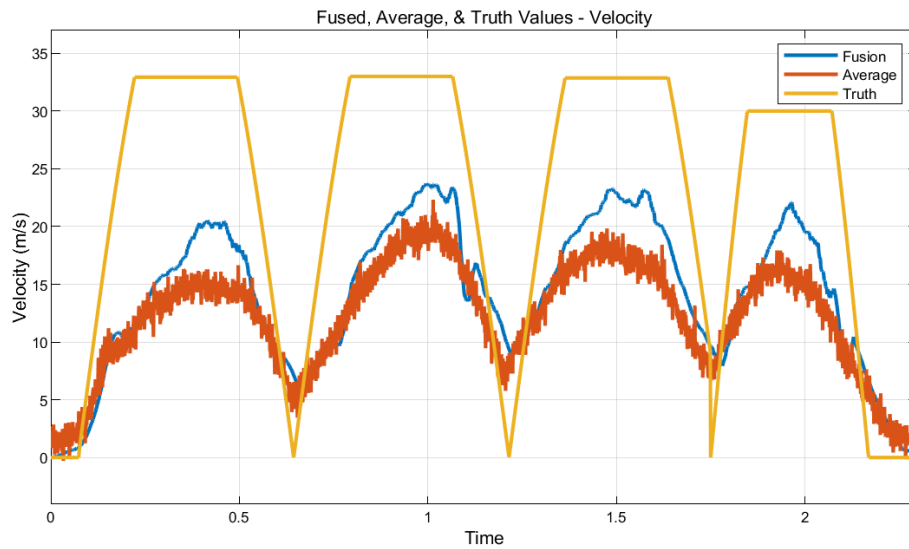


Figure 4-23 Fused, truth, & raw (averaged) values of Velocity– IMU2 noisy
 It can clearly be seen that the raw data from GPS (converted), GPS (satellite), IMU1, and IMU2 noisy data cannot estimate a true output. The raw data from all four sensors

were averaged to combine all outputs together. However, the averaging method failed to display what happened to the vehicle. In this scenario not only did the IMU2 sensor readings contain noise but GPS (satellite) yielded such distorted data that it was basically considered as noise. Under these conditions with multiple sources of poor data, the estimates generated by the fusion algorithm provided better estimates than the averaged values.

4.5 Results and Comparisons – Experiment 2

This experiment uses additional sensors to avoid failure modes occurring from the same sensors which happened in the first experiment. The sensors include: Pixhawk (fused values), magnetometer (raw magnetic fields), GPS, and Lidar. The purpose of implementing a new experiment is to monitor the outcome of the fusion when additional sensors are applied, and an environmental noise factor causes a sensor to generate false outputs or data is missing. The experiments were conducted on the fourth floor of Woolf Hall on the UTA Campus in an area where air handling equipment produce significant distortions to the earth's magnetic fields. This electromagnetic noise adversely affects the outputs of several of the sensors commonly found on inexpensive mobile platforms. This was particularly the case for the magnetometers used in the specific experiments described in this work. The test area for these experiments, which consisted of a 14-meter straight path down a hallway directly past the air handling equipment, was intentionally selected. In addition to this electromagnetic interference, sensors may still fail, or have missing values, at times due to reasons such as internal error. The experimental run is shown in figure 4-24.



Figure 4-24 Experiment 2 run

Five scenarios are considered in this experiment. Each scenario follows the structure of previous experiment which consists of monitoring the fused estimates on Easting, Northing, Heading, and Velocity. They are also simulated using the GNC block described in section 3.2.3. Sensors used in each dimension are indicated below.

Easting and Northing contributing sensors: IMU1 (converted position from acceleration), IMU2 (converted position from acceleration), GPS (converted position from latitude and longitude), and Lidar (converted position from ranges and angles).

Heading contributing sensors: Fused heading from the Pixhawk (from two internal IMUs), Lidar, Gyro, IMU1 magnetometer(raw), IMU2 magnetometer(raw), and GPS magnetometer(raw).

Velocity contributing sensors: GPS (converted velocity from latitude and longitude), GPS (direct velocity from satellites), IMU1 (converted velocity from acceleration), IMU2 (converted position from acceleration), and Lidar (converted velocity from ranges and angles).

The following clarifies possible misunderstandings/discrepancies in the figures subsequent to these notes.

- It is possible that one type of data from one sensor contributes more/less often than another type of converted data from the same sensor. An example of this may be seen in the Easting and Northing displacements from the IMU1 sensor. While IMU1 may contribute low/high in Easting, it may contribute differently in Northing. Although a sensor (such as IMU1) may have noisy data, how that data aligns with other sensors is the fundamental fact for which the weight is assigned.
- Easting and Northing values gained from IMU1 and IMU2 depend on the fused heading in each scenario as explained in section 3.2.3. This is why their values may be different from one scenario to another.
- The average value is shown in all scenarios for comparison purposes and remains the same. This signifies that the average signal in each scenario shows the average of all values presented in Scenario 0.
- In order to simulate the results in the GNC block, Heading values are converted to radians. Heading values shown in all scenarios are in degrees for better understanding but they are converted to radians (multiplied by $\text{Pi}/180$) when used in the Simulink environment. This is for consistency purposes (GNC with other parts) in Simulink and has no effect on the results.

4.5.1 Scenario 0 - Average

This scenario simply uses the average of raw values from all sensors. There is no fusion or algorithm happening in this scenario. Each sensor contributes individually. Scenario 0 intends to show what occurs when all inputs are combined without intelligence.

Figure 4-25 shows Easting displacement.

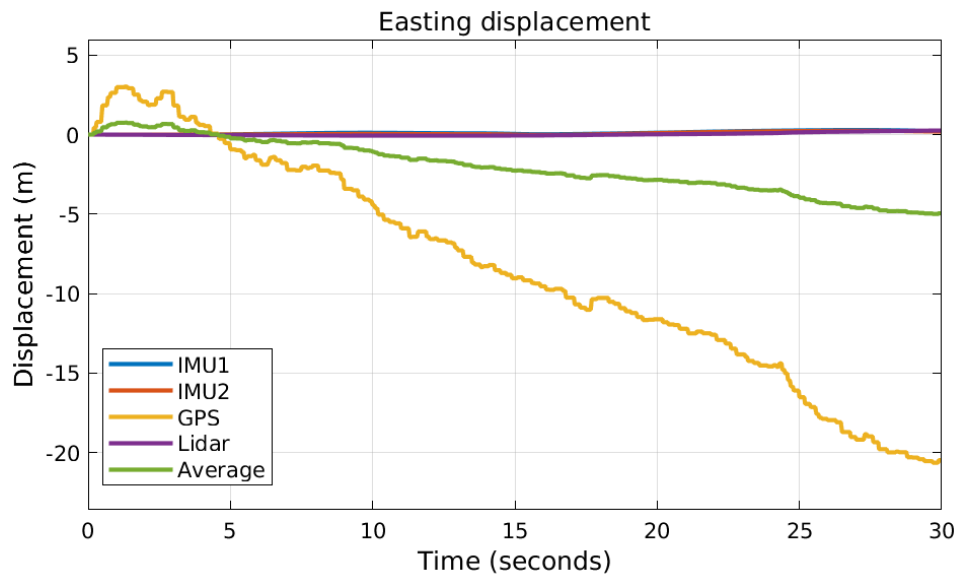


Figure 4-25 Easting displacement – Scenario 0

To better illustrate the graph, a scaled Easting plot is shown below.

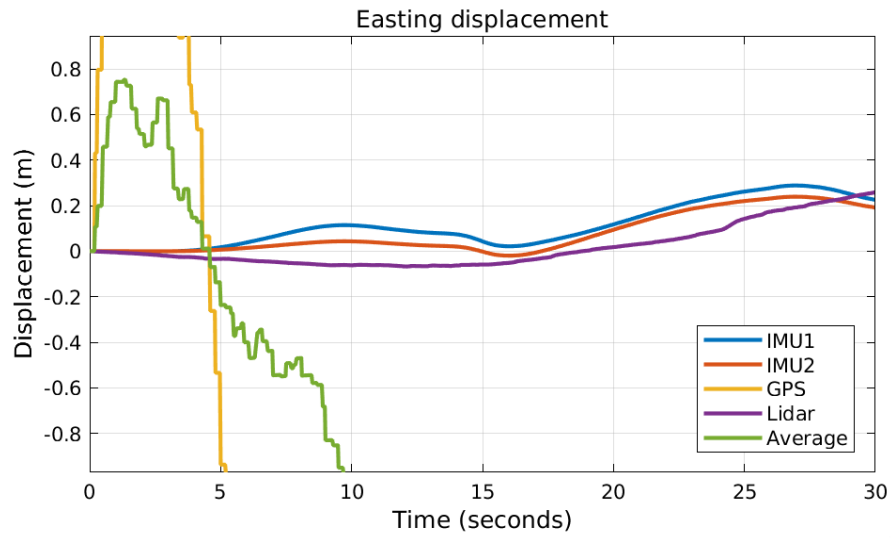


Figure 4-26 Scaled Easting displacement – Scenario 0

In Easting displacement, the average value of all sensors shows 5m movement toward west. However, the expected traversed distance by the vehicle is almost zero.

Figure 4-27 shows Northing displacement.

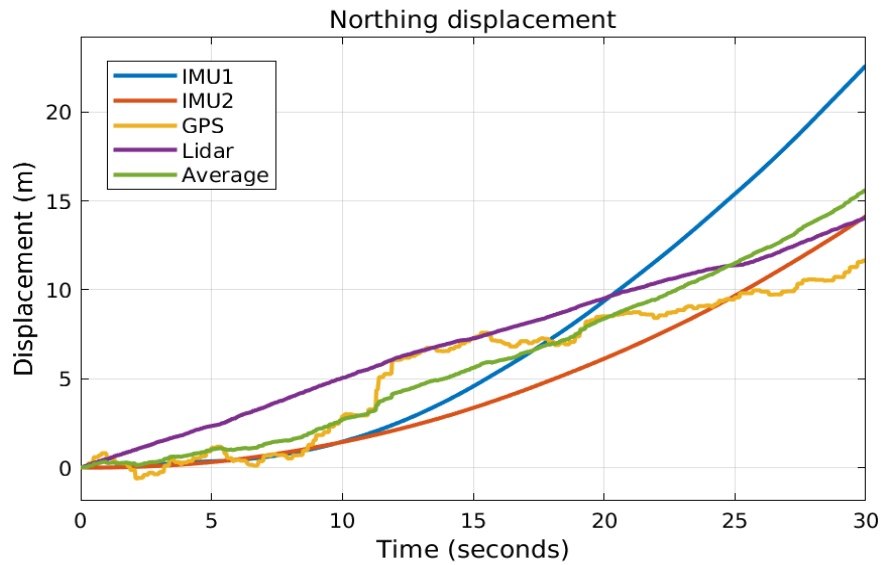


Figure 4-27 Northing displacement – Scenario 0

In Northing displacement, the average value of all sensors shows 15.5m movement toward north. However, the expected traversed distance by the vehicle is almost 14m.

Figure 4-28 shows Heading Angle.

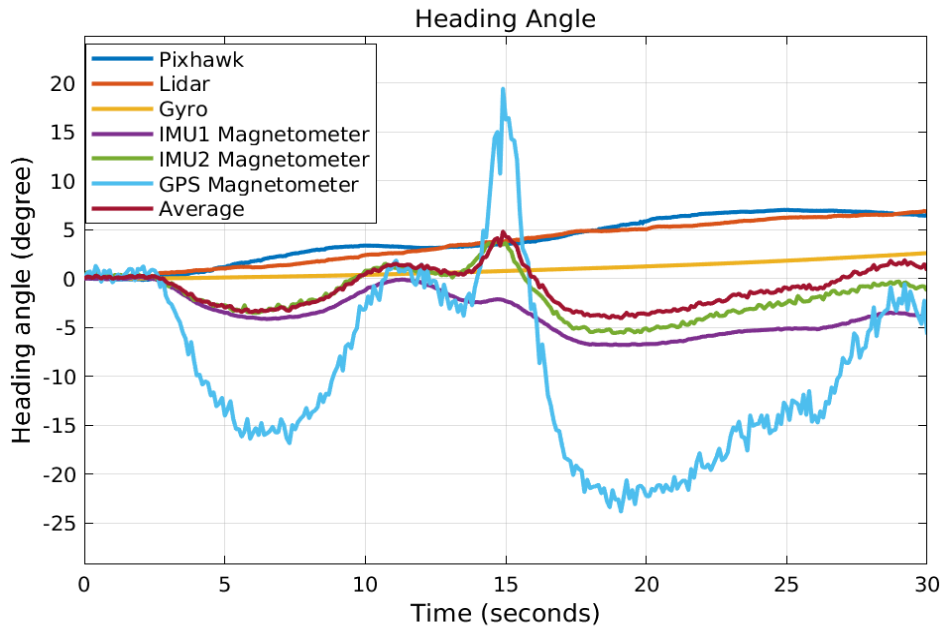


Figure 4-28 Heading angle – Scenario 0

To better illustrate Gyro's values, a scaled Heading plot is shown next.

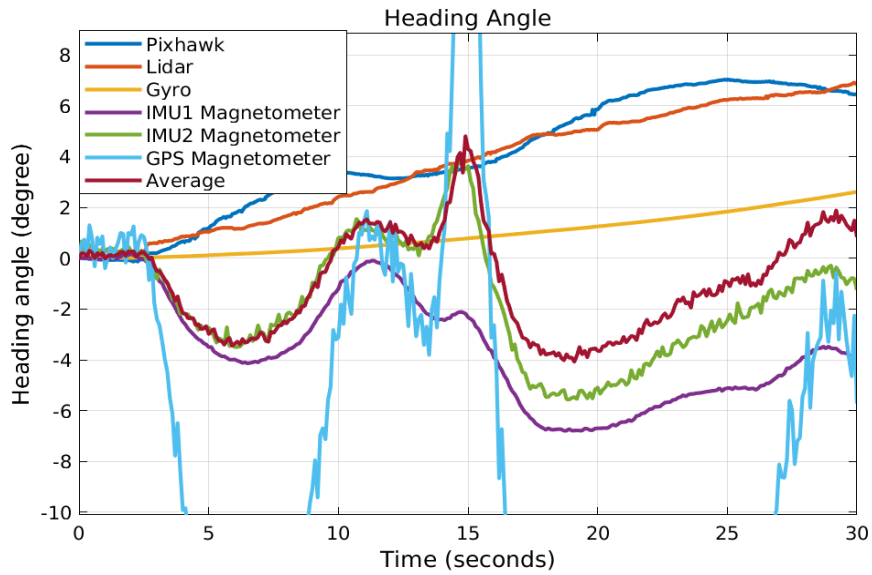


Figure 4-29 Scaled Heading angle – Scenario 0

In Heading Angle, the average value of all sensors shows a total range of 9 degrees in the entire run including many fluctuations.

Figure 4-30 shows Velocity.

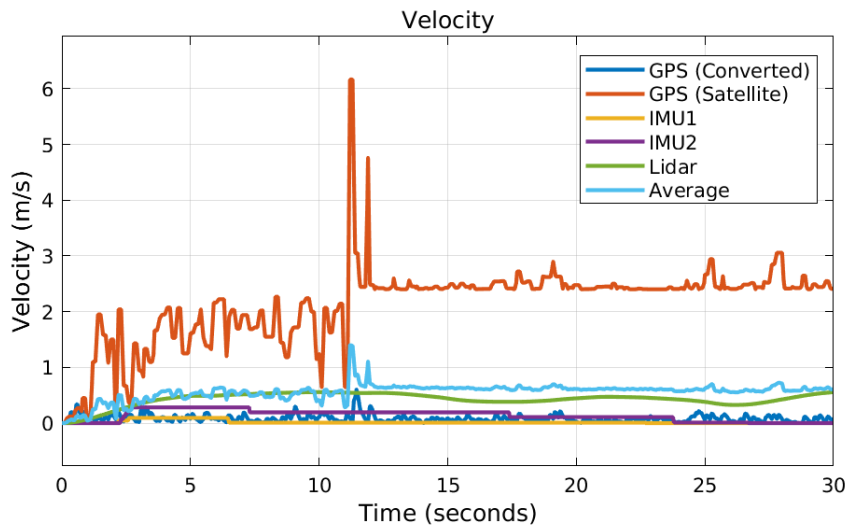


Figure 4-30 Velocity – Scenario 0

To better illustrate the previous graph, a scaled Velocity plot is shown below.

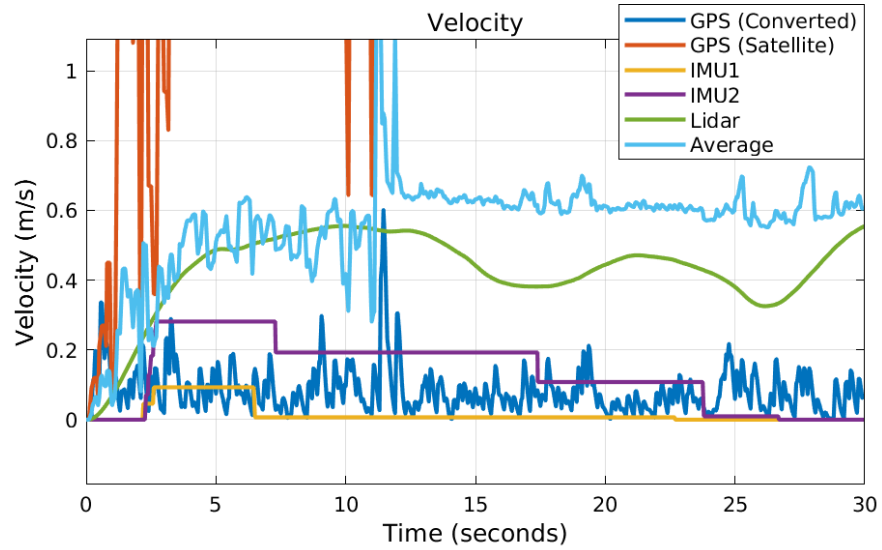


Figure 4-31 Scaled Velocity – Scenario 0

In Velocity, the average value of all sensors shows fluctuating values. However, it is expected that the velocity would be more constant.

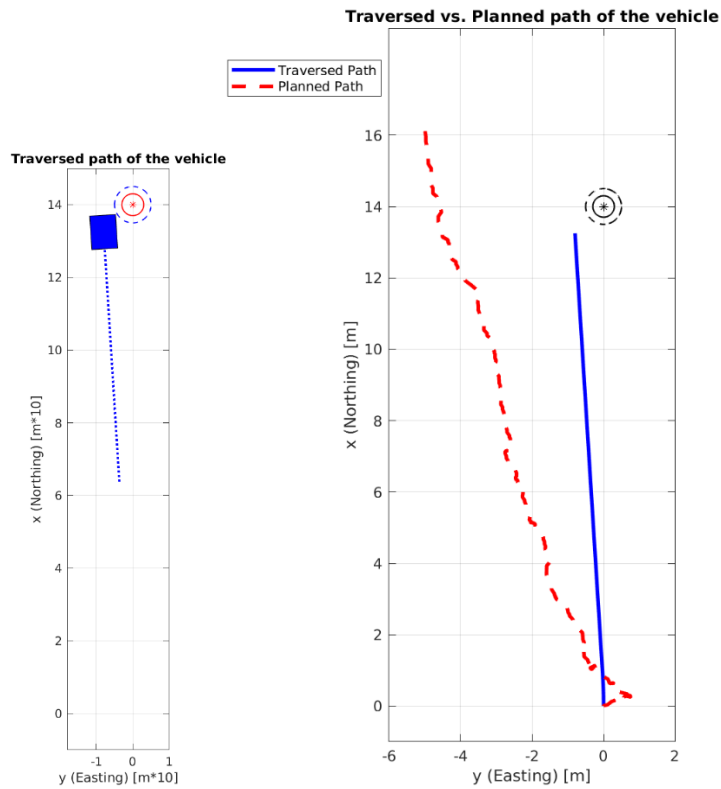


Figure 4-32 Animated traversed path – Scenario 0

As the figures in scenario 0 show, using the average of all sensors leads to the wrong traversing path by the vehicle. The planned path using only the obtained Easting and Northing displacements, and Velocity is shown as a red dotted line in figure 4-32. The traversed path using the modified Easting and Northing displacements, Heading angle, and Velocity is shown as a solid blue line in figure 4-32. The traversed path applies modifications to the values for Easting and Northing displacements, Heading angle, and Velocity simultaneously in order for the GNC block to direct the vehicle.

Scenario 0 concluded that relying only on the raw sensor values and averaging with no intelligence results in incorrect outputs that are not reliable. Improvements to this scenario will be discussed in the next scenarios.

4.5.2 Scenario 1 – Fusion Algorithm Added

This scenario shows the effect of including the proposed fusion algorithm. All sensors contribute individually and no integration of the sensors occurs yet. Adding the fusion algorithm is a considerable step toward more stable and robust output.

By allowing each sensor to vote individually, this scenario also intends to show how this can impact the fused estimates when sensors with the same failure modes behave incorrectly.

Figure 4-33 shows Easting displacement weights.

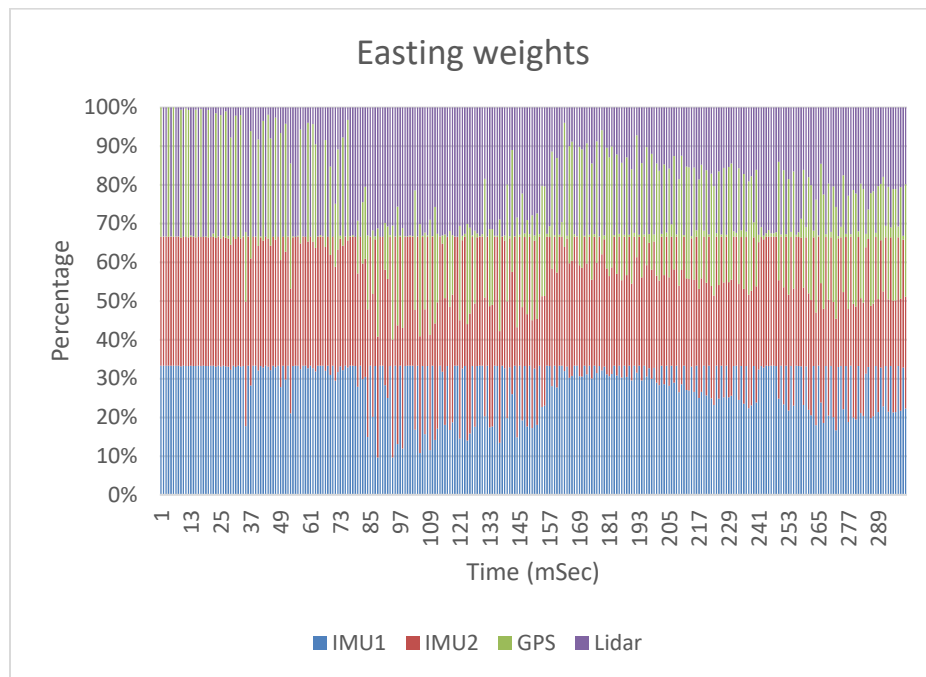


Figure 4-33 Easting weights – Scenario 1

In the Easting weights, the IMUs provide the highest contribution among the sensors. This is due to their individual rights to vote which align with each other. In the next scenario,

where sensors with the same failure modes are grouped, their contribution will be diminished.

Figure 4-34 shows Easting displacement.

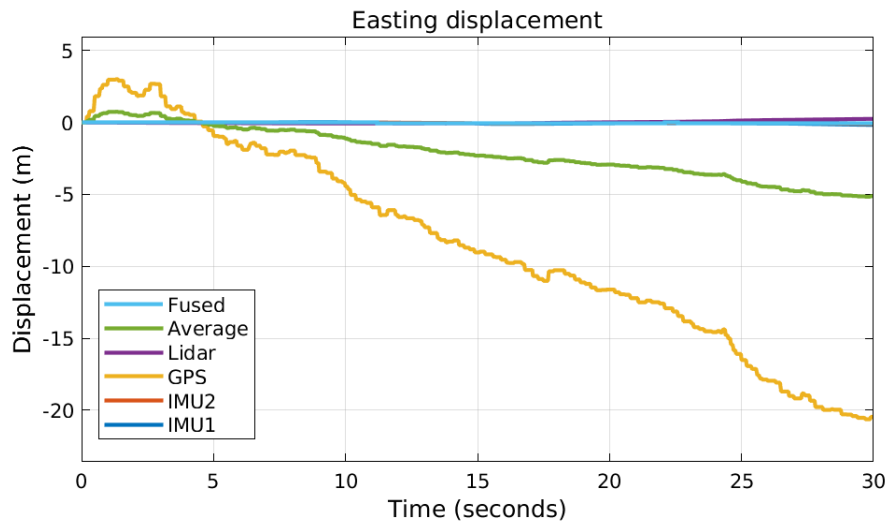


Figure 4-34 Easting displacement – Scenario 1

To better illustrate the above graph, a scaled Easting plot is shown below.

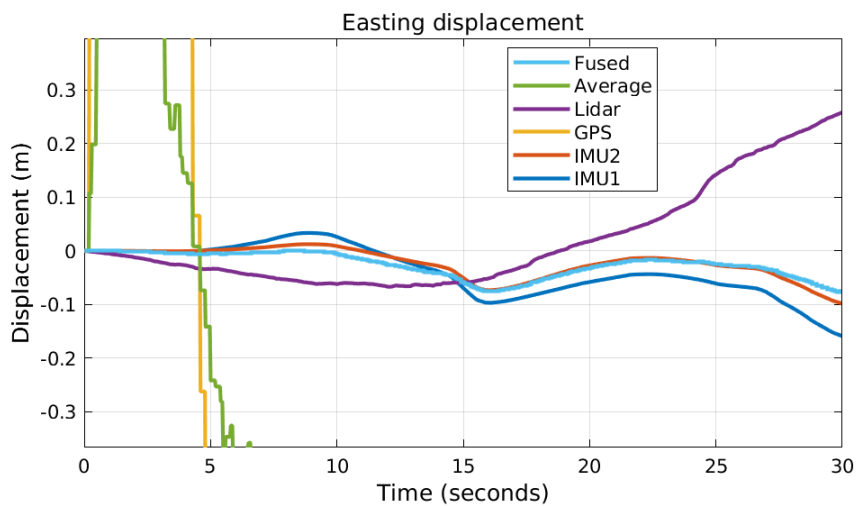


Figure 4-35 Scaled Easting displacement – Scenario 1

In Easting, the fused value of all sensors shows close to zero movement which is desirable.

Figure 4-36 shows Northing displacement weights.

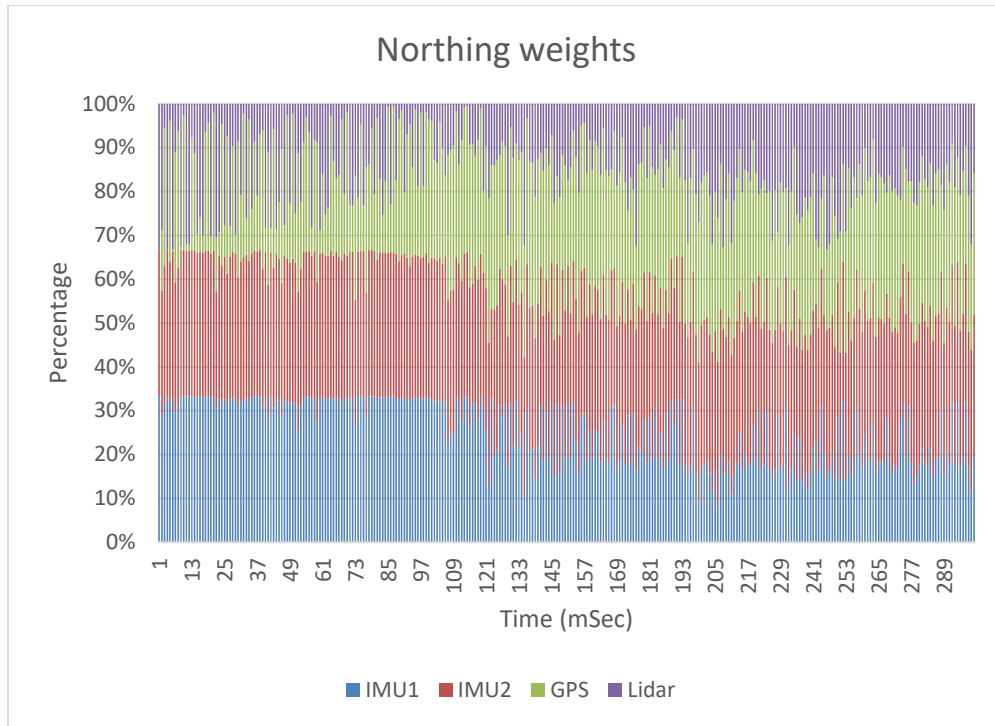


Figure 4-36 Northing weights – Scenario 1

In the Northing weights, the IMUs provide the highest contribution among the sensors. This is due to their individual rights to vote which align with each other. Starting from almost the middle of the run, their contribution decreases since their values start deviating. This fact is shown in the next figure.

Figure 4-37 shows Northing displacement.

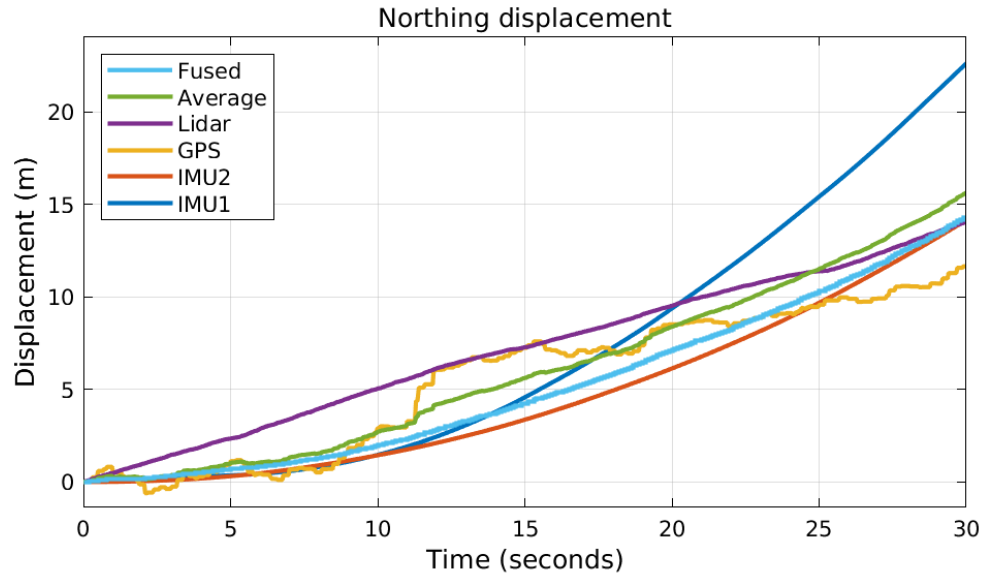


Figure 4-37 Northing displacement – Scenario 1

In Northing, the fused value of all the sensors shows 14.34m movement which is desirable. However, it displays a concave shape in the middle of the run caused by the influence of the IMU1 and IMU2 values.

Figure 4-38 shows Heading weights.

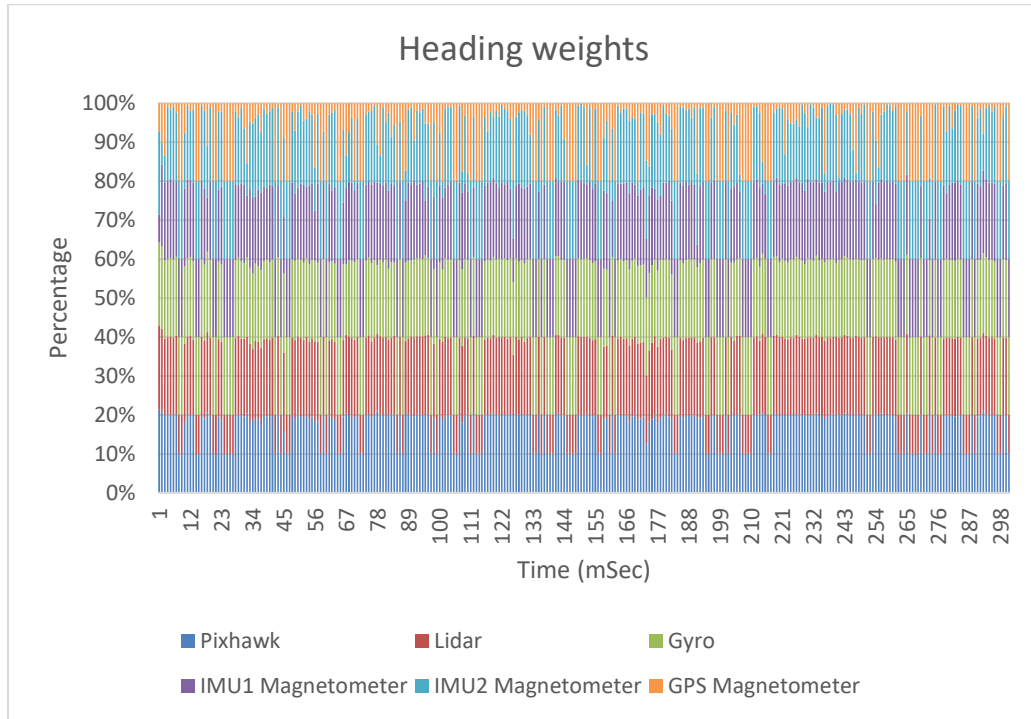


Figure 4-38 Heading weights – Scenario 1

In Heading weights, all sensors are contributing randomly and there is no predominant sensor.

Figure 4-39 shows Heading Angle.

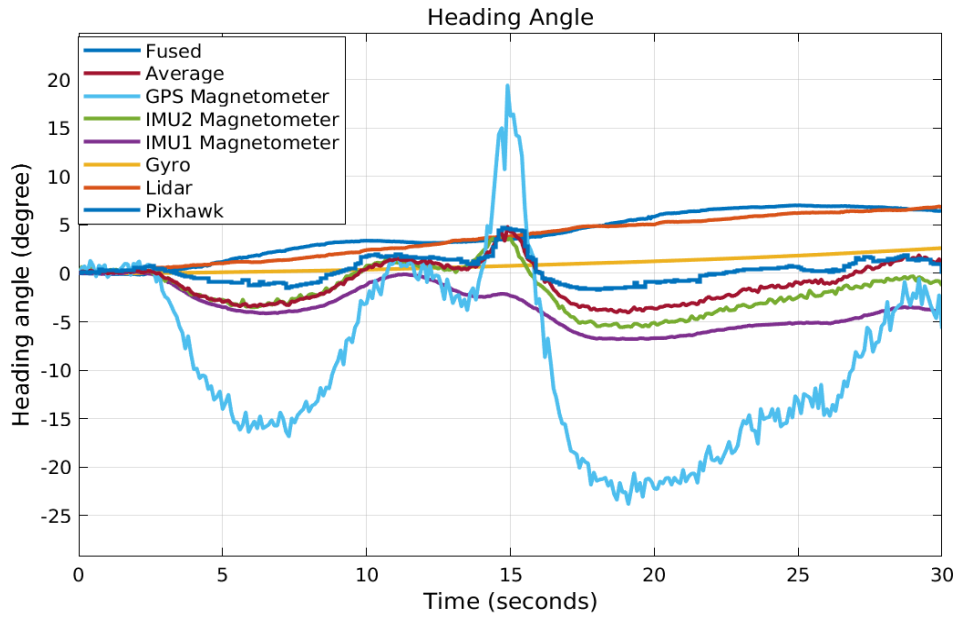


Figure 4-39 Heading angle – Scenario 1

In Heading, the fused value of all sensors shows a range of 7 degrees, performing better than the average.

Figure 4-40 shows Velocity weights.

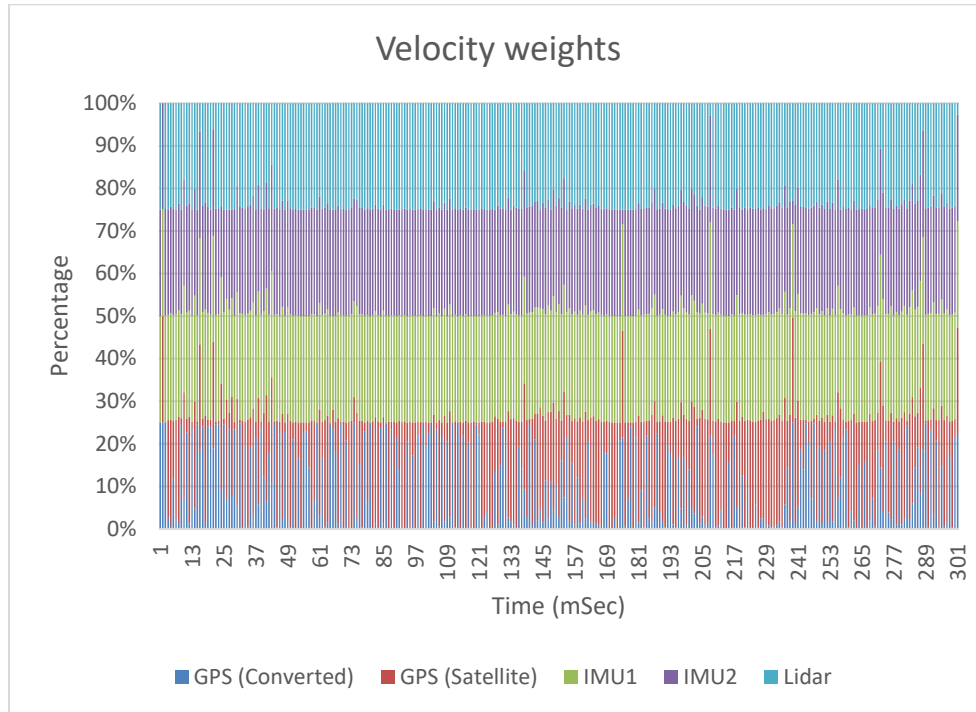


Figure 4-40 Velocity weights – Scenario 1

In Velocity weights, the IMUs and Lidar have almost the same contribution which sum up to 75%. The rest is randomly distributed between the GPS(converted) and GPS(satellite) values.

Figure 4-41 shows Velocity.

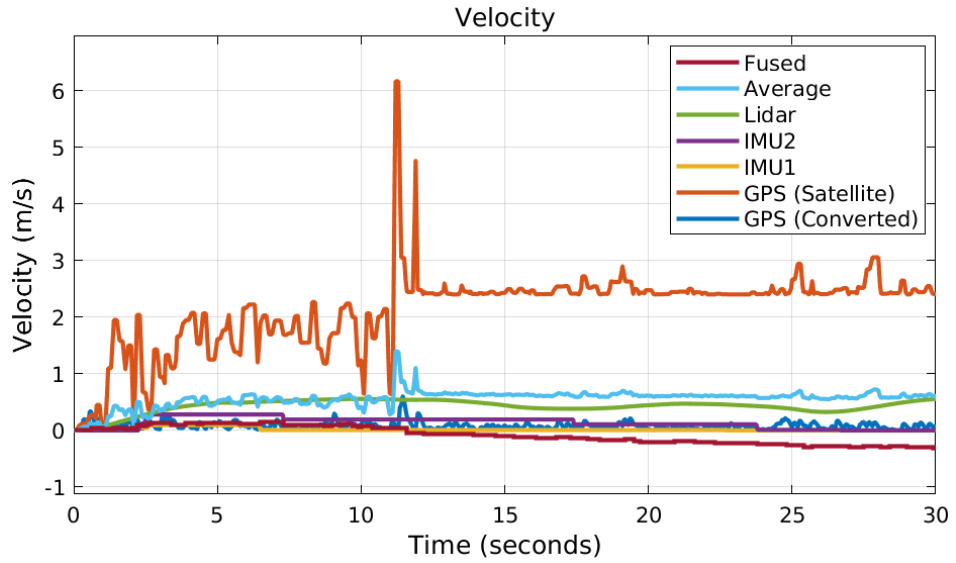


Figure 4-41 Velocity – Scenario 1

To better illustrate the above graph, a scaled Velocity plot is shown below.

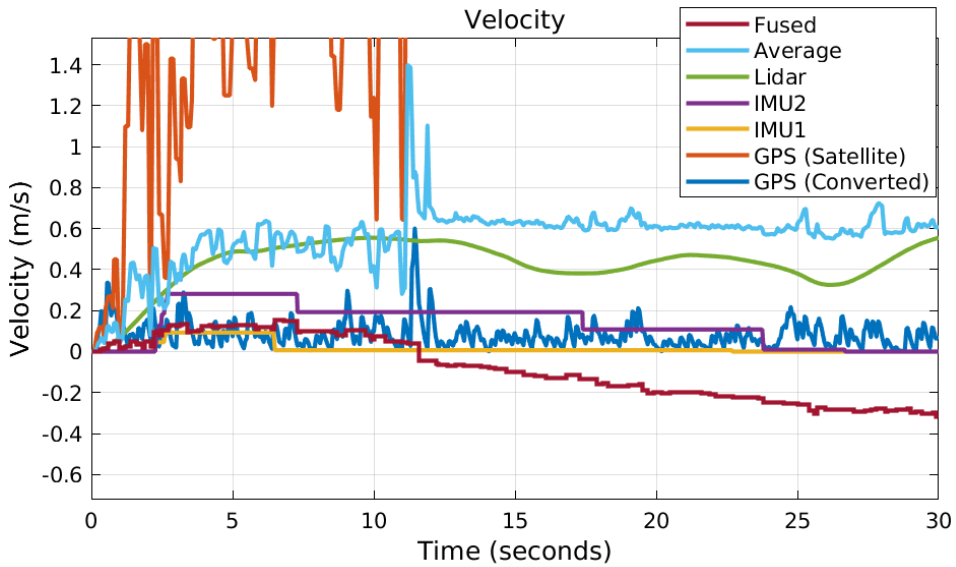


Figure 4-42 Scaled Velocity – Scenario 1

At 11.7 seconds the fused velocity starts generating negative values. The difference (delta) values and the corresponding weights in that moment are shown below.

Table 4-1 Justification of Velocity in Scenario 1

	GPS (Converted)	GPS (Satellite)	IMU1	IMU2	Lidar
Delta Value	-0.289	-0.6056	0	0	-0.0012
Weight	0.2028	0.047	0.2497	0.2497	0.25

Multiplying the delta values by their corresponding weights and adding all of them together concludes -0.0877 as the fused estimate at this time step. This happens due to the fact that the first two sensors (velocity from converted latitude-longitude of the GPS, and direct velocity from GPS satellites) have the right to vote independently. Since none of them obtained an appropriate estimate in this time step (and the time steps after) they are contributing remarkably high and deviate the results. The IMU sensors are also impacting the fused value by having independent voting rights. The next scenario will show how grouping the same sensors (with the same failure modes) diminishes deviating impacts.

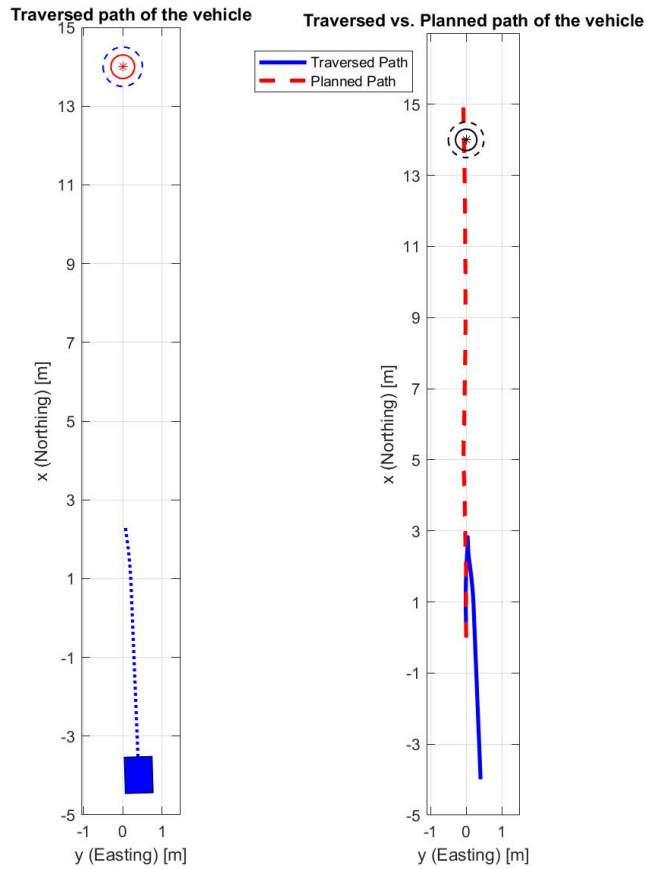


Figure 4-43 Animated traversed path – Scenario 1

As the figures in scenario 1 show, the fused value leads to a better traversing path by the vehicle. The planned path using only the obtained Easting and Northing displacements, and Velocity is shown as a red dotted line in figure 4-43. The traversed path using the modified Easting and Northing displacements, Heading angle, and Velocity is shown as a solid blue line in figure 4-43. The traversed path applies modifications to the values for Easting and Northing displacements, Heading angle, and Velocity simultaneously in order for the GNC block to direct the vehicle.

Scenario 1 concluded that including a smart fusion algorithm can perform remarkably better compared to when raw values are merely averaged. However, it showed that

allowing the sensors with the same failure modes, such as IMU1 and IMU2, to contribute individually can deviate the results which requires attention. Scenario 2 will deal with this issue.

4.5.3 Scenario 2 – Grouping The Same-Class Sensors

The previous scenario allowed for all sensors to be considered individually. This scenario is designed to show how grouping the sensors with the same failure modes, such as IMU1 and IMU2, can impact the fused estimates. Sensors that are in a group have $1/n$ right to vote where n is the number of sensors in that particular group. This grouping strategy is used for the rest of the scenarios starting from this scenario.

Easting combines IMU1 and IMU2 as one group. They are simply averaged and one value (the average of IMUs) participates in the fusion process.

Figure 4-44 shows Easting weights.

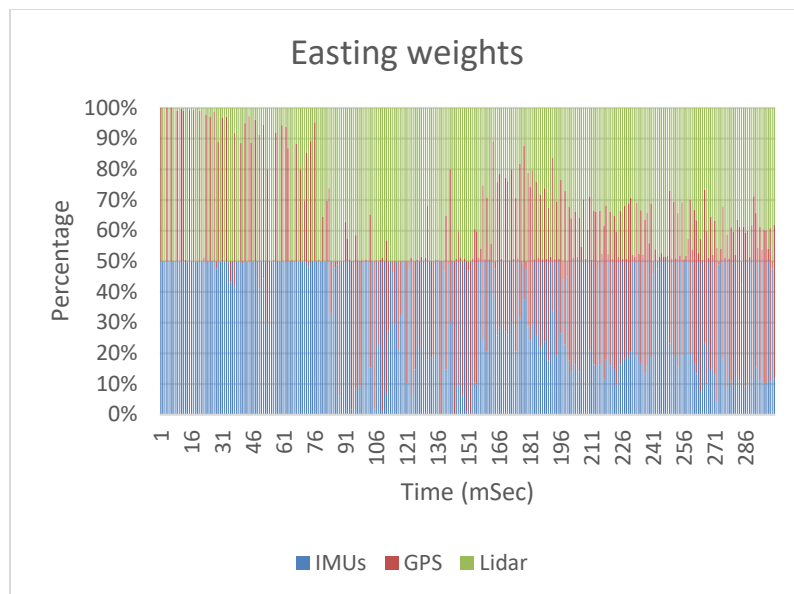


Figure 4-44 Easting weights – Scenario 2

In Easting weights, most of the weights are distributed to the IMUs and Lidar. The GPS does not contribute remarkably since the values are not aligned with the other sensors.

Figure 4-45 shows Easting displacement.

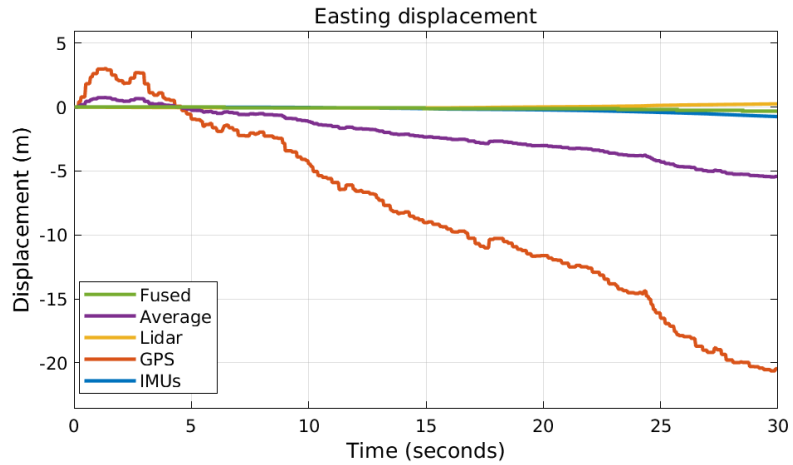


Figure 4-45 Easting displacement – Scenario 2

To better illustrate the above graph, a scaled Easting plot is shown below.

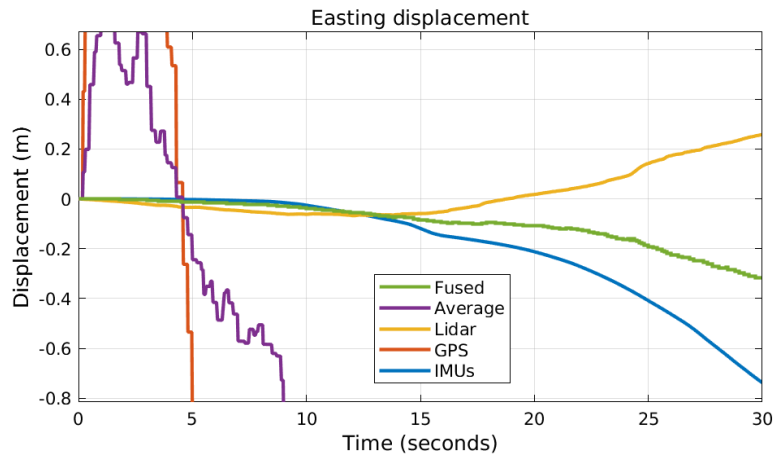


Figure 4-46 Scaled Easting displacement – Scenario 2

In Easting, the fused value of all sensors shows close to zero movement which is desirable.

Northing combines IMU1 and IMU2 as one group. They are simply averaged and one value (the average of the IMUs) participates in the fusion process.

Figure 4-47 shows Northing weights.

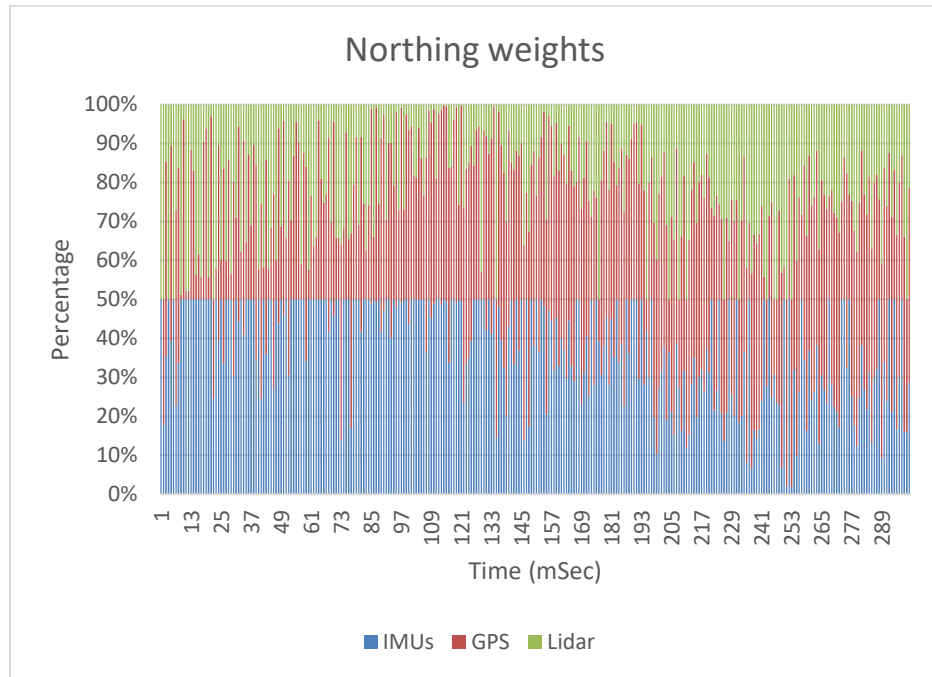


Figure 4-47 Northing weights – Scenario 2

In the Northing weights, 50% of the weights are allocated to the IMUs mostly by the middle of the run. The GPS and Lidar contribute randomly in different time steps.

Figure 4-48 shows Northing displacement.

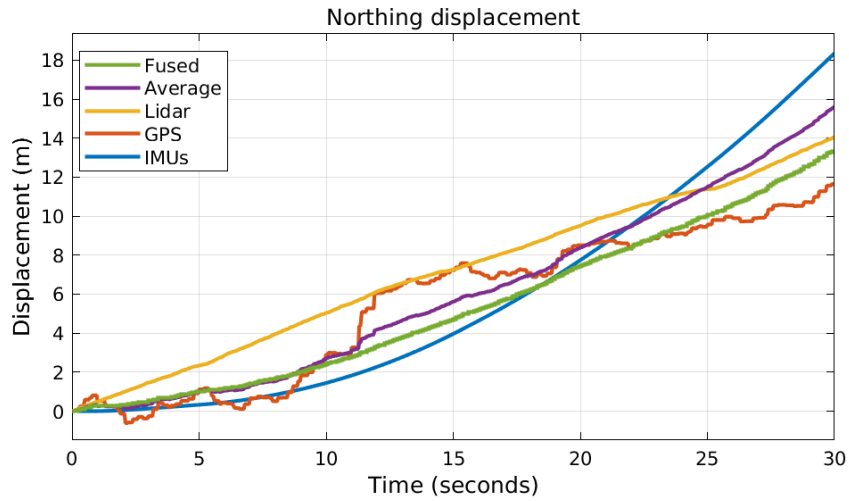


Figure 4-48 Northing displacement – Scenario 2

In Northing, considering the same-class sensors (IMU1 and IMU2) in one group decreases the fused Northing estimate to 13.41m, which is slightly lower than the previous scenario. The concavity of the fused values is also diminished remarkably.

Figure 4-49 shows Heading weights. The Heading weights in this scenario combine the IMU1 magnetometer, IMU2 magnetometer, and GPS magnetometer into one group. They are simply averaged and one value (the average of the Magnetometers) participates in the fusion process.

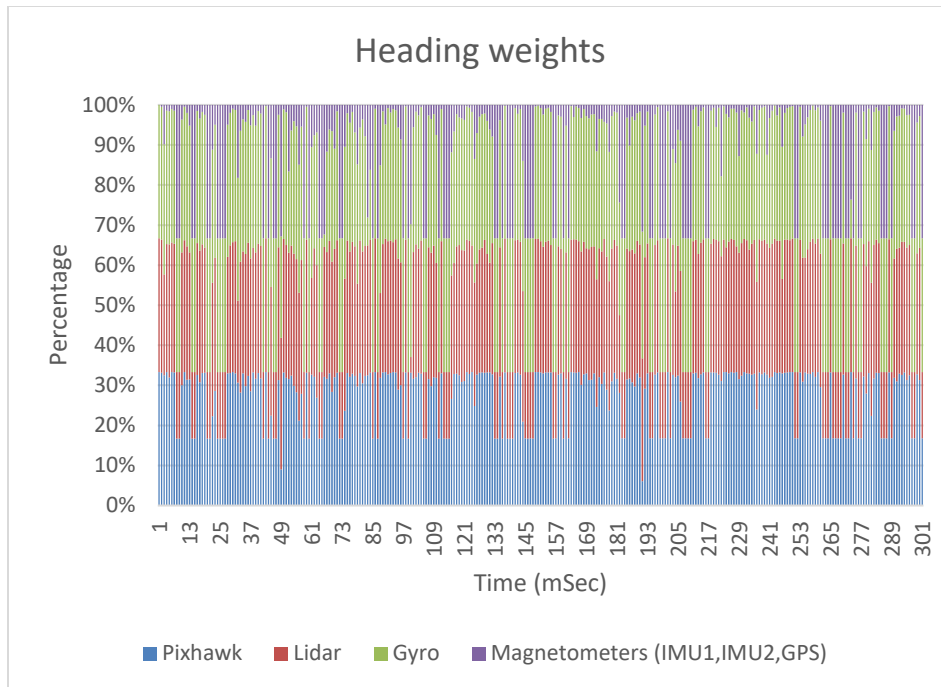


Figure 4-49 Heading weights – Scenario 2

In the Heading weights, Pixhawk contributes almost 30% most of the time. The rest is distributed randomly among the other sensors.

Figure 4-50 shows Heading Angle.

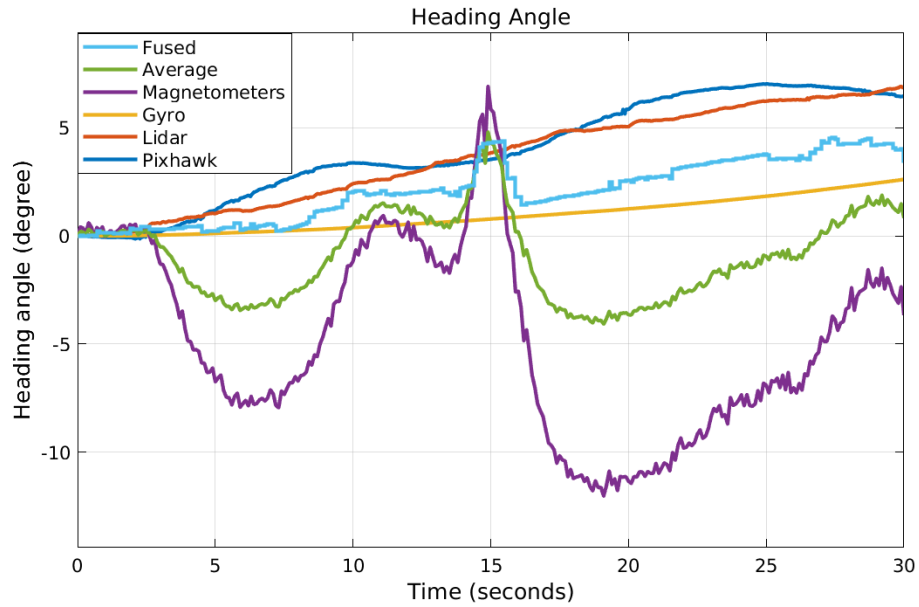


Figure 4-50 Heading angle – Scenario 2

By comparing the heading in this scenario with the fused estimate in the previous scenario it can clearly be seen that grouping sensors with similar failure modes decreases the fluctuations and causes the estimate to be less sensitive to the shifts with a total range of 4 degrees.

Figure 4-51 shows Velocity weights. The Velocity weights in this scenario combine the IMU1 and IMU2 sensor readings into one group, and GPS (converted velocity from position) and GPS (direct velocity from satellites) sensor values from another group. They are simply averaged and one value (the average of the IMUs and the average of the GPSs) participates in the fusion process.

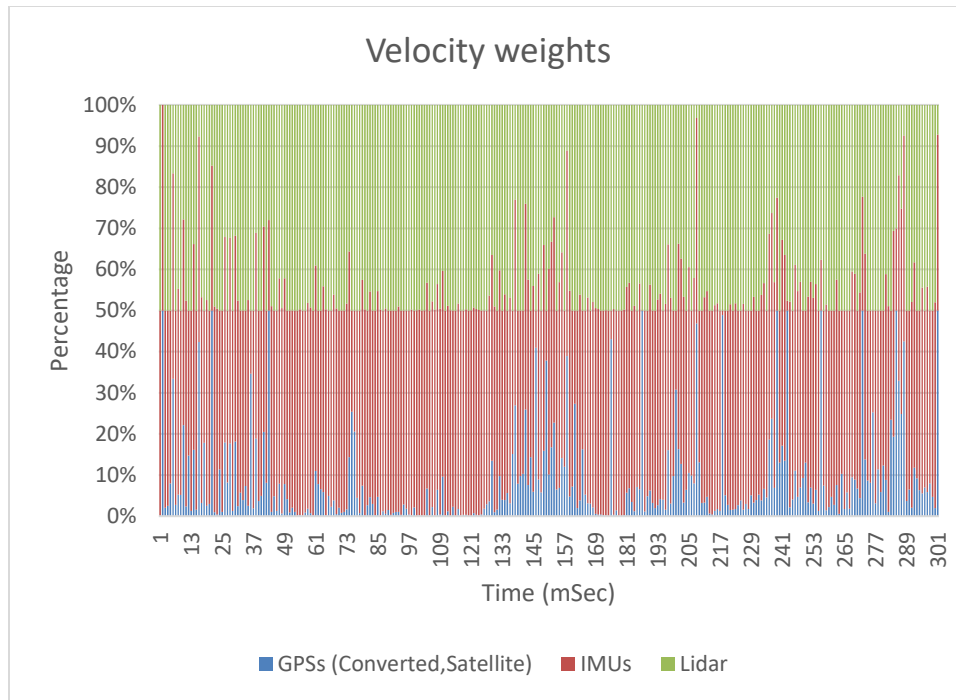


Figure 4-51 Velocity weights – Scenario 2

In the Velocity weights, nearly all of the weights are dedicated to the IMUs and Lidar. The GPS values are merely given close to 5% in the entire run. Contrary to the previous scenario, the Velocity estimates are mostly influenced by the IMUs and Lidar values in this scenario.

Figure 4-52 shows Velocity.

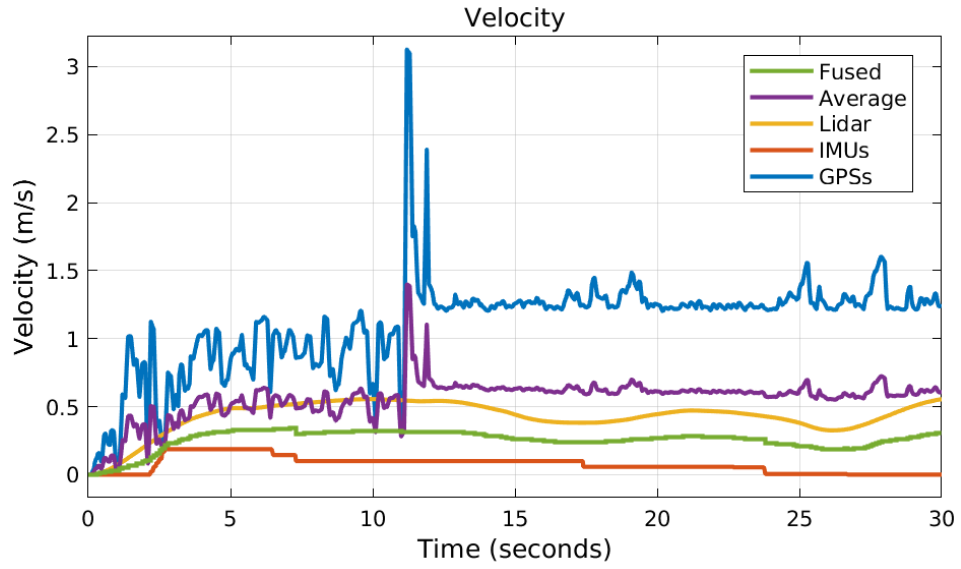


Figure 4-52 Velocity – Scenario 2

In Velocity, grouping sensors with the same failure modes led to a rational fused estimate compared to the previous scenario.

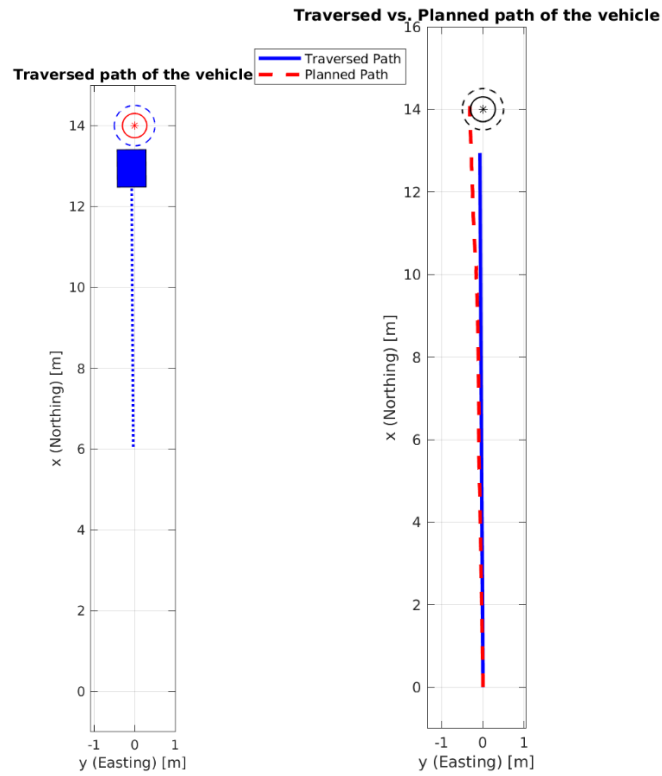


Figure 4-53 Animated traversed path – Scenario 2

As the figures in scenario 2 show, the fused value leads to a better traversing path by the vehicle. The planned path using only the obtained Easting and Northing displacements, and Velocity is shown as a red dotted line in figure 4-53. The traversed path using the modified Easting and Northing displacements, Heading angle, and Velocity is shown as a solid blue line in figure 4-53. The traversed path applies modifications to the values for Easting and Northing displacements, Heading angle, and Velocity simultaneously in order for the GNC block to direct the vehicle.

Scenario 2 concluded that grouping the sensors with the same failure modes is a compelling rule to apply to the sensor weights in order to generate less distorted outputs. This rule is retained for scenarios 3 and 4.

4.5.4 Scenario 3 – System Model Added

This scenario adds another input to the set of sensors, called the System Model, which is calculated to be the closest estimation of what was commanded to the vehicle. Considering a known and expected path for the vehicle to traverse, this System Model acts as an additional virtual sensor to the set of sensors with a separate and individual right to vote in the fusion process. This scenario performs as a transition mode from Scenario 2 to Scenario 4 in which a System Model is introduced as a guide but its weight will later be removed.

Figure 4-54 shows Easting weights.

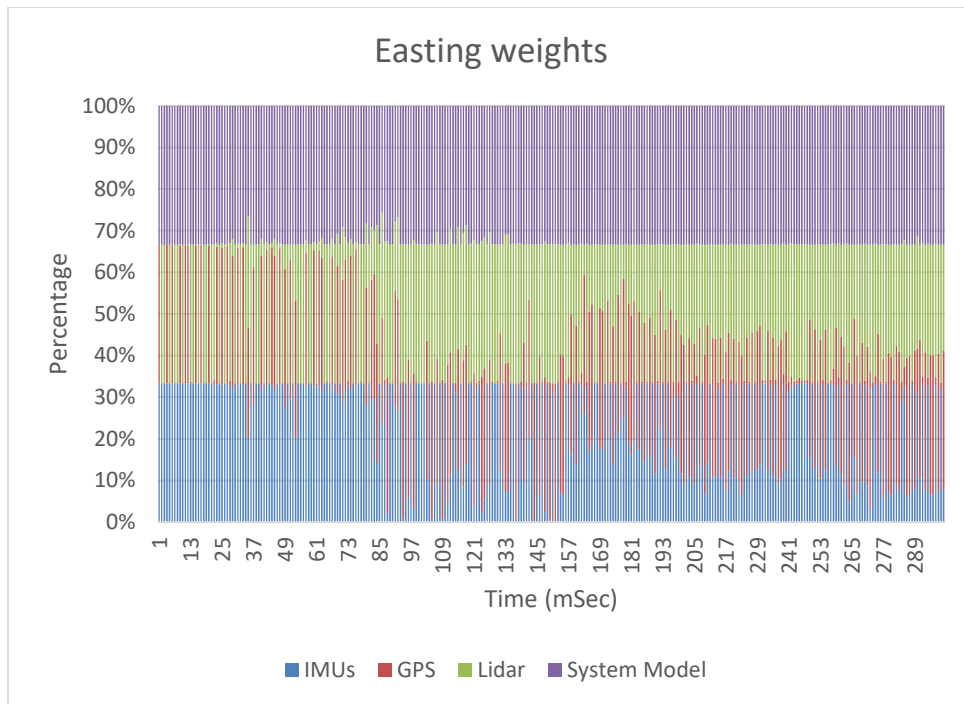


Figure 4-54 Easting weights – Scenario 3

In the Easting weights, the weights are mostly allocated to the IMUs, Lidar, and System Model. A small proportion is also devoted to GPS at times. Due to the fact that the

System Model, IMUs, and Lidar have closer values to each other, this led the three sensors to contribute almost equally. The GPS does not have close values to the other sensors and that is why the GPS is given trivial weights. The next figure shows Easting displacement.

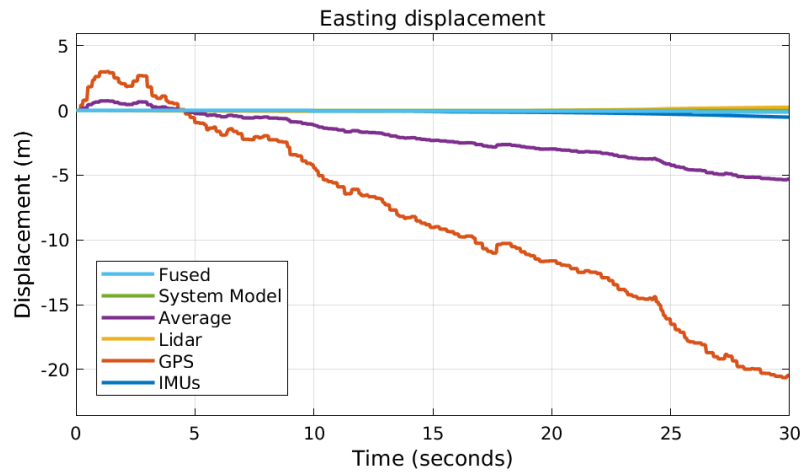


Figure 4-55 Easting displacement – Scenario 3

To better illustrate the above graph, a scaled Easting plot is shown below.

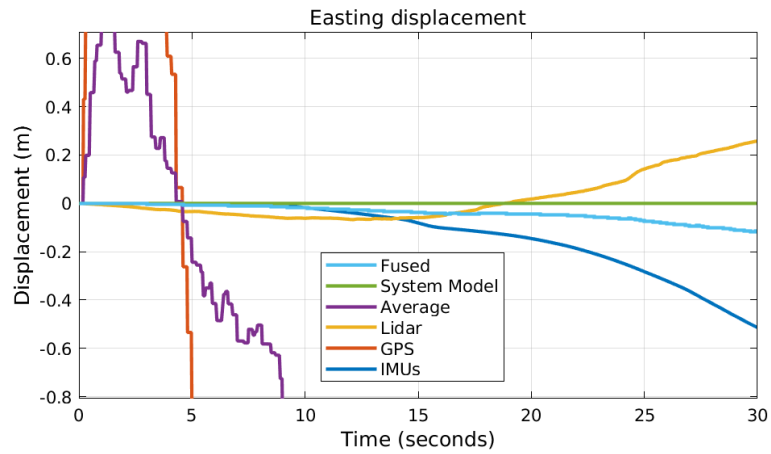


Figure 4-56 Scaled Easting displacement – Scenario 3

In Easting, the fused value of all sensors shows close to zero movement which is desirable.

Figure 4-57 shows Northing weights.

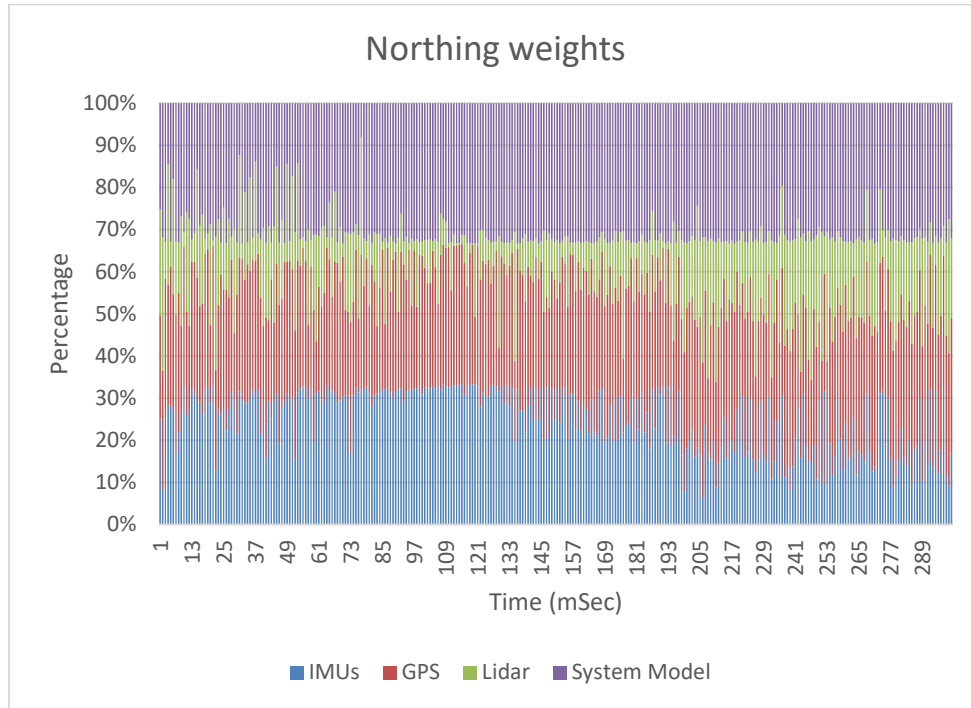


Figure 4-57 Northing weights – Scenario 3

In the Northing weights, although Lidar is a reliable sensor it is not given significant weights in the entire run. This implies that delta values from Lidar did not agree remarkably to that of the other sensors.

Figure 4-58 shows Northing displacement.

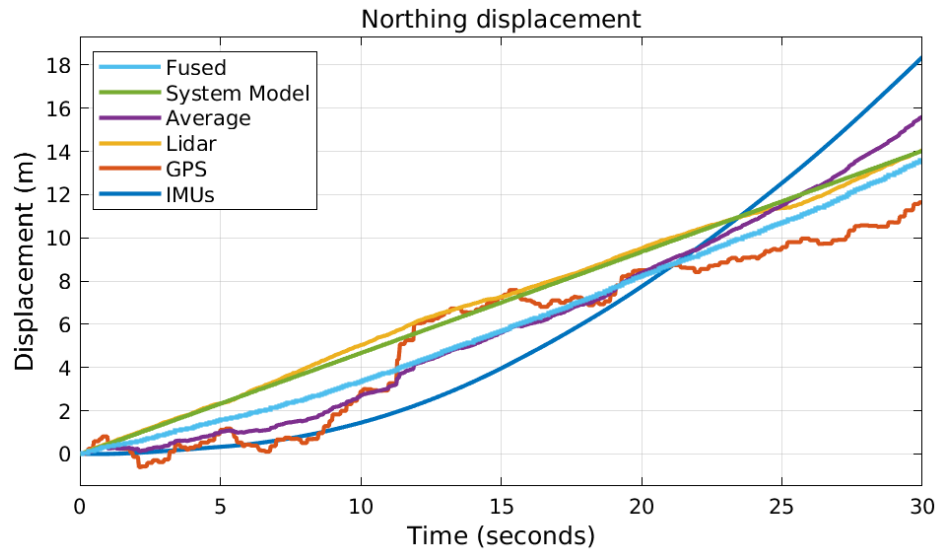


Figure 4-58 Northing displacement – Scenario 3

In the Northing displacement, the fused value of all the sensors shows a 13.64m movement which is desirable.

Figure 4-59 shows Heading weights.

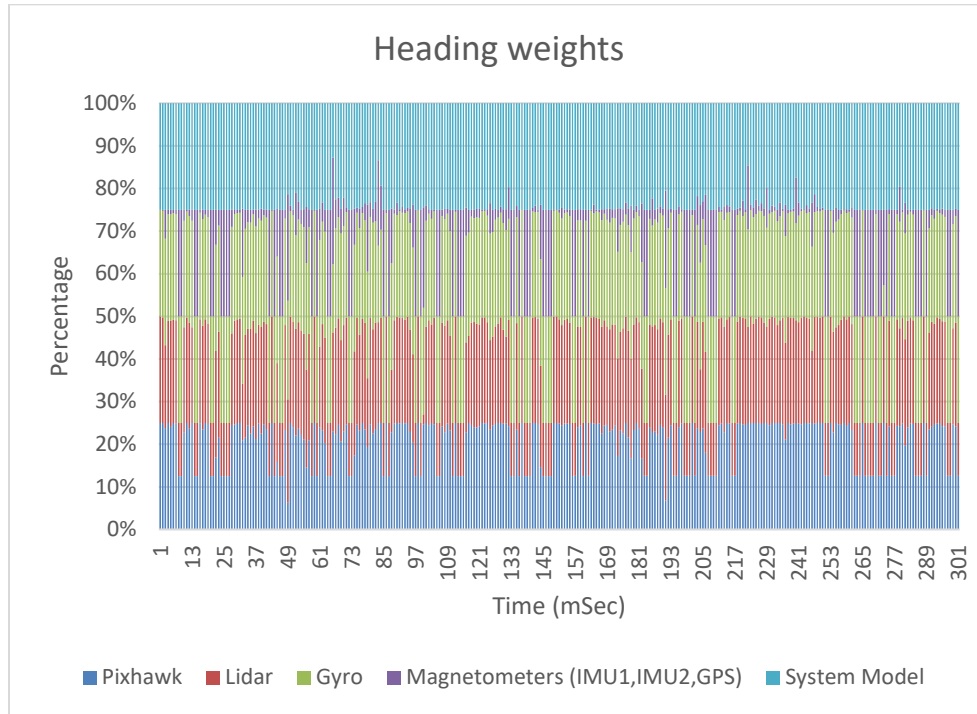


Figure 4-59 Heading weights – Scenario 3

In the Heading weights, the System Model and Pixhawk are allocated consistent percentage weights over the entire run. Other sensors contribute distinctly in different time steps.

Figure 4-60 shows Heading Angle.

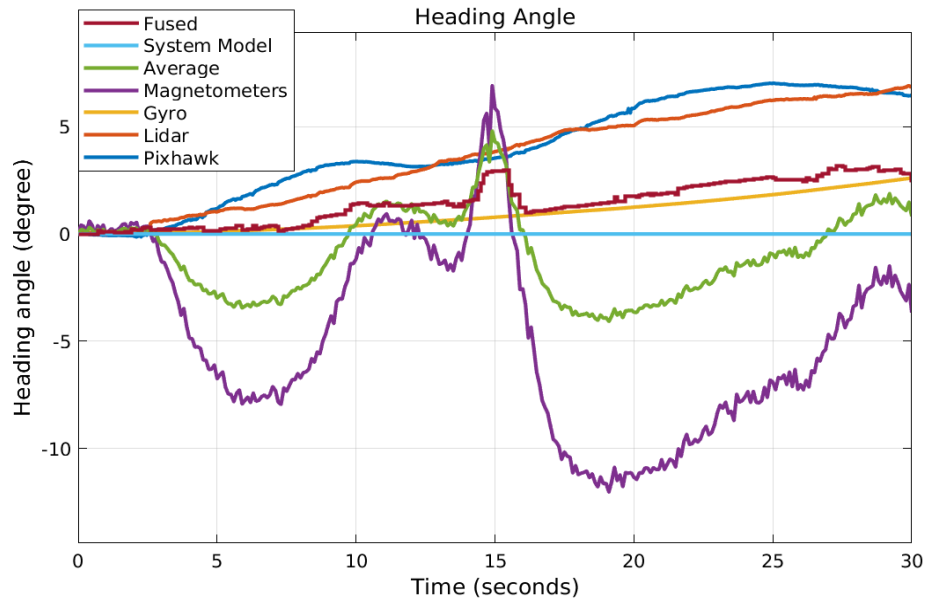


Figure 4-60 Heading angle – Scenario 3

In the Heading Angle, the fused value of all the sensors shows a total range of 3 degrees in the heading values. This demonstrates an increased performance compared to the previous scenarios.

Figure 4-61 shows Velocity weights.

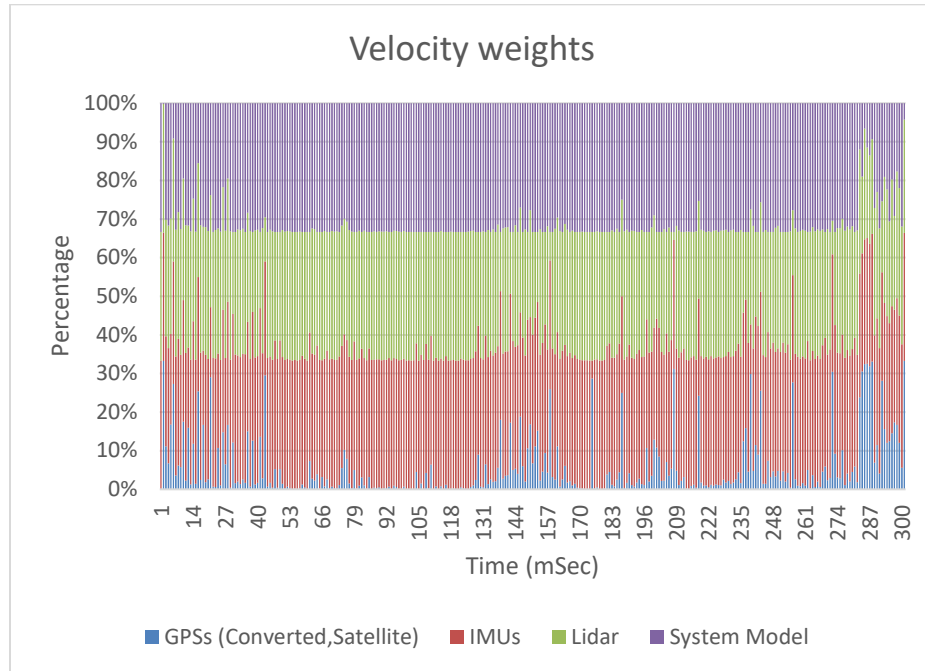


Figure 4-61 Velocity weights – Scenario 3

In the Velocity weights, the IMUs, Lidar, and System Model are all given almost the same weights of 30% over the entire run. However, the GPSs have trivial weights.

Next figure shows Velocity.

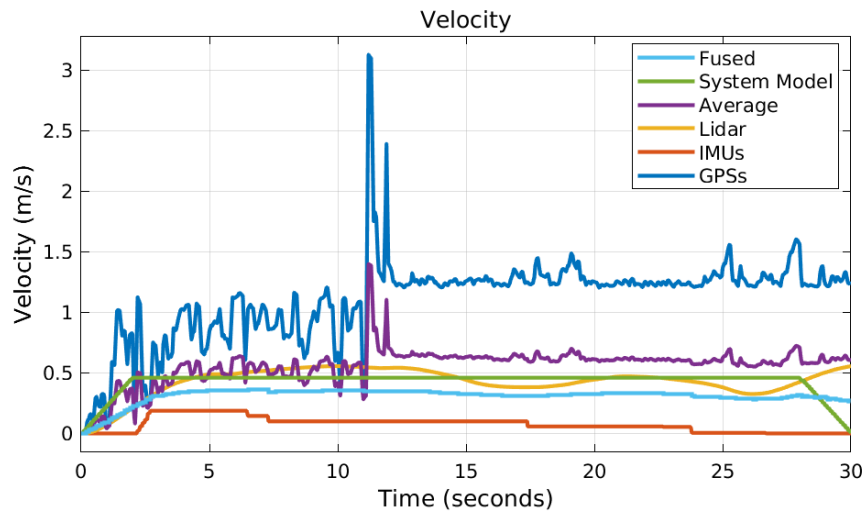


Figure 4-62 Velocity – Scenario 3

In Velocity, adding System Model to Velocity decreased the variation slightly compared to the previous scenario.

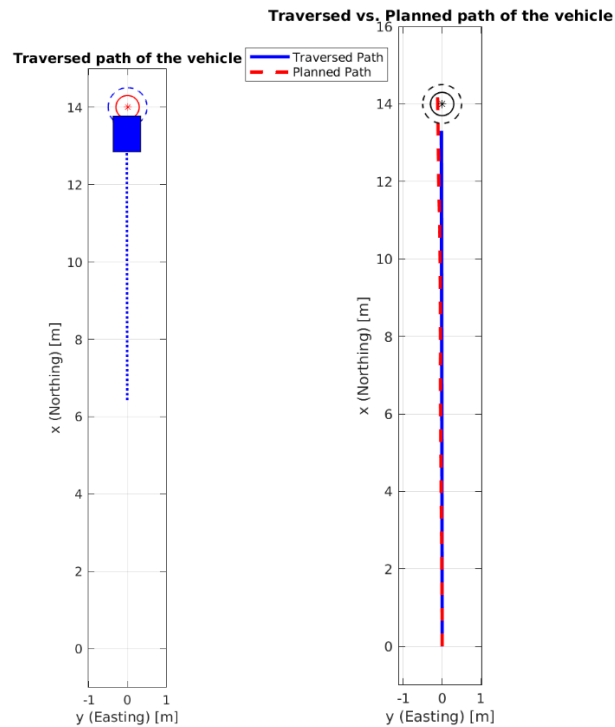


Figure 4-63 Animated traversed path – Scenario 3

As the figures in scenario 3 show, the fused value leads to a better traversing path by the vehicle. The planned path using only the obtained Easting and Northing displacements, and Velocity is shown as a red dotted line in figure 4-63. The traversed path using the modified Easting and Northing displacements, Heading angle, and Velocity is shown as a solid blue line in figure 4-63. The traversed path applies modifications to the values for Easting and Northing displacements, Heading angle, and Velocity simultaneously in order for the GNC block to direct the vehicle.

Scenario 3 concluded that adding the System Model as an additional sensor helped decrease the fluctuations. It also allowed for closer estimates to expected values. This scenario is a transition state for the next scenario where the System Model will be present for voting purposes but holds zero weight.

4.5.5 Scenario 4 – System Model's Weight Removed

This scenario removes the weight from System Model introduced in the previous scenario, although it exists as guidance for other sensors. The intention is to use piece of data but to assign zero weight to it so that we do not deviate the output toward the System Model.

Figure 4-64 shows Easting weights.

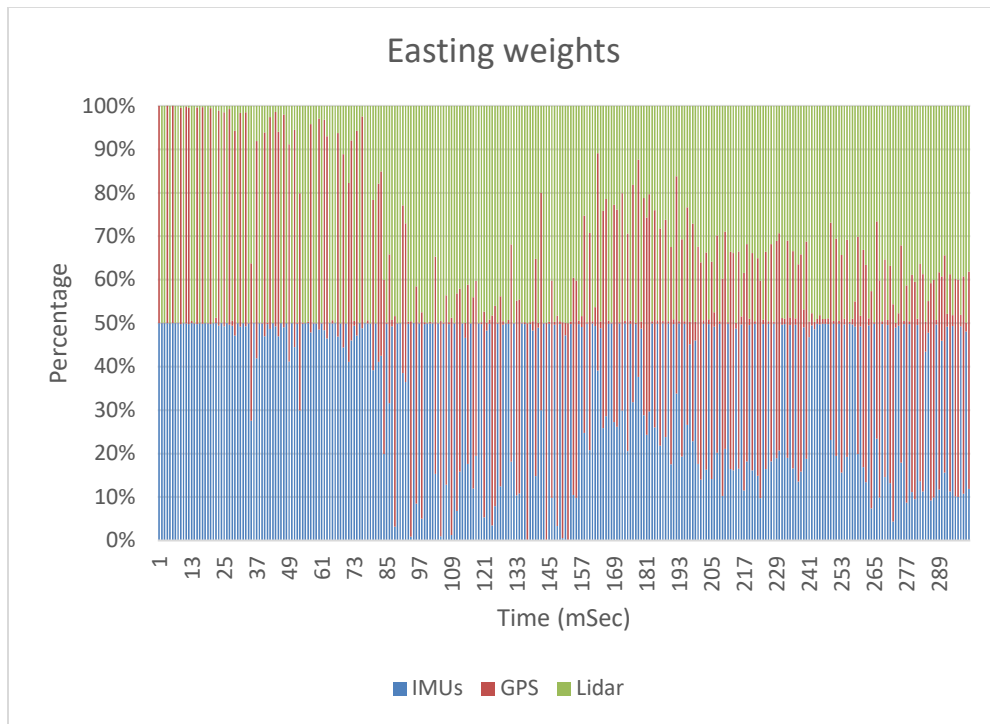


Figure 4-64 Easting weights – Scenario 4

In the Easting weights, Lidar and IMUs are allocated most of the weights. By removing the weight from the System Model in this scenario Easting weight shows close results to Scenario 2.

Figure 4-65 shows Easting displacement.

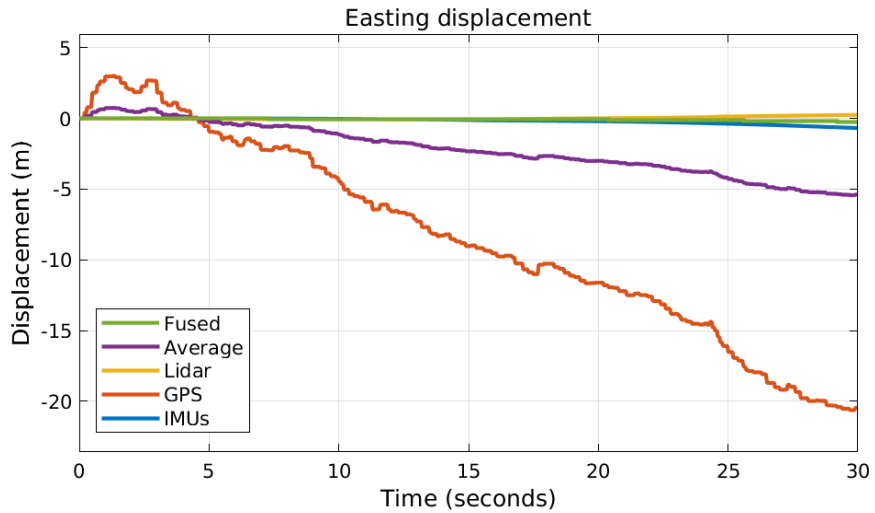


Figure 4-65 Easting displacement – Scenario 4

To better illustrate the above graph, a scaled Easting plot is shown below.

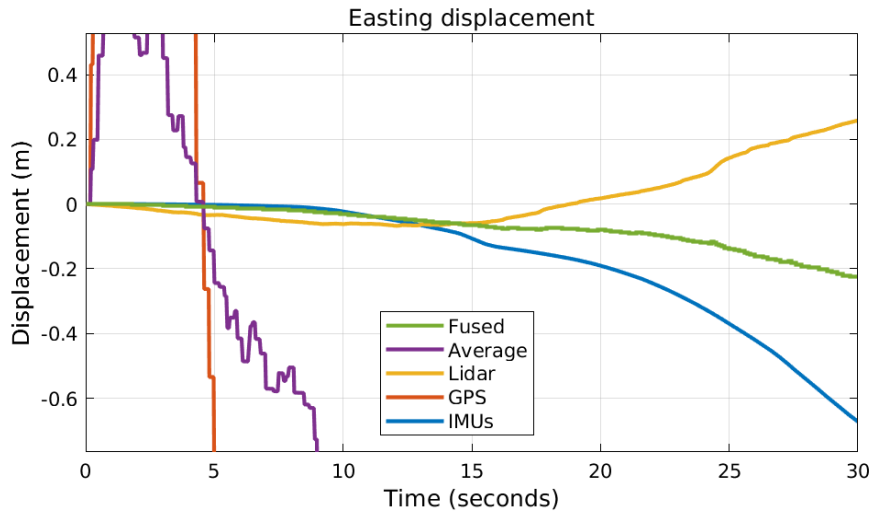


Figure 4-66 Scaled Easting displacement – Scenario 4

In Easting displacement, removing the weight from the System Model did not cause the fused estimate a considerable change compared to scenarios 2 and 3.

Figure 4-67 shows Northing weights.

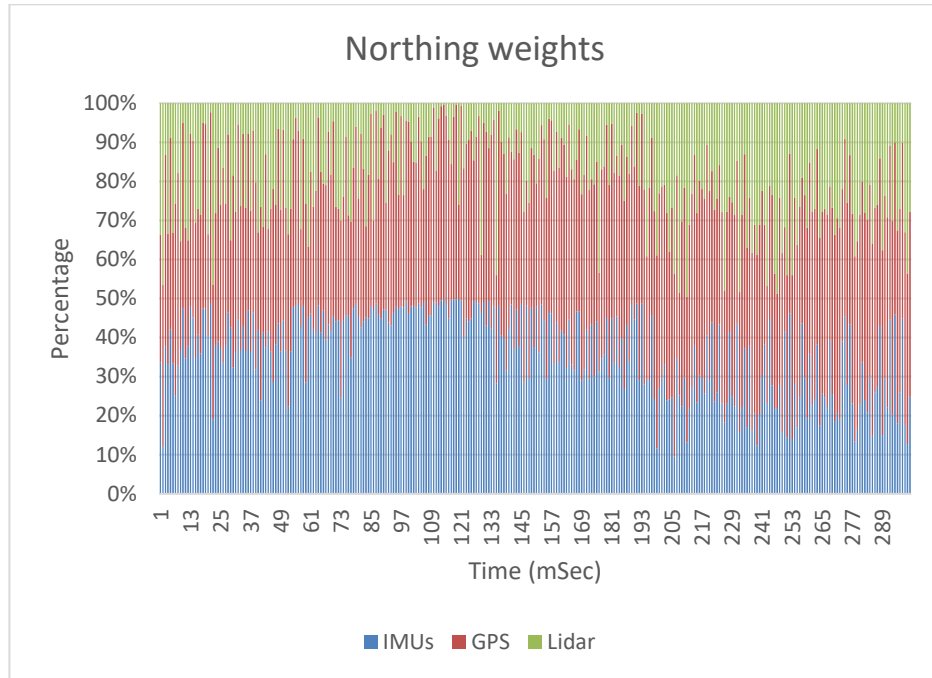


Figure 4-67 Northing weights – Scenario 4

In the Northing weights, the GPS and IMUs are allocated considerable weights. Although Lidar is a reliable sensor in this study, due to not having close delta values to other sensors it is dedicated less weights than the other sensors.

Figure 4-68 shows Northing displacement.

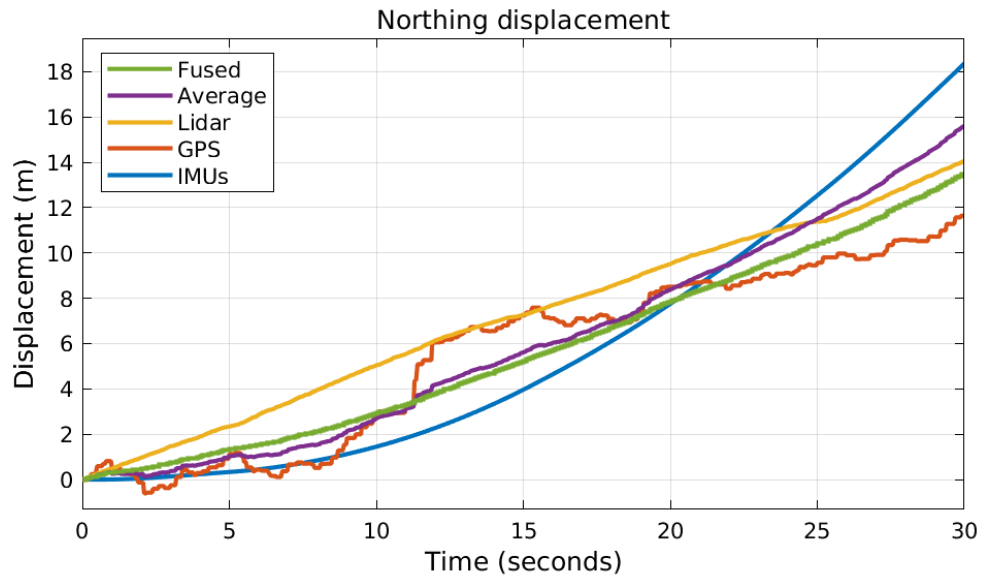


Figure 4-68 Northing displacement – Scenario 4

In Northing displacement, the fused value of all the sensors shows a 13.54m movement which is desirable and slightly lower than the previous scenario.

Figure 4-69 shows Heading weights.

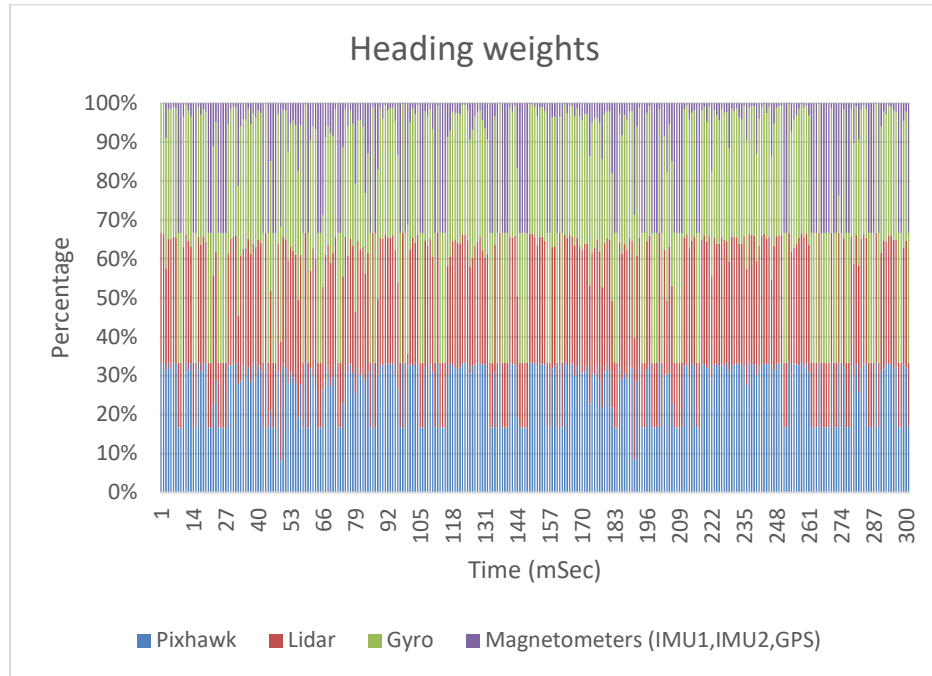


Figure 4-69 Heading weights – Scenario 4

In Heading weights, the Pixhawk contributes over 30% and other sensors contribute distinctly in different time steps.

Figure 4-70 shows Heading Angle.

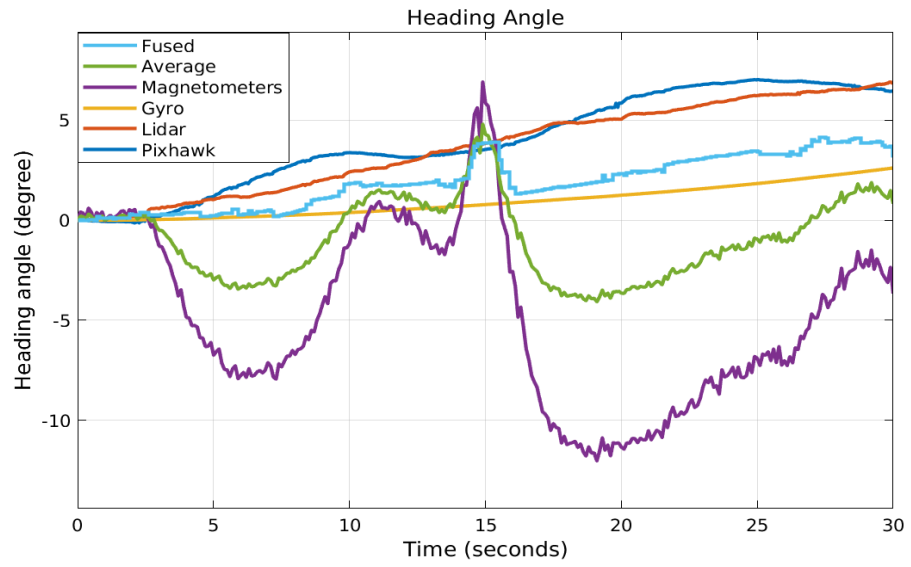


Figure 4-70 Heading angle – Scenario 4

In Heading Angle, the fused value of all sensors shows a total range of 4 degrees.

Figure 4-71 shows Velocity weights.

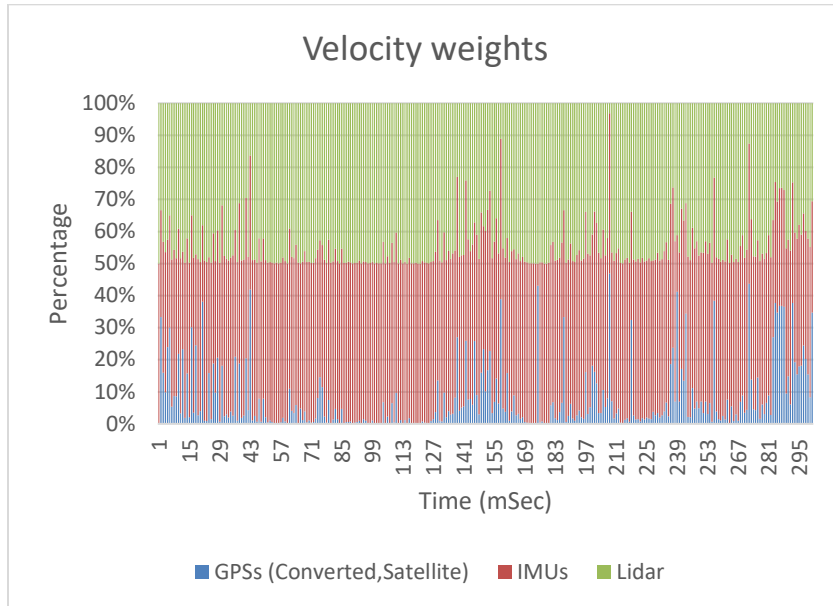


Figure 4-71 Velocity weights – Scenario 4

In Velocity weights, the IMUs and Lidar are allocated most of the weight. Grouping GPS sensors caused them to receive a small portion of weight. The next figure shows Velocity.

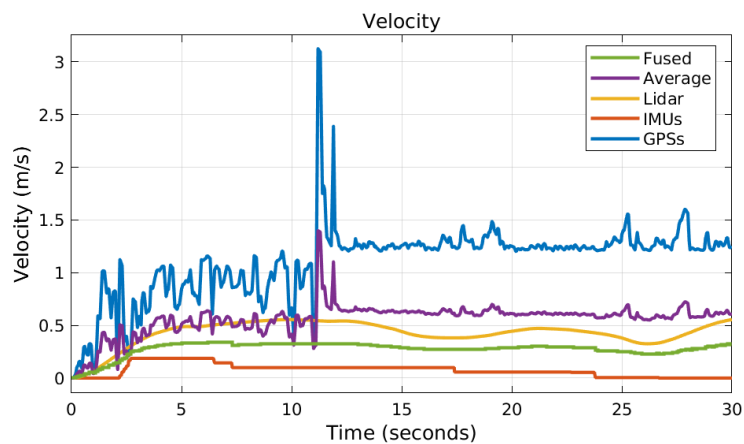


Figure 4-72 Velocity – Scenario 4

In Velocity, results show close outputs to scenarios 2 and 3.

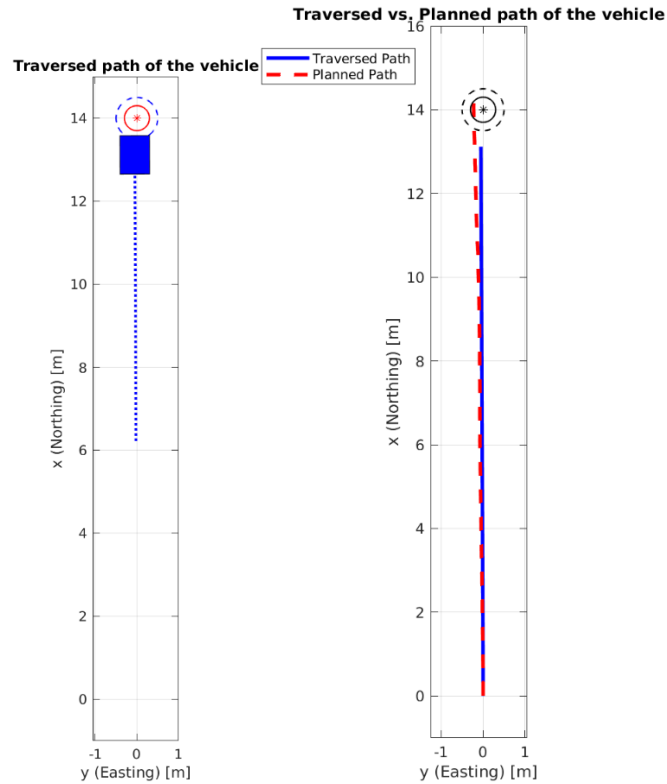


Figure 4-73 Animated traversed path – Scenario 4

As the figures in scenario 4 show, the fused value leads to a desirable traversing path by the vehicle. The planned path using only the obtained Easting and Northing displacements, and Velocity is shown as a red dotted line in figure 4-73. The traversed path using the modified Easting and Northing displacements, Heading angle, and Velocity is shown as a solid blue line in figure 4-73. The traversed path applies modifications to the values for Easting and Northing displacements, Heading angle, and Velocity simultaneously in order for the GNC block to direct the vehicle.

Scenario 4 generates similar results to Scenario 3 where a System Model is added but its weight is removed. Scenario 3 was introduced as a transition stage to this scenario, meaning that Scenario 4 is more valid since the System Model is given zero weight.

4.6 Performance Overview – Average and Fusion Scenarios

To better understand the differences/improvements in each scenario, all of them are included in separate sheets for each dimension. Weights are also included for a greater realization on how the scenarios function. Scenario 0 (average) does not include figures for weights. The most important tables are shown in what follows and the remaining tables are presented in the Appendix.

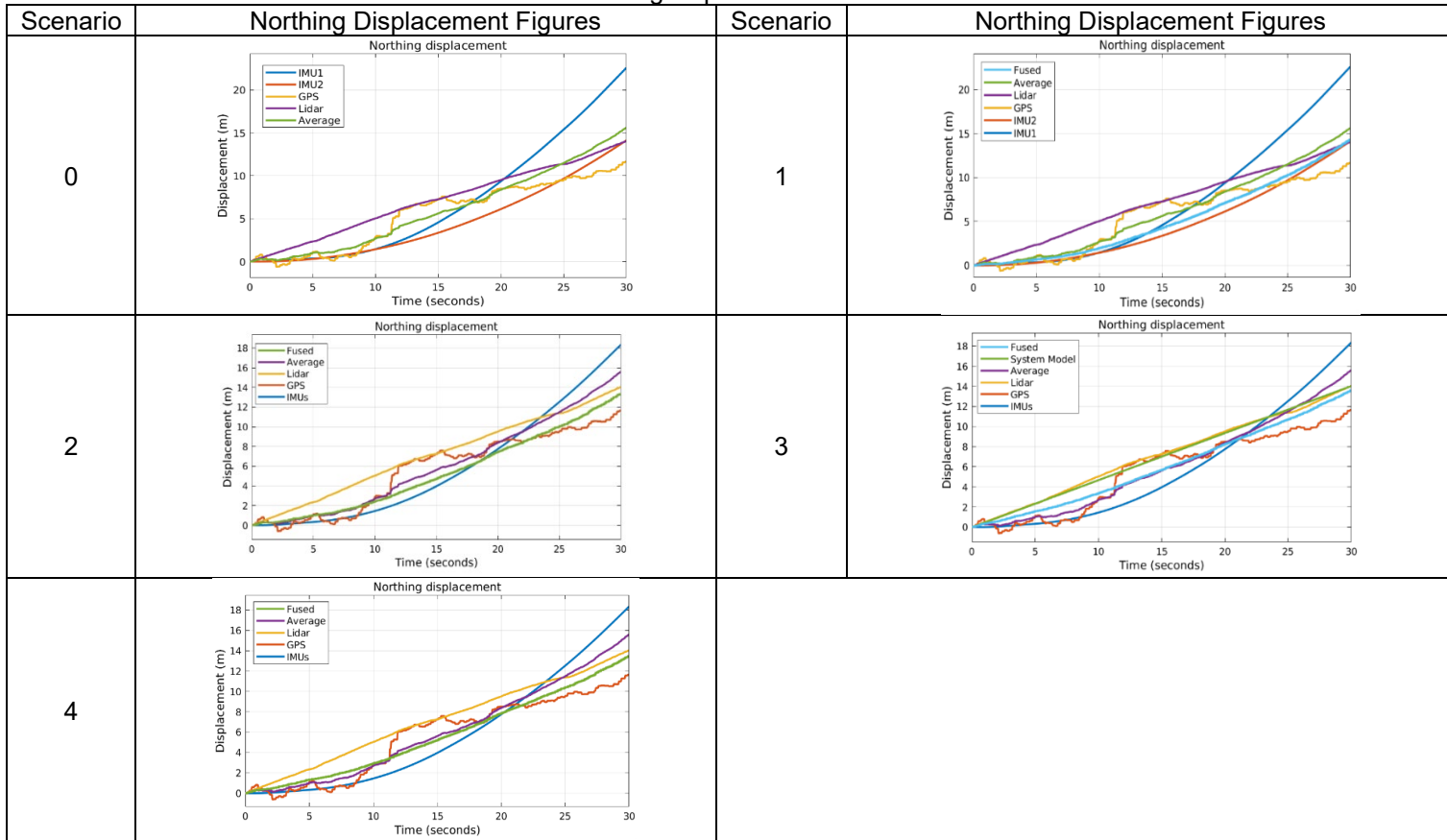
According to the results from the four fusion scenarios, including “System Model” may improve the performance. In the following tables where all figures are presented simultaneously, this fact is easier to see. The corresponding weight tables in Appendix also confirm this. A primary conclusion is that grouping the sensors with the same failure modes, and adding a System Model signal while using the proposed fusion algorithm shows reliable results.

Table 4-2 Easting displacement for all scenarios

Scenario	Easting Displacement Figures	Scenario	Easting Displacement Figures
0		1	
2		3	
4			

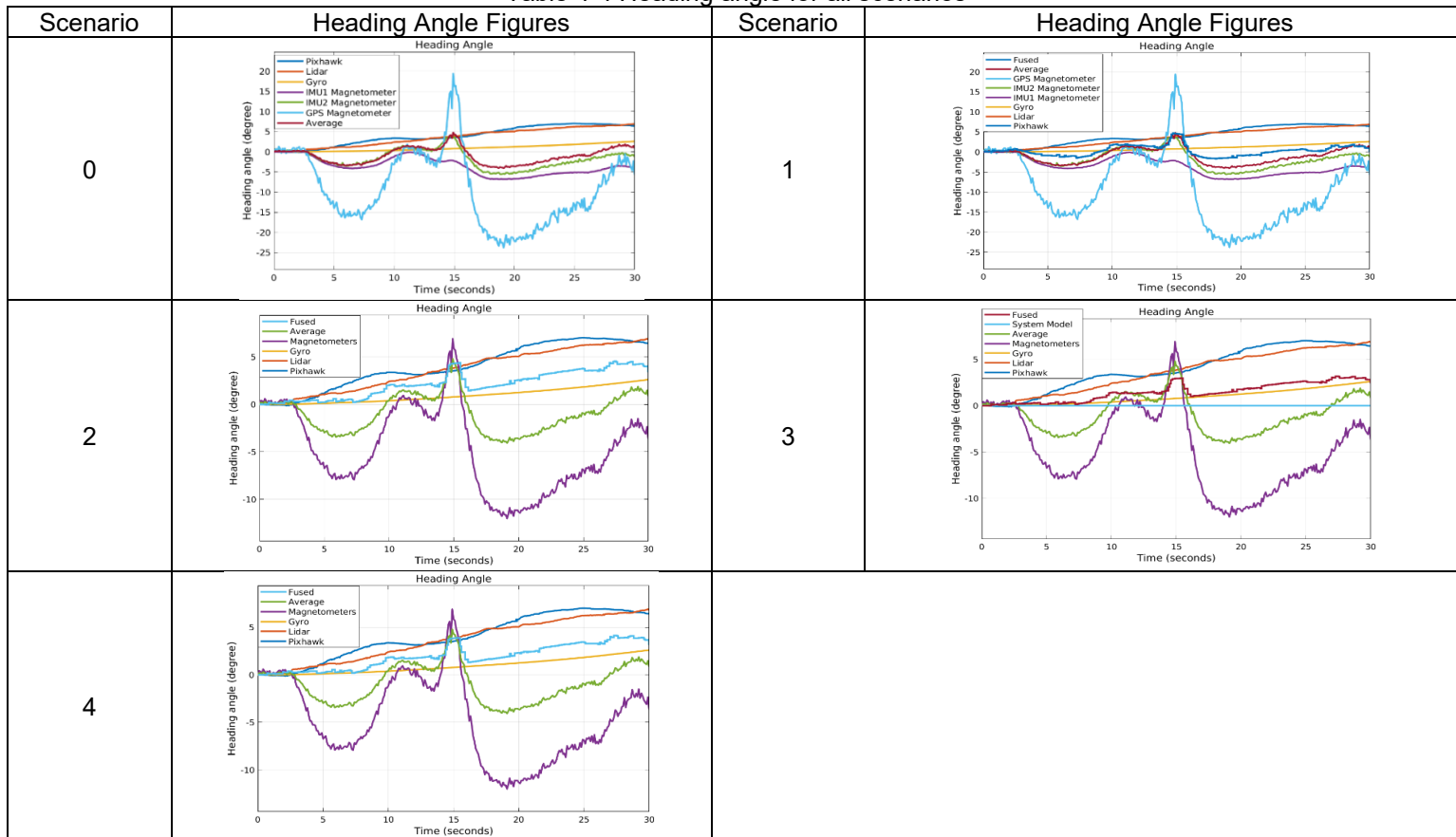
As shown in Table 4-2, including the fusion algorithm and System Model concludes a closer estimate to expected movement of zero meters for Easting. The Easting weights are presented in the Appendix.

Table 4-3 Northing displacement for all scenarios



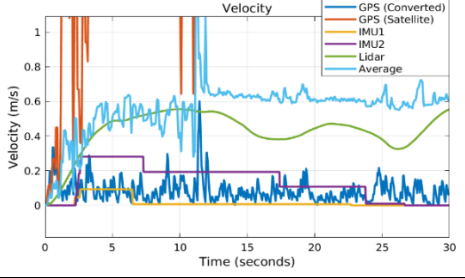
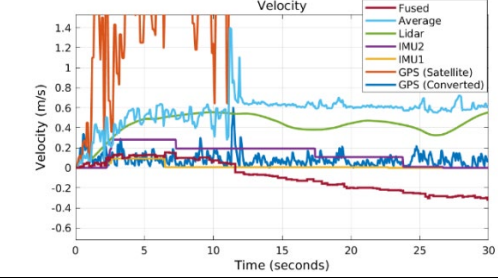
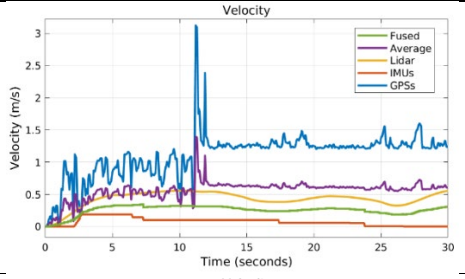
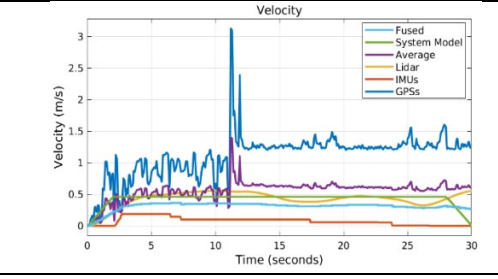
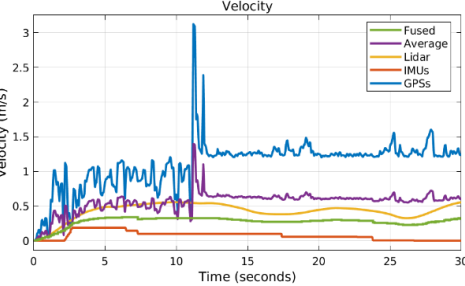
As shown in Table 4-3, including the fusion algorithm and System Model provides a closer estimate to the expected movement of 14 meters for Northing. The Northing weights are presented in the Appendix.

Table 4-4 Heading angle for all scenarios



As shown in Table 4-4, including the fusion algorithm and System Model produces a closer estimate to the expected range of approximately 0 degrees for Heading. Heading weights are presented in the Appendix.

Table 4-5 Velocity for all scenarios

Scenario	Velocity Figures	Scenario	Velocity Figures
0	 <p>Velocity (m/s) vs Time (seconds) for Scenario 0. The plot shows multiple data series: GPS (Converted) in blue, GPS (Satellite) in orange, IMU1 in purple, IMU2 in green, Lidar in yellow, and Average in cyan. The y-axis ranges from 0 to 1.0 m/s, and the x-axis ranges from 0 to 30 seconds. The Average line shows a steady increase from 0 to approximately 0.6 m/s.</p>	1	 <p>Velocity (m/s) vs Time (seconds) for Scenario 1. The plot shows: Fused in red, Average in cyan, Lidar in green, IMU2 in yellow, IMU1 in purple, GPS (Satellite) in orange, and GPS (Converted) in blue. The y-axis ranges from -0.6 to 1.4 m/s. The Average line increases to about 0.6 m/s, while the Fused line shows a slight decrease over time.</p>
2	 <p>Velocity (m/s) vs Time (seconds) for Scenario 2. The plot shows: Fused in green, Average in purple, Lidar in yellow, IMUs in orange, and GPSs in blue. The y-axis ranges from 0 to 3.0 m/s. The Average line increases to about 0.6 m/s, and the GPSs line shows a sharp spike at 10 seconds.</p>	3	 <p>Velocity (m/s) vs Time (seconds) for Scenario 3. The plot shows: Fused in cyan, System Model in red, Average in purple, Lidar in yellow, IMUs in orange, and GPSs in blue. The y-axis ranges from 0 to 3.0 m/s. The Average line increases to about 0.6 m/s, and the GPSs line shows a sharp spike at 10 seconds.</p>
4	 <p>Velocity (m/s) vs Time (seconds) for Scenario 4. The plot shows: Fused in green, Average in purple, Lidar in yellow, IMUs in orange, and GPSs in blue. The y-axis ranges from 0 to 3.0 m/s. The Average line increases to about 0.6 m/s, and the GPSs line shows a sharp spike at 10 seconds.</p>		

As shown in Table 4-5, including the fusion algorithm and considering sensors in groups produces a closer estimate to the expected range of 0.46 meters/sec movement for Velocity. The Velocity weights are presented in the Appendix.

The following figures show the fusion results. They do not present raw values, as shown in the discussed scenarios, for better concluding remarks and comparisons.

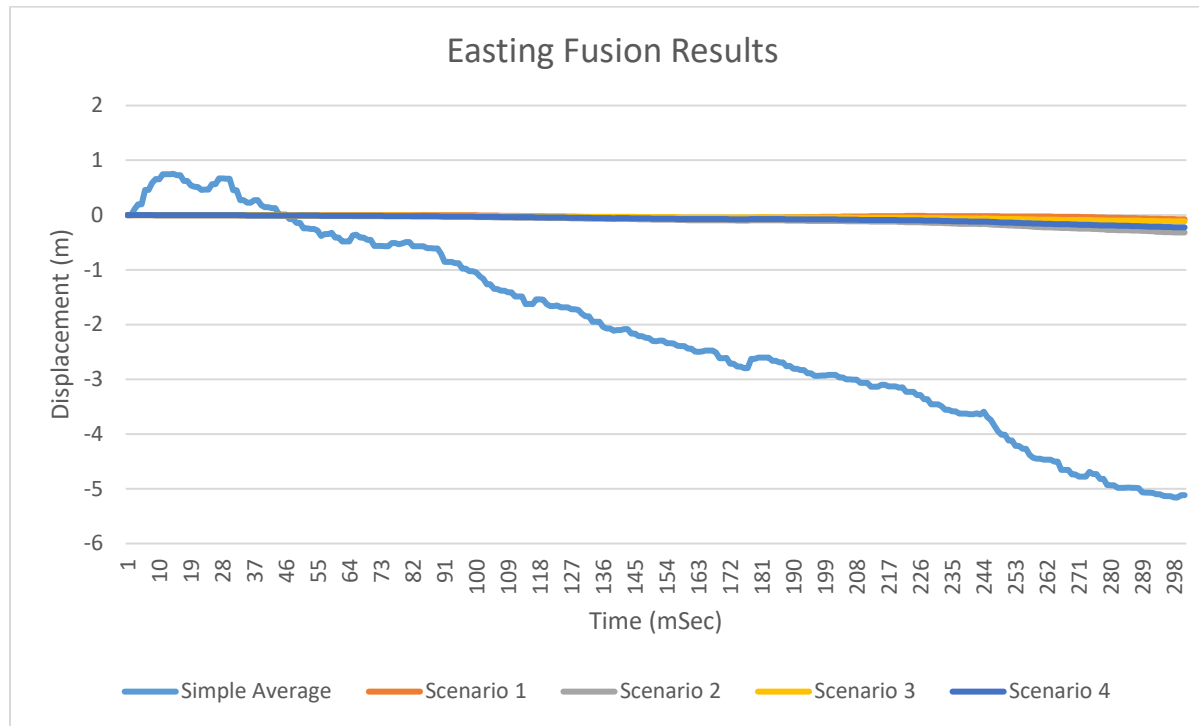


Figure 4-74 Easting fusion results

To better understand the details, a scaled Easting plot is shown next.

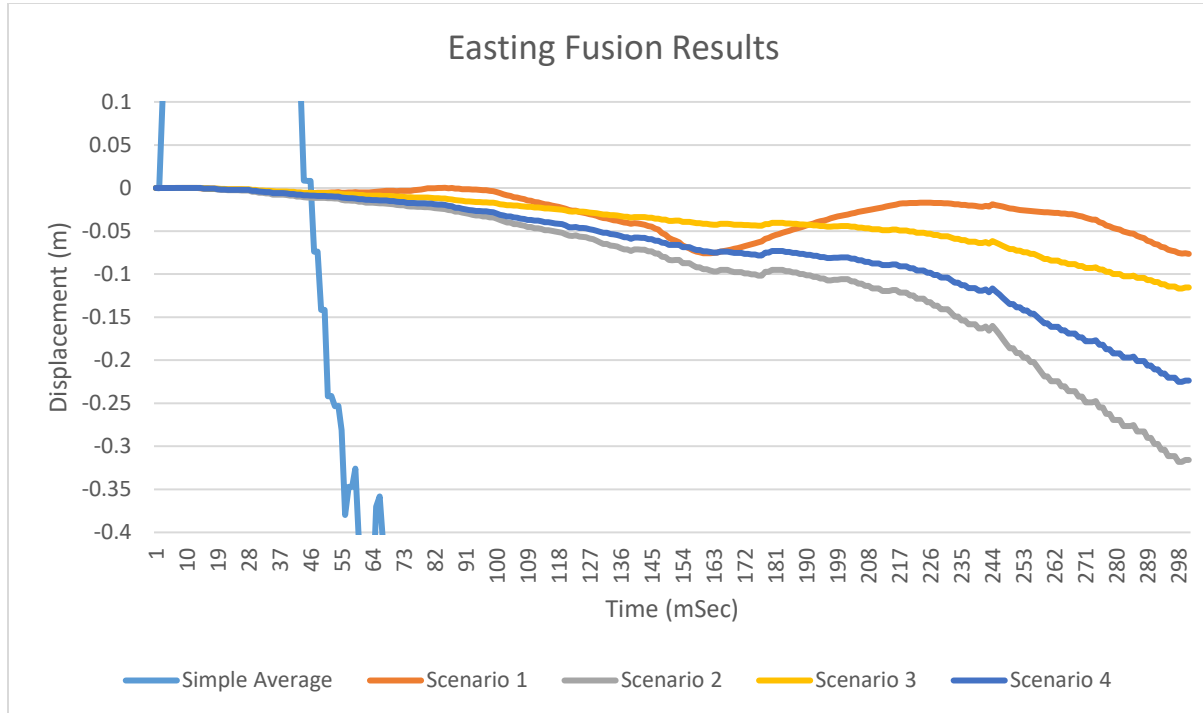


Figure 4-75 Easting fusion results – Scaled

Adding the fusion algorithm in Scenario 1 improved the Easting estimate remarkably. When sensors with the same failure modes are grouped in Scenario 2, the results show less variability and larger range of change. In addition, adding a System Model in Scenarios 3-4 leads to lower variability and better outcomes compared to Scenario 2.

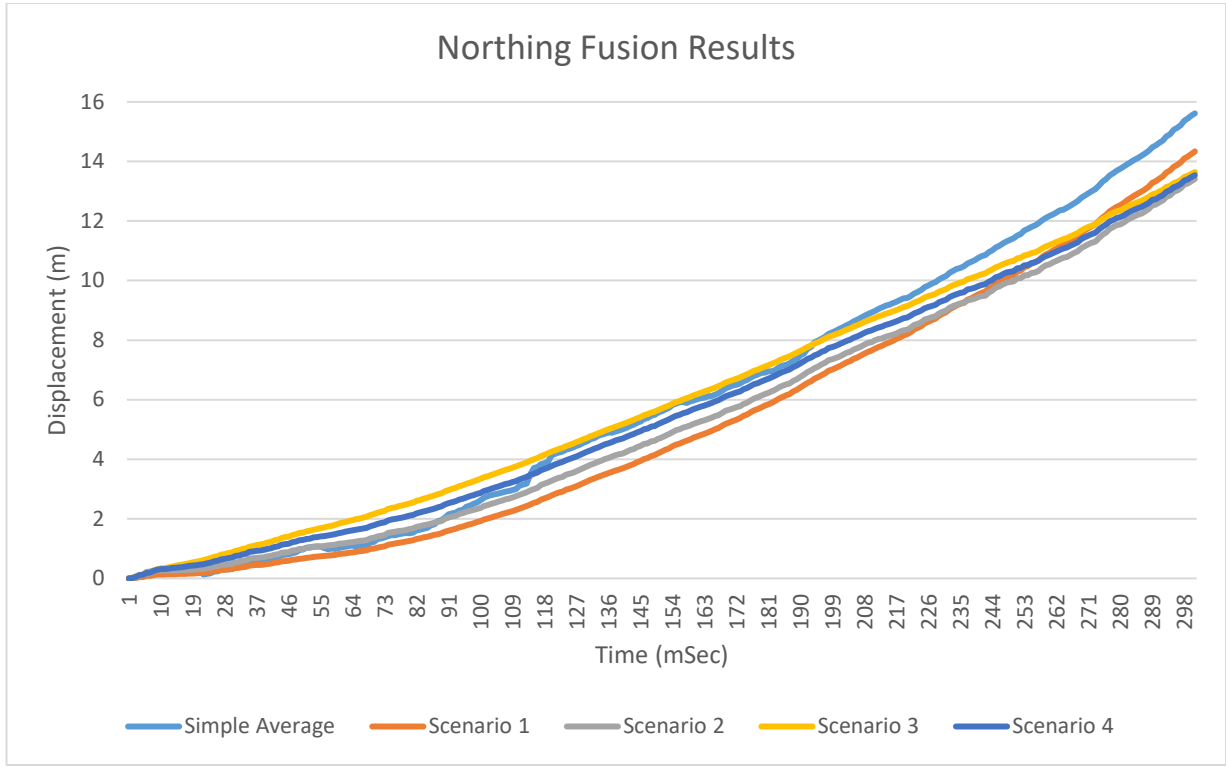


Figure 4-76 Northing fusion results

The scenarios show a close pattern to each other for Northing. However, starting at 23 seconds Scenario 0 (simple average) includes a steeper slope than other scenarios and concludes a total movement of 16 meters. All other scenarios agree with the movement at a range of 14 meters. In addition, less concavity is represented starting from Scenario 1.

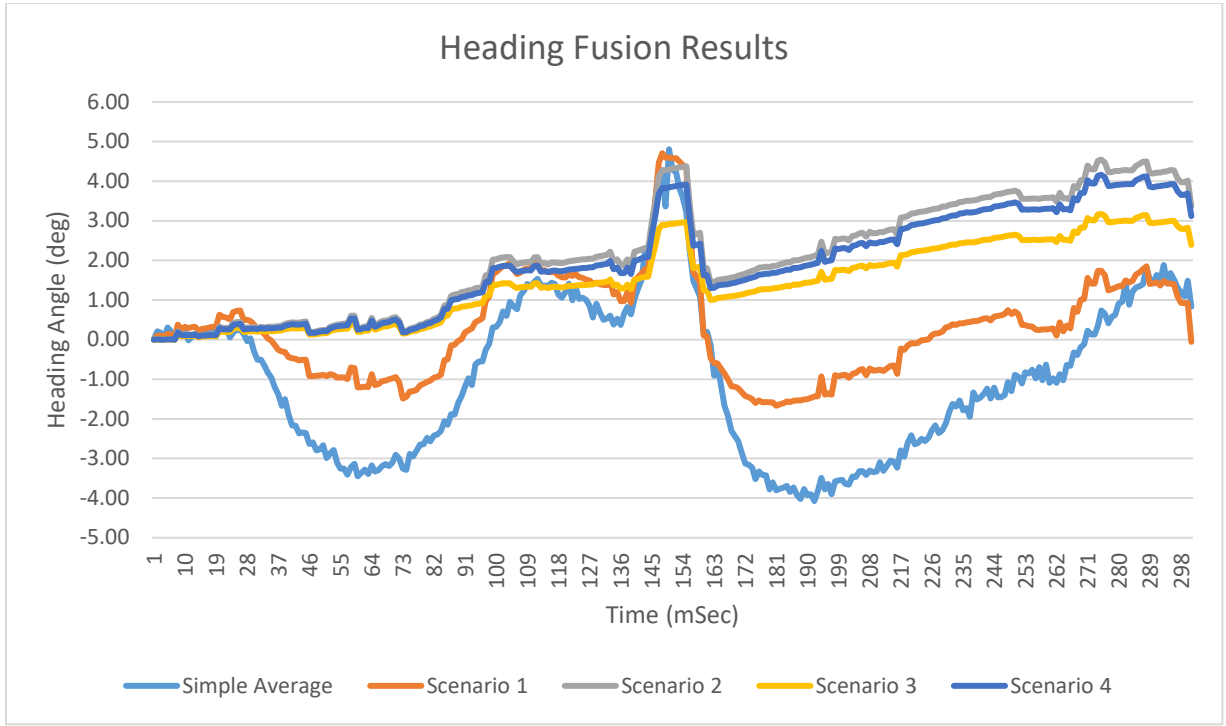


Figure 4-77 Heading fusion results

Adding the fusion algorithm in Scenario 1 improved the Heading estimates. When sensors with the same failure modes are grouped in Scenario 2, the results show less variability and range of change. In addition, adding a System Model in Scenarios 3-4 leads to slightly better outcomes compared to Scenario 2.

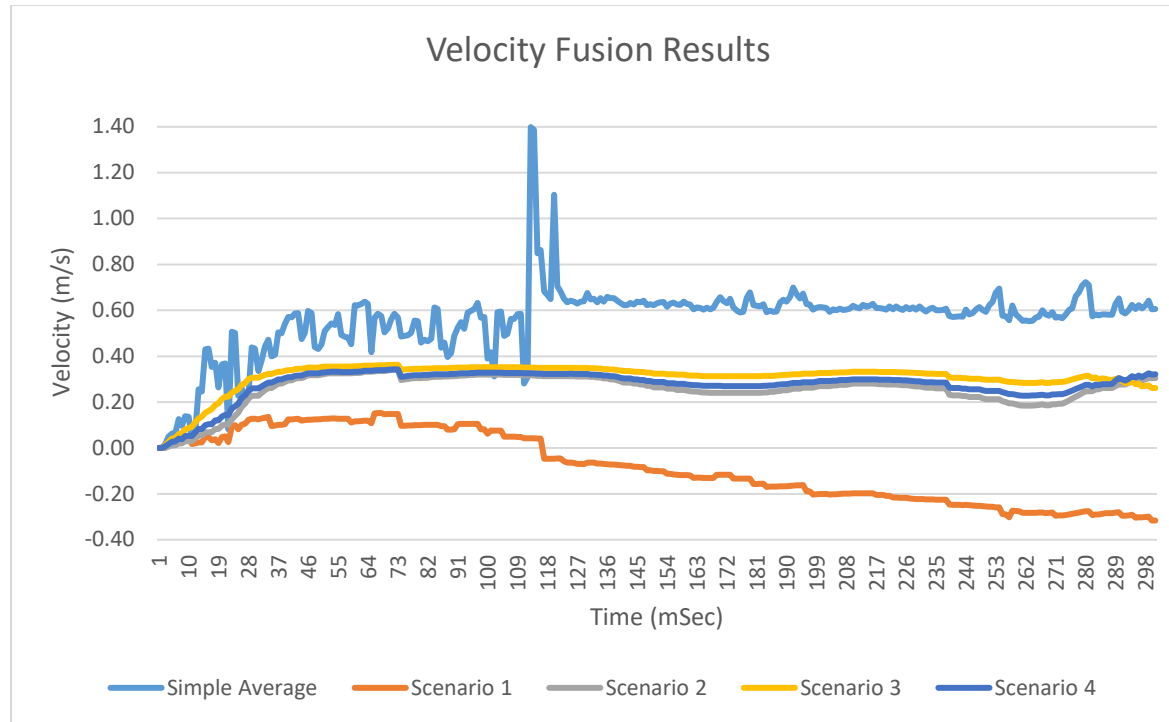


Figure 4-78 Velocity fusion results

Scenario 1 in Velocity fails to output valid estimates. As explained in section 4.5.2, receiving incorrect values from several sensors leads to wrong estimation. However, starting from Scenario 2 the estimated velocity indicates feasible outputs as sensors with the same failure modes are grouped.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

This chapter briefly expresses the achievements throughout the entire research and a few points in regards to the future work.

5.1 Concluding Remarks

This research designed an integrated system to implement dynamic sensor fusion using a set of homogeneous and heterogeneous sensors in a navigational framework. It was discussed and implemented such that it would be extendable to other applications as well where sources of evidence and noise exist. The adaptive concept was proposed which implied that sensors were contributing after they are compared to each other as well as sources of evidence which led to a more reliable way of fusing. The measuring parameters to accomplish the experimental components included the followings:

- Easting estimated position using GPS, IMU, and Lidar sensors,
- Northing estimated position using GPS, IMU, and Lidar sensors,
- Heading angle using GPS, IMU, digital compass, and Lidar sensors,
- Velocity using GPS, IMU, and Lidar sensors.

Direct and converted values from sensors were utilized where required. This allowed for additional sources of evidence that each sensor's output could be reviewed to ensure reliable contribution from every sensor.

An argument on how good the discussed algorithm is and how it works includes the fact that there might be several facts or pieces of evidence for any application which can be helpful to reduce the limitations of the decision-making process. This information could

be the kinematics of the system or other relevant/recorded data. The significance is that they should be transformed from concept, theory or fact to comprehensible, easily-readable, and mathematical outputs. This data extraction also requires an appropriate understanding of the system and subsystems characteristics. In other words, one should study and understand the system with which they are working to be able to access how well it can operate under different situations.

Furthermore, this pre-processing filter controls the inputs and does the fusion operation subsequently. Even for a well-known fusion algorithm such as Kalman filter, the data is inputted without prior cleaning. Kalman filter also assumes Gaussian sensor errors but this study avoided the assumption regarding sensor errors distribution. The strategy presented in this research includes less complicated computations since limited outlying data can enter the system.

The experiments showed that the averaging method can reflect sudden variations caused by the distorted data in only one sensor. However, the proposed fusion methodology generated a smoother trend and avoided sharp/sudden shifts.

After implementing a second experimental run under the new environment, the results showed that when a group of sensors with the same failure modes are applied, they need to be grouped and not treated individually. Considering them separately creates a bias for a higher voting permission. This occurred in the first experimental run. The second experiment, however, applied several improvements.

5.2 Future Work

Adding more and diverse sources (sensors) according to the user's knowledge may improve the decision-making process under more complicated and severe circumstances.

This diversity needs relevant sources according to the application based on the user's selection. In the navigational framework, dead-reckoning support can also be considered in which encoders provide estimation on the travelled distance.

This research is applying the slowest sensor (GPS – 5 hz) as the leading sensor to determine the data collection frequency. Faster sensors such as an IMU (with a frequency of 1000 hz – 1000 data points in each second) only send in the data at a rate of the slowest sensor to match timing. This means that a considerable amount of the data from faster sensors is not being used. As an example, while GPS can send only two data points at times 0 and 0.2 (frequency of 5 hz, which means one data point every 0.2 of a second), IMU can send additional data points between times 0 and 0.2. IMU has a frequency of 1000 hz meaning that it can send one data point every 0.001 of a second which is remarkably faster than the GPS. However, this research picks data from IMU only at time steps that GPS has a new data point available (every 0.2 of a second; times 0, 0.2, 0.4, 0.6, ...). One may try to include all of the data from faster sensors (IMU in this application) to increase accuracy and robustness.

The algorithm applied in this research is extendable. It uses delta values as a basis for comparisons and weight assignment. Conducting comparisons and allocating weights can be done for actual values as well. In addition, combining the two mentioned strategies may lead to increased improvement which can be further examined in future work.

APPENDIX

Difference Between Two Consecutive Data

```
function [x1,u3]= fcn(u1,u2)
    u3=u1;
    x1=u1-u2;
```

Fusion Algorithm – Easting Displacement

```
function [d,I,Flag,Estimate,W1,W2,W3,W4,P1,P2,P3,P4,P5,P6] = fcn(I1,I2,I3,I4)
%% Missing data %%
% This code shows if a piece of data in a particular period is missing,
% replace that by a big number so that it is actually removed
% from calculations
if isempty(abs(I1(1)))
    I1 = 1000;
end
if isempty(abs(I2(1)))
    I2 = 1000;
end
if isempty(abs(I3(1)))
    I3 = 1000;
end
if isempty(abs(I4(1)))
    I4 = 1000;
end
%%
% Finding the distances between the nodes (sensors)
d1 = max(I1(1),I2(1))-min(I1(1),I2(1)); d2 = max(I1(1),I3(1))-min(I1(1),I3(1));
d3 = max(I2(1),I3(1))-min(I2(1),I3(1)); d4 = max(I3(1),I4(1))-min(I3(1),I4(1));
d5 = max(I2(1),I4(1))-min(I2(1),I4(1)); d6 = max(I4(1),I1(1))-min(I4(1),I1(1));
% Finding the furthest distance
```



```

u = max([d1,d2,d3,d4,d5,d6]);
% Giving the furthest edge(s) the lowest weights and the closest the highest
P1 = 1-(d1/u); P2 = 1-(d2/u); P3 = 1-(d3/u); P4 = 1-(d4/u); P5 = 1-(d5/u); P6 = 1-(d6/u);
% Normalizing the weights
P = (P1+P2+P3+P4+P5+P6);
N1 = P1/(P); N2 = P2/(P); N3 = P3/(P); N4 = P4/(P); N5 = P5/(P); N6 = P6/(P);
% Assigning each node (sensor) a portion of its neighbors` weights
W1 = sum(N1+N2+N6)/2; W2 = sum(N1+N3+N5)/2;
W3 = sum(N2+N3+N4)/2; W4 = sum(N4+N5+N6)/2;
%%
W = W1+W2+W3+W4;
W1 = W1/W; W2 = W2/W; W3 = W3/W; W4 = W4/W;
Estimate = W1*I1(1)+W2*I2(1)+W3*I3(1)+W4*I4(1);
Flag = 1;
%%
I = [I1(1) I2(1) I3(1) I4(1)];
d = [d1 d2 d3 d4 d5 d6];

```

Fusion Algorithm – Northing Displacement

```

function [d,I,Flag,Estimate,W1,W2,W3,W4,P1,P2,P3,P4,P5,P6] = fcn(I1,I2,I3,I4)
%% Missing data %%
% This code shows if a piece of data in a particular period is missing,
% replace that by a big number so that it is actually removed
% from calculations
if isempty(abs(I1(1)))
    I1 = 1000;
end
if isempty(abs(I2(1)))
    I2 = 1000;
end
if isempty(abs(I3(1)))
    I3 = 1000;
end

```

```

if isempty(abs(I4(1)))
    I4 = 1000;
end
%%
% Finding the ditances between the nodes (sensors)
d1 = max(I1(1),I2(1))-min(I1(1),I2(1)); d2 = max(I1(1),I3(1))-min(I1(1),I3(1));
d3 = max(I2(1),I3(1))-min(I2(1),I3(1)); d4 = max(I3(1),I4(1))-min(I3(1),I4(1));
d5 = max(I2(1),I4(1))-min(I2(1),I4(1)); d6 = max(I4(1),I1(1))-min(I4(1),I1(1));
% Finding the furthest distance
u = max([d1,d2,d3,d4,d5,d6]);
% Giving the furthest edge(s) the lowest weights and the closest the highest
P1 = 1-(d1/u); P2 = 1-(d2/u); P3 = 1-(d3/u); P4 = 1-(d4/u); P5 = 1-(d5/u); P6 = 1-(d6/u);
% Normalizing the weights
P = (P1+P2+P3+P4+P5+P6);
N1 = P1/(P); N2 = P2/(P); N3 = P3/(P); N4 = P4/(P); N5 = P5/(P); N6 = P6/(P);
% Assigning each node (sensor) a portion of its neighbors` weights
W1 = sum(N1+N2+N6)/2; W2 = sum(N1+N3+N5)/2;
W3 = sum(N2+N3+N4)/2; W4 = sum(N4+N5+N6)/2;
%%
W = W1+W2+W3+W4;
W1 = W1/W; W2 = W2/W; W3 = W3/W; W4 = W4/W;
Estimate = W1*I1(1)+W2*I2(1)+W3*I3(1)+W4*I4(1);
Flag = 1;
%%
I = [I1(1) I2(1) I3(1) I4(1)];
                                d = [d1 d2 d3 d4 d5 d6];

```

Fusion Algorithm – Heading Angle

```

function [d,I,Flag,Estimate,W1,W2,W3,W4,W5,W6,...
P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15] = fcn(I1,I2,I3,I4,I5,I6)
%% Missing data %%
% This code shows if a piece of data in a particular period is missing,
% replace that by a big number so that it is actually removed

```

```

% from calculations
if isempty(abs(I1(1)))
    I1 = 1000;
end
if isempty(abs(I2(1)))
    I2 = 1000;
end
if isempty(abs(I3(1)))
    I3 = 1000;
end
if isempty(abs(I4(1)))
    I4 = 1000;
end
if isempty(abs(I5(1)))
    I5 = 1000;
end
if isempty(abs(I6(1)))
    I6 = 1000;
end
%%
% Finding the ditances between the nodes (sensors)
d1 = max(I1(1),I2(1))-min(I1(1),I2(1)); d2 = max(I1(1),I3(1))-min(I1(1),I3(1));
d3 = max(I2(1),I3(1))-min(I2(1),I3(1)); d4 = max(I3(1),I4(1))-min(I3(1),I4(1));
d5 = max(I2(1),I4(1))-min(I2(1),I4(1)); d6 = max(I4(1),I1(1))-min(I4(1),I1(1));
d7 = max(I1(1),I5(1))-min(I1(1),I5(1)); d8 = max(I5(1),I2(1))-min(I5(1),I2(1));
d9 = max(I5(1),I4(1))-min(I5(1),I4(1)); d10 = max(I5(1),I3(1))-min(I5(1),I3(1));
d11 = max(I6(1),I1(1))-min(I6(1),I1(1)); d12 = max(I6(1),I2(1))-min(I6(1),I2(1));
d13 = max(I6(1),I3(1))-min(I6(1),I3(1)); d14 = max(I6(1),I4(1))-min(I6(1),I4(1));
d15 = max(I6(1),I5(1))-min(I6(1),I5(1));
% Finding the furthest distance
u = max([d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,d12,d13,d14,d15]);
% Giving the furthest edge(s) the lowest weights and the closest the highest
P1 = 1-(d1/u); P2 = 1-(d2/u); P3 = 1-(d3/u); P4 = 1-(d4/u); P5 = 1-(d5/u); P6 = 1-(d6/u);

```

```

P7 = 1-(d7/u); P8 = 1-(d8/u); P9 = 1-(d9/u); P10 = 1-(d10/u); P11 = 1-(d11/u);
P12 = 1-(d12/u); P13 = 1-(d13/u); P14 = 1-(d14/u); P15 = 1-(d15/u);
% Normalizing the weights
P = (P1+P2+P3+P4+P5+P6+P7+P8+P9+P10+P11+P12+P13+P14+P15);
N1 = P1/(P); N2 = P2/(P); N3 = P3/(P); N4 = P4/(P); N5 = P5/(P);
N6 = P6/(P); N7 = P7/(P); N8 = P8/(P); N9 = P9/(P); N10 = P10/(P);
N11 = P11/(P); N12 = P12/(P); N13 = P13/(P); N14 = P14/(P); N15 = P15/(P);
% Assigning each node (sensor) a portion of its neighbors` weights
W1 = sum(N1+N2+N6+N7+N11)/2; W2 = sum(N1+N3+N5+N8+N12)/2;
W3 = sum(N2+N3+N4+N10+N13)/2; W4 = sum(N4+N5+N6+N9+N14)/2;
W5 = sum(N7+N8+N9+N10+N15)/2; W6 = sum(N11+N12+N13+N14+N15)/2;
%%
W = W1+W2+W3+W4+W5+W6;
W1 = W1/W; W2 = W2/W; W3 = W3/W; W4 = W4/W; W5 = W5/W; W6 = W6/W;
Estimate = W1*I1(1)+W2*I2(1)+W3*I3(1)+W4*I4(1)+W5*I5(1)+W6*I6(1);
Flag = 1;
%%
I = [I1(1) I2(1) I3(1) I4(1) I5(1) I6(1)];
d = [d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15];

```

Fusion Algorithm – Velocity

```

function [d,I,Flag,Estimate,W1,W2,W3,W4,W5,...
P1,P2,P3,P4,P5,P6,P7,P8,P9,P10] = fcn(I1,I2,I3,I4,I5)
%% Missing data %%
% This code shows if a piece of data in a particular period is missing,
% replace that by a big number so that it is actually removed
% from calculations
if isempty(abs(I1(1)))
    I1 = 1000;
end
if isempty(abs(I2(1)))
    I2 = 1000;
end

```

```

if isempty(abs(I3(1)))
    I3 = 1000;
end
if isempty(abs(I4(1)))
    I4 = 1000;
end
if isempty(abs(I5(1)))
    I5 = 1000;
end
%%
% Finding the ditances between the nodes (sensors)
d1 = max(I1(1),I2(1))-min(I1(1),I2(1)); d2 = max(I1(1),I3(1))-min(I1(1),I3(1));
d3 = max(I2(1),I3(1))-min(I2(1),I3(1)); d4 = max(I3(1),I4(1))-min(I3(1),I4(1));
d5 = max(I2(1),I4(1))-min(I2(1),I4(1)); d6 = max(I4(1),I1(1))-min(I4(1),I1(1));
d7 = max(I1(1),I5(1))-min(I1(1),I5(1)); d8 = max(I5(1),I2(1))-min(I5(1),I2(1));
d9 = max(I5(1),I4(1))-min(I5(1),I4(1)); d10 = max(I5(1),I3(1))-min(I5(1),I3(1));
% Finding the furthest distance
u = max([d1,d2,d3,d4,d5,d6,d7,d8,d9,d10]);
% Giving the furthest edge(s) the lowest weights and the closest the highest
P1 = 1-(d1/u); P2 = 1-(d2/u); P3 = 1-(d3/u); P4 = 1-(d4/u); P5 = 1-(d5/u); P6 = 1-(d6/u);
P7 = 1-(d7/u); P8 = 1-(d8/u); P9 = 1-(d9/u); P10 = 1-(d10/u);
% Normalizing the weights
P = (P1+P2+P3+P4+P5+P6+P7+P8+P9+P10);
N1 = P1/(P); N2 = P2/(P); N3 = P3/(P); N4 = P4/(P); N5 = P5/(P);
N6 = P6/(P); N7 = P7/(P); N8 = P8/(P); N9 = P9/(P); N10 = P10/(P);
% Assigning each node (sensor) a portion of its neighbors` weights
W1 = sum(N1+N2+N6+N7)/2; W2 = sum(N1+N3+N5+N8)/2; W3 =
sum(N2+N3+N4+N10)/2;
W4 = sum(N4+N5+N6+N9)/2; W5 = sum(N7+N8+N9+N10)/2;
%%
W = W1+W2+W3+W4+W5;
W1 = W1/W; W2 = W2/W; W3 = W3/W; W4 = W4/W; W5 = W5/W;
Estimate = W1*I1(1)+W2*I2(1)+W3*I3(1)+W4*I4(1)+W5*I5(1);

```

```

Flag = 1;
%%
I = [I1(1) I2(1) I3(1) I4(1) I5(1)];
d = [d1 d2 d3 d4 d5 d6 d7 d8 d9 d10];

```

IMU Acceleration Filter (Zeroing down)

```

function [y,Flag] = fcn(Evidence,u,PrevEst)
    if u > Evidence || u < -(Evidence)
        y = PrevEst;
        Flag = 1;
    else
        y = u;
        Flag = 2;
    end

```

Determining the Velocity Based on Heading Angle for IMUs (Northing)

```

function y = fcn(vel,hdg)
if (hdg>=0) && (hdg<=45)
    y = 0+vel*cosd(hdg);
elseif (hdg>45)&&(hdg<90)
    y = vel*sind(90-hdg);
elseif hdg == 90
    y = 0;
elseif (hdg>90)&&(hdg<=135)
    y = vel*sind(hdg-90);
elseif (hdg>135) && (hdg<180)
    y = 0+vel*cosd(hdg);
elseif hdg == 180
    y = -vel;
elseif (hdg>180) && (hdg<225)
    y = 0+vel*cosd(hdg);
elseif (hdg>=225)&&(hdg<270)

```

```

    y = vel*sind(hdg-90);
elseif hdg == 270
    y = 0;
elseif (hdg>270)&&(hdg<=315)
    y = vel*cosd(hdg-90);
elseif (hdg>315) && (hdg<=360)
    y = 0+vel*cosd(hdg);
else
    y = vel;
end

```

Determining the Velocity Based on Heading Angle for IMUs (Easting)

```

function y = fcn(vel,hdg)
if (hdg>=0) && (hdg<=45)
    y = 0+vel*sind(hdg);
elseif (hdg>45)&&(hdg<90)
    y = vel*cosd(90-hdg);
elseif hdg == 90
    y = vel;
elseif (hdg>90)&&(hdg<=135)
    y = vel*cosd(hdg-90);
elseif (hdg>135) && (hdg<=180)
    y = 0+vel*sind(hdg);
elseif (hdg>=180) && (hdg<225)
    y = 0+vel*sind(hdg);
elseif (hdg>=225)&&(hdg<270)
    y = vel*cosd(hdg-90);
elseif hdg == 270
    y = -vel;
elseif (hdg>270)&&(hdg<=315)
    y = vel*cosd(hdg-90);
elseif (hdg>315) && (hdg<=360)
    y = 0+vel*sind(hdg);

```

```
else
    y = vel;
end
```

Lidar MatchScan

```
function [x,y,Heading] = Lidar(u1,u2)
refRanges = u1;
refAngles = linspace(-pi,pi,1081);
currRanges = u2;
currAngles = linspace(-pi,pi,1081);
pose = matchScans(currRanges,currAngles,refRanges,refAngles);
x = pose(1);
y = pose(2);
Heading = pose(3)*180/pi;
if Heading > 360 && Heading < 720
    Heading = Heading - 360;
elseif Heading < -360 && Heading > -720
    Heading = Heading + 360;
end
```

GPS Magnetic Field Conversion (1)

```
function Direction = fcn(x,y)
if y>0
    Direction = 90-(atan(x/y))*(180/pi);
elseif y<0
    Direction = 270-(atan(x/y))*(180/pi);
elseif y==0 && x<0
    Direction = 180;
elseif y==0 && x>0
    Direction = 0;
else
    Direction = 1000;
```


end

GPS Magnetic Field Conversion (2)

```
function y = fcn(u)
```

```
if u > 360
```

```
    y = u-360;
```

```
else
```

```
    y = u;
```

```
end
```

Table A-1 Easting weights for all scenarios



Table A-2 Northing weights for all scenarios

Scenario	Northing Weights Figures	Scenario	Northing Weights Figures
1	<p>Percentage</p> <p>Time (mSec)</p> <p>■ IMU1 ■ IMU2 ■ GPS ■ Lidar</p>	2	<p>Percentage</p> <p>Time (mSec)</p> <p>■ IMUs ■ GPS ■ Lidar</p>
3	<p>Percentage</p> <p>Time (mSec)</p> <p>■ IMUs ■ GPS ■ Lidar ■ System Model</p>	4	<p>Percentage</p> <p>Time (mSec)</p> <p>■ IMUs ■ GPS ■ Lidar</p>

Table A-3 Heading angle weights for all scenarios

Scenario	Heading Weights Figures	Scenario	Heading Weights Figures
1	<p>Stacked area chart for Scenario 1. The y-axis represents Percentage (0% to 100%) and the x-axis represents Time (mSec) from 1 to 301. The legend includes Pixhawk (blue), Lidar (red), Gyro (green), IMU1 Magnetometer (purple), IMU2 Magnetometer (cyan), and GPS Magnetometer (orange). The weights fluctuate significantly over time, with IMU2 Magnetometer and IMU1 Magnetometer showing high variability.</p>	2	<p>Stacked area chart for Scenario 2. The y-axis represents Percentage (0% to 100%) and the x-axis represents Time (mSec) from 1 to 295. The legend includes Pixhawk (blue), Lidar (red), Gyro (green), and Magnetometers (IMU1, IMU2, GPS) (purple). The weights fluctuate, with Magnetometers and Lidar showing significant contributions.</p>
3	<p>Stacked area chart for Scenario 3. The y-axis represents Percentage (0% to 100%) and the x-axis represents Time (mSec) from 1 to 301. The legend includes Pixhawk (blue), Lidar (red), Gyro (green), Magnetometers (IMU1, IMU2, GPS) (purple), and System Model (cyan). The System Model weight is notably high, often exceeding 50%.</p>	4	<p>Stacked area chart for Scenario 4. The y-axis represents Percentage (0% to 100%) and the x-axis represents Time (mSec) from 1 to 295. The legend includes Pixhawk (blue), Lidar (red), Gyro (green), and Magnetometers (IMU1, IMU2, GPS) (purple). The weights fluctuate, with Magnetometers and Lidar showing significant contributions.</p>

Table A-4 Velocity weights for all scenarios



REFERENCES

1. Ahmadi, H., M. Gholamzadeh, L. Shahmoradi, M. Nilashi, and P. Rashvand. 2018. Diseases diagnosis using fuzzy logic methods: A systematic and meta-analysis review. *Computer Methods and Programs in Biomedicine* 161:145-172.
2. Al-Sharman, M. K., B. J. Emran, M. A. Jaradat, H. Najjaran, R. Al-Husari, and Y. Zweiri. 2018. Precision landing using an adaptive fuzzy multi-sensor data fusion architecture. *Applied soft computing* 69:149-164.
3. Barbu, T. 2013. Variational Image Denoising Approach with Diffusion Porous Media Flow. *Abstract and Applied Analysis* 2013:8.
4. Boeing, G. 2016. Visual Analysis of Nonlinear Dynamical Systems: Chaos, Fractals, Self-Similarity and the Limits of Prediction. *Systems* 4.
5. Bogaert, P., and S. Gengler. 2018. Bayesian maximum entropy and data fusion for processing qualitative data: theory and application for crowdsourced cropland occurrences in Ethiopia. *Stochastic Environmental Research and Risk Assessment* 32:815-831.
6. Chang, J., L. Zhu, H. Li, F. Xu, B. Liu, and Z. Yang. 2018. Noise reduction in Lidar signal using correlation-based EMD combined with soft thresholding and roughness penalty. *Optics Communications* 407:290-295.
7. Chromy, A., and O. Klima. 2017. A 3D Scan Model and Thermal Image Data Fusion Algorithms for 3D Thermography in Medicine. *Journal of Healthcare Engineering*:9.
8. Clearpath Robotics. 2014. ROS 101: Intro to the Robot Operating System.
9. Crick, C., G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins. 2017. Rosbridge: ROS for Non-ROS Users. *in* H. I. Christensen and O. Khatib, editors. *Robotics Research*, Isrr. Springer-Verlag Berlin, Berlin.
10. Dempster, A. P. 1967. Upper and Lower Probabilities Induced by a Multivalued Mapping. *Ann. Math. Statist.* 38:325-339.
11. Ding, I. J., and S. K. Lin. 2017. Performance Improvement of Kinect Software Development Kit-Constructed Speech Recognition Using a Client-Server Sensor Fusion Strategy for Smart Human-Computer Interface Control Applications. *IEEE* 5:4154-4162.
12. Djuric, P. M., J. H. Kotecha, J. Q. Zhang, Y. F. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez. 2003. Particle filtering. *Ieee Signal Processing Magazine* 20:19-38.
13. Du, J., Q. Wang, Y. H. Tu, X. Bao, L. R. Dai, and C. H. Lee. 2015. An Information Fusion Approach To Recognizing Microphone Array Speech In The Chime-3 Challenge Based On A Deep Learning Framework. *IEEE*, New York.
14. Elkin, C., R. Kumarasiri, D. B. Rawat, and V. Devabhaktuni. 2017. Localization in wireless sensor networks: A Dempster-Shafer evidence theoretical approach. *Ad Hoc Networks* 54:30-41.
15. Fung, M. L., M. Z. Q. Chen, and Y. H. Chen. 2017. Sensor fusion: A review of methods and applications. Pages 3853-3860 *in* 2017 29th Chinese Control And Decision Conference (CCDC).
16. Gui, L. Y., X. P. Yang, A. B. Cremers, and Y. Chen. 2017. Dempster-Shafer Evidence Theory-Based CV Model for Renal Lesion Segmentation of Medical Ultrasound Images. *Journal of Medical Imaging and Health Informatics* 7:595-606.
17. Guo, J., X. Yuan, and C. Han. 2017. Sensor selection based on maximum entropy fuzzy clustering for target tracking in large-scale sensor networks. *IET Signal Processing* 11:613-621.

18. Gustafsson, F., F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. J. Nordlund. 2002. Particle filters for positioning, navigation, and tracking. *Ieee Transactions on Signal Processing* 50:425-437.
19. Hou, Y. B., R. Fan, X. Z. Chen, and T. T. Dong. 2016. The compensation algorithm based on Bayes theory of multi point measurement fusion for methane concentration. *IEEE*, New York.
20. Hsiao, K., H. de Plinval-Salgues, and J. Miller. 2005. Particle Filters and Their Applications. *in* MIT, editor., *Cognitive Robotics*.
21. Hu, J., Z. D. Wang, S. Liu, and H. J. Gao. 2016. A variance-constrained approach to recursive state estimation for time-varying complex networks with missing measurements. *Automatica* 64:155-162.
22. Hui, K. H., M. H. Lim, M. S. Leong, and S. M. Al-Obaidi. 2017a. Dempster-Shafer evidence theory for multi-bearing faults diagnosis. *Engineering Applications of Artificial Intelligence* 57:160-170.
23. Hui, K. H., M. H. Lim, S. Leong, and Asme. 2017b. DEMPSTER-SHAFER-BASED SENSOR FUSION APPROACH FOR MACHINERY FAULT DIAGNOSIS. *Amer Soc Mechanical Engineers*, New York.
24. Hui, K. H., C. S. Ooi, M. H. Lim, and M. S. Leong. 2016. A hybrid artificial neural network with Dempster-Shafer theory for automated bearing fault diagnosis. *Journal of Vibroengineering* 18:4409-4418.
25. Ji, S., Z. Y. Chen, P. Guo, Y. J. Sun, J. Shen, J. Wang, and C. F. Lai. 2016. Bayesian Approach for Multi-Sensor Data Fusion Based on Compressed Sensing for Wireless Structural Damage Signal. *Journal of Internet Technology* 17:1363-1371.
26. Jian, Z., C. Hongbing, S. Jie, and L. Haitao. 2011. Data Fusion for Magnetic Sensor Based on Fuzzy Logic Theory. Pages 87-92 *in* 2011 Fourth International Conference on Intelligent Computation Technology and Automation.
27. Jiang, W., W. W. Hu, and C. H. Xie. 2017. A New Engine Fault Diagnosis Method Based on Multi-Sensor Data Fusion. *Applied Sciences-Basel* 7:18.
28. Jiang, W., B. Y. Wei, C. H. Xie, and D. Y. Zhou. 2016. An evidential sensor fusion method in fault diagnosis. *Advances in Mechanical Engineering* 8:7.
29. Julier, S. J., and J. K. Uhlmann. 1997. A new extension of the Kalman filter to nonlinear systems. *Spie - Int Soc Optical Engineering*, Bellingham.
30. Julier, S. J., and J. K. Uhlmann. 2004. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE* 92:401-422.
31. Kalman, R. E. 1960. A New Approach to Linear Filtering and Prediction Problems. *Transactions of ASME, Series D, Journal of Basic Engineering* 82:35-45.
32. Kasebzadeh, P., G. S. Granados, and E. S. Lohan. 2014. Indoor localization via WLAN path-loss models and Dempster-Shafer combining. *in* J. Nurmi, L. Ruotsalainen, E. S. Lohan, J. Salcedo, and S. Thombre, editors. *Proceedings of 2014 International Conference on Localization and Gns. IEEE*, New York.
33. Khazaee, M., A. Banakar, B. Ghobadian, M. Mirsalim, S. Minaei, M. Jafari, and P. Sharghi. 2016. Fault detection of engine timing belt based on vibration signals using data-mining techniques and a novel data fusion procedure. *Structural Health Monitoring-an International Journal* 15:583-598.
34. Kim, J., and S. Lee. 2016. A vehicular positioning with GPS/IMU using adaptive control of filter noise covariance. *ICT Express* 2:41-46.
35. Li, J. G., J. Y. Zhou, J. Yang, J. Liu, F. L. Zeng, and L. Yang. 2015a. Localizing Multiple Odor Sources with a Mobile Robot in Time-varying Airflow Environments using Dempster-Shafer Inference. Pages 6072-6077 *in* Q. Zhao and S. Liu, editors. 2015 34th Chinese Control Conference. *IEEE*, New York.

36. Li, S. B., H. Ma, T. Saha, and G. N. Wu. 2018. Bayesian information fusion for probabilistic health index of power transformer. *IET Generation Transmission & Distribution* 12:279-287.
37. Li, Z., G. Wen, and N. Xie. 2015b. An approach to fuzzy soft sets in decision making based on grey relational analysis and Dempster–Shafer theory of evidence: An application in medical diagnosis. *Artificial Intelligence in Medicine* 64:161-171.
38. Lin, K. H., Z. T. Xu, M. Qiu, X. L. Wang, and T. X. Han. 2016. Noise Filtering, Trajectory Compression and Trajectory Segmentation on GPS Data. Pages 490-495 2016 11th International Conference on Computer Science & Education. IEEE, New York.
39. Lv, F., N. Du, and H. L. Du. 2012. A method of multi-classifier combination based on Dempster-Shafer evidence theory and the application in the fault diagnosis. Pages 1402-1406 *in* R. Chen and W. P. Sung, editors. *Mechatronics and Intelligent Materials II, Pts 1-6*. Trans Tech Publications Ltd, Stafa-Zurich.
40. Majumder, S., and D. K. Pratihari. 2018. Multi-sensors data fusion through fuzzy clustering and predictive tools. *Expert Systems with Applications* 107:165-172.
41. Markovic, I., and I. Petrovic. 2014. Bayesian Sensor Fusion Methods for Dynamic Object Tracking - A Comparative Study. *Automatika* 55:386-398.
42. Martinez Romero, A. 2014. ROS Computation Graph Level.
43. Mehranfar, A., N. Ghadiri, M. Kouhsar, and A. Golshani. 2017. A Type-2 fuzzy data fusion approach for building reliable weighted protein interaction networks with application in protein complex detection. *Computers in Biology and Medicine* 88:18-31.
44. Meier, L. 2019. <https://pixhawk.org>.
45. Mishra, R. 2013. Optimized Estimates Based On Multiple Sensor Configuration Knowledge. The University of Texas at Arlington.
46. Mitra, V., H. Franco, and A. Int Speech Commun. 2016. Coping with Unseen Data Conditions: Investigating Neural Net Architectures, Robust Features, and Information Fusion for Robust Speech Recognition. Pages 3783-3787 17th Annual Conference of the International Speech Communication Association. Isca-Int Speech Communication Assoc, Baixas.
47. Mohamed, A. H., and K. P. Schwarz. 1999. Adaptive Kalman filtering for INS GPS. *Journal of Geodesy* 73:193-203.
48. Moraru, L., C. D. Obreja, N. Dey, and A. S. Ashour. 2018. Dempster-Shafer Fusion for Effective Retinal Vessels' Diameter Measurement. Pages 149-160 *Soft Computing Based Medical Image Analysis*. Academic Press.
49. Mulyani, Y., E. F. Rahman, Herbert, and L. S. Riza. 2016. A New Approach on Prediction of Fever Disease by Using a Combination of Dempster Shafer and Naive Bayes. IEEE, New York.
50. Munguía, R. 2014. A GPS-aided inertial navigation system in direct configuration. *Journal of Applied Research and Technology* 12:803-814.
51. Nada, D., M. Bousbia-Salah, and M. Bettayeb. 2018. Multi-sensor Data Fusion for Wheelchair Position Estimation with Unscented Kalman Filter. *International Journal of Automation and Computing* 15:207-217.
52. Oxford University Press. 2019. Oxford Dictionary. Oxford University Press, <https://en.oxforddictionaries.com/definition/evidence>.
53. Pan, T. Y., C. H. Kuo, and M. C. Hu. 2016. A Noise Reduction Method For Imu And Its Application On Handwriting Trajectory Reconstruction. 2016 IEEE International Conference on Multimedia & Expo Workshops. IEEE, New York.
54. Safavi, S., and I. Mporas. 2017. COMBINATION OF RULE-BASED AND DATA-DRIVEN FUSION METHODOLOGIES FOR DIFFERENT SPEAKER VERIFICATION MODES OF OPERATION. IEEE, New York.

55. Seiti, H., and A. Hafezalkotob. 2018. Developing pessimistic–optimistic risk-based methods for multi-sensor fusion: An interval-valued evidence theory approach. *Applied soft computing* 72:609-623.
56. Selin, I. 1964. The Kalman filter and nonlinear estimates of multivariate normal processes. *IEEE transactions on automatic control* 9:319-319.
57. Shafer, G. 1976. *A Mathematical Theory of Evidence*. Technometrics 20.
58. Stover, J. A., D. L. Hall, and R. E. Gibson. 1996. A fuzzy-logic architecture for autonomous multisensor data fusion. *Ieee Transactions on Industrial Electronics* 43:403-410.
59. Sun, Y. L., L. W. Guan, Z. Y. Chang, C. J. Li, and Y. B. Gao. 2019. Design of a Low-Cost Indoor Navigation System for Food Delivery Robot Based on Multi-Sensor Information Fusion. *Sensors* 19:26.
60. Tang, Y. C., D. Y. Zhou, M. Y. Zhuang, X. Y. Fang, and C. H. Xie. 2017. An Improved Evidential-IOWA Sensor Data Fusion Approach in Fault Diagnosis. *Sensors* 17:15.
61. Thrun, S. 2002. Particle Filters in Robotics. *in* Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI).
62. Wan, E. A., and R. van der Merwe. 2000. *The unscented Kalman Filter for nonlinear estimation*. IEEE, New York.
63. Wang, D. S., Y. J. Lu, L. Zhang, and G. P. Jiang. 2019. Intelligent Positioning for a Commercial Mobile Platform in Seamless Indoor/Outdoor Scenes based on Multi-sensor Fusion. *Sensors* 19:18.
64. Wang, J., Y. Hu, F. Xiao, X. Deng, and Y. Deng. 2016. A novel method to use fuzzy soft sets in decision making based on ambiguity measure and Dempster–Shafer theory of evidence: An application in medical diagnosis. *Artificial Intelligence in Medicine* 69:1-11.
65. Wieland, A., and C. Marcus Wallenburg. 2012. Dealing with supply chain risks: Linking risk management practices and strategies to performance. *International Journal of Physical Distribution & Logistics Management* 42:887-905.
66. Xiao, F. 2019. Multi-sensor data fusion based on the belief divergence measure of evidences and the belief entropy. *Information Fusion* 46:23-32.
67. Xiao, F., and B. Qin. 2018. A Weighted Combination Method for Conflicting Evidence in Multi-Sensor Data Fusion. *Sensors (Basel, Switzerland)* 18.
68. Xiao, F. Y. 2017. A Novel Evidence Theory and Fuzzy Preference Approach-Based Multi-Sensor Data Fusion Technique for Fault Diagnosis. *Sensors* 17:20.
69. Xu, X., H. Zhang, M. Luo, Z. Tan, M. Zhang, H. Yang, and Z. Li. 2019. Research on target echo characteristics and ranging accuracy for laser radar. *Infrared Physics & Technology* 96:330-339.
70. Yao, X. M., S. B. Li, and J. J. Hu. 2017. Improving Rolling Bearing Fault Diagnosis by DS Evidence Theory Based Fusion Model. *Journal of Sensors*:14.
71. Zadeh, L. A. 1965. Fuzzy sets. *Information and Control* 8:338-353.
72. Zaidner, G., and A. Shapiro. 2016. A novel data fusion algorithm for low-cost localisation and navigation of autonomous vineyard sprayer robots. *Biosystems Engineering* 146:133-148.
73. Zheng, H., A. M. Cai, Q. Zhou, P. T. Xu, L. C. Zhao, C. Li, B. J. Dong, and H. C. Gao. 2017. Optimal preprocessing of serum and urine metabolomic data fusion for staging prostate cancer through design of experiment. *Analytica Chimica Acta* 991:68-75.
74. Zhou, D. J., T. T. Wei, H. S. Zhang, S. X. Ma, and F. Wei. 2018. An Information Fusion Mode Based on Dempster-Shafer Evidence Theory for Equipment Diagnosis. *Asce-Asme Journal of Risk and Uncertainty in Engineering Systems Part B-Mechanical Engineering* 4:8.

BIOGRAPHICAL INFORMATION

Mohammad Amin Javadi was born in Iran, received his Bachelor`s from Shahid Beheshti University and Master`s from Islamic Azad University (South-Tehran branch) both in Industrial Engineering. He is currently working on obtaining his PhD degree from the University of Texas at Arlington. Mohammad Amin`s research interests include data/sensor fusion, analytics and simulation in manufacturing and production processes.